Парадигма развития науки

Методологическое обеспечение

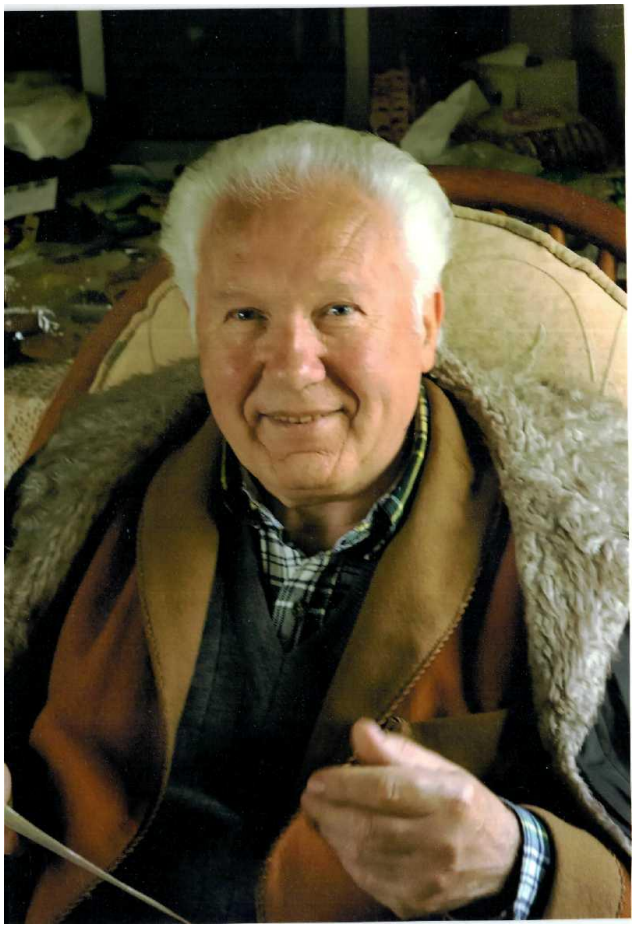
А. Е. Кононюк

ОБЩАЯ ТЕОРИЯ КОММУНИКАЦИЙ

Книга 1

Функции, предписания, директивы

Киев
Освіта України
2014



Кононюк Анатолий Ефимович

УДК 51 (075.8)
ББК В161.я7
К213

Рецензент:
Н. К. Печурин — д-р техн. наук, проф. (Национальный ави—ационный университет).

Кононюк А. Е.
К213 Общая теория коммуникаций. — В 4-х кн. Кн. 1.— К.: Освіта України. 2014. — 488 с.

ISBN 978-966-373-693-8 (многотомное издание)
ISBN 978-966-373-694-5 (книга 1)

В предлагаемом многотомном научно-учебном издании предпринята попытка раскрыть с единой идеологической и методологической точки зрения чрезвычайно сложную научную проблему - общую коммуникаций. В работе достаточно полно, на взгляд автора, раскрыты методы системного подхода и системного анализа, используемые при анализе и синтезе различного класса коммуникационных систем и сетей. Определяются основные (базовые) понятия теории коммуникационных систем и сетей, раскрывается содержание системного анализа, его технология.

Для бакалавров, специалистов, магистров, аспирантов, докторантов всех специальностей.

УДК 51 (075.8)
ББК В161.я7

ISBN 978-966-373-693-8 (многотомное издание)
ISBN 978-966-373-694-5 (книга 1)

Оглавление

Введение

Можно утверждать, что в социально-научном знании происходит формирование новой парадигмы - коммуникативной парадигмы. Двадцатый век был веком математико-компьютерной лингвистики, и можно ожидать, что двадцать первый век будет веком коммуникаций. В этом утверждении отражается, несмотря на его категоричность, смелая попытка отрефлексировать место коммуникативного знания как отдельной обеспечивающей науки в среде системы наук.

В становлении современной теории коммуникации эволюционно можно выделить несколько этапов, или "поколений", привнесших свои особенности в понимание феномена коммуникации. Несмотря на то, что исследования, которые относят к коммуникативным, проводились на протяжении всего двадцатого века в рамках различных социальных наук (Shramm, 1997; Craig, 1999; Littlejohn, 2002], после второй мировой войны, одновременно с развитием информационных наук и технологий, стали оформляться новые подходы к изучению феномена коммуникации. Характерным для этих подходов было рассмотрение коммуникации как *передачи информации*, где необходимыми составляющими признавались источник информации, сообщение, получатель информации, канал передачи и шум. Пример - "присвоенная" коммуникативными дисциплинами математическая модель коммуникации Шеннона и Вивера (Shannon and Weaver, 1948), описывающая коммуникацию как процесс передачи и восприятия информации от одного источника к другому (Craig, 1999). Модель была модифицирована в коммуникативной литературе и получила широкое распространение как модель информационной трансмиссии (*information transmission model*). По сей день, несмотря на признанные ограничения, акцентирующие линейный, однонаправленный характер коммуникации, эта модель широко используется в коммуникативной теории и практике, особенно в массовой коммуникации и в публичных выступлениях (Craig, 1999; Miller, 2002; Griffin, 2003).

В теории коммуникации 50-60-х годов происходит "психологизация" коммуникации, когда она начинает все больше рассматриваться как человеческая коммуникация. Формируется отрасль дисциплины, известная ныне как межличностная коммуникация. Заметный вклад в развитие этого направления вносят антропологи, психологи, психиатры и психотерапевты, среди них антрополог Грегори Бейтсон (Gregory

Bateson) и калифорнийская группа исследователей, *Palo Alto Group*, во главе с психотерапевтом Полем Вацлавиком (Watzlawick et al., 1967). В их подходах коммуникация рассматривается прежде всего как *взаимодействие*, признается, что каждый из участников оказывает влияние на ход этого взаимодействия. Подчеркивается, что коммуникация есть не просто передача-прием информации, но создание некой общности, некой степени взаимопонимания между участниками. Акцентируется необходимость обратной связи (*feedback*) и наложения сфер личного опыта (*fields of experience*) в создании этого взаимопонимания. В этой связи особое внимание отводится проблеме генерирования смысла (*meaning*) в коммуникативном взаимодействии. Выделяются два уровня смысла: содержательный (*content meaning*) и отношенческий (*relationship meaning*) (Watzlawick et al., 1967). Как отмечает один из современных теоретиков дисциплины Роберт Крейг (Craig, 1995), характерно, что активно возникающие в этот период кафедры межкультурной коммуникации испытывают явное влияние со стороны психологических наук и бихевиоризма, и проходит некоторое время, прежде чем межличностная коммуникация самоопределяется как отдельная от психологии дисциплина.

Следующий этап в становлении теории коммуникации, включающий и настоящее время, характеризуется расширением системного подхода к анализу коммуникации. **Коммуникация рассматривается как *социальный процесс*** (Pearce and Cronen, 1980; Pearce, 1989; Leeds-Hurwitz, 1995). Подчеркивается ее не просто интерактивный, но трансактный (*transactional*) характер, заключающийся в том, что любой субъект коммуникации является отправителем и получателем сообщения не последовательно, а одновременно, и что любой коммуникативный процесс включает в себя, помимо настоящего (конкретной ситуации общения), непременно и прошлое (пережитый опыт), а также проецируется в будущее. Несмотря на то, что большинство коммуникативных ситуаций имеет для нас четко определяемое начало и конец, то есть являются дискретными, мы, тем не менее, не знаем, где, когда, с кем и каким образом наш разговор с одним коммуникативным партнером может получить продолжение в наших отношениях-общениях с другими людьми. В этом смысле, мы участвуем в процессе общения бесконечно, и границы нашего "коммуницирования" не всегда можно четко определить (Miller, 2002; Wood, 2002).

Рассмотрим наиболее важные положения процесса человеческой коммуникации, которые можно обобщить следующим образом:

1. *Коммуникация - это основной социальный процесс сотворения, сохранения-поддержания и преобразования социальных реальностей (Communication is a primary social process of co-creating, maintaining and transforming social realities)* (Pearce and Pearce, 2000; Pearce, 1989; Pearce and Cronen, 1980). Коммуникация, и в более узком смысле, общение, не есть служебный процесс, не есть лишь средство достижения определенных целей (например, я вступаю в разговор, чтобы выразить уже существующие внутри меня мысли, чувства, отношения). **Коммуникация есть формирующий процесс**. **В самом фундаментальном смысле коммуникация есть состояние человеческого бытия (*human condition*), способ человеческого существования, тот основополагающий, первичный социальный процесс, в котором мы, его неизбежные участники, совместно создаем, воспроизводим и преобразуем наши социальные миры, качества нашего существования. Коммуникация, в этом смысле, есть сама жизнь, и, по известной аксиоме Поля Вацлавика, *one cannot not communicate* (Watzlawick et al., 1967).**

Тезис о социально-созидающей роли коммуникации считают фундаментальным для осмысления этого явления. Он помогает понять, что как формирующий, конституирующий процесс, коммуникация одновременно есть и социально-политический процесс. Наглядное подтверждение тому - состояние коммуникативных дисциплин на данный момент в науке и образовании. Является уже очевидным и признанным, что коммуникативная традиция достаточно нова в системы образования: **специальность "коммуникация" отсутствует пока в вузах**. Как подойти к проблеме? - Очевидно создавать вокруг нее общественно-политический дискурс (что и набирает силу в данный момент) - **наполнять интеллектуальный рынок новыми публикациями по коммуникативной тематике, формируя таким образом язык и "лицо" дисциплины; организовывать коммуникативные конференции; разрабатывать на кафедрах коммуникативные курсы; привлекать специалистов в области образования (*decision-makers*) к обсуждению вопроса и принятию соответствующих решений, то есть конституировать эту область знания и образования и, в конечном итоге, трансформировать социальную реальность .**

2. *Генерирование смысла: как социальные смыслы создаются (Generation of meaning: how social meanings are created*). Проблема создания смыслов в процессе коммуникации является ключевой для теоретиков этого направления (Cronen, 1995). Коммуникация - не

просто процесс обмена информацией, это процесс создания некой общности, в котором мы осмысливаем информацию и соотносим наши смыслы со смыслами наших коммуникативных партнеров, создавая таким образом определенную степень взаимопонимания. При этом происходит не столько самовыражение и передача-прием уже сформированных смыслов, сколько совместное смыслосозидание. Сколько бы мы, например, ни старались порой спланировать, проиграть в голове предстоящий с кем-либо разговор, реальный разговор никогда не разворачивается точно так, как мы его себе представляли. Реальное течение разговора конструируется высказываниями, реакциями на высказывания и осмыслениями этих реакций одновременно каждым из участников. Реальное течение разговора всегда динамично и диалогично, поэтому динамичен и диалогичен и смысл, который мы создаем в процессе разговора. Если высказывание, прозвучавшее в начале разговора, может первоначально восприниматься нами в одном смысле, то в свете каждого нового высказывания предыдущие смыслы, как правило, изменяются, уточняются, дополняются. Смыслы, таким образом, - постоянно изменяющиеся, "текущие" образования, *"values in the running", "always-in-process phenomena"* (Cronen, 1995).

Динамичность смыслов характерна не только для межличностного общения, но и для любого другого вида коммуникации, включая политический дискурс. Вспомним, например, как отшлифовывалось в средствах массовой информации называние того, что произошло в Нью-Йорке и Вашингтоне 11 сентября 2001 года. Первоначально американские СМИ использовали риторически бьющие, эмоциональные ярлыки: *"attack on America" , "war against America" , "horrific event"* и т.д. Понадобилось время, новая информация, публичное обсуждение и осмысление различных сторон этого не поддающегося однозначному определению события, прежде чем в социальном дискурсе установилось его нейтральное название: *"nine-eleven"*.

3. *Роль контекста в коммуникации, культура как наиболее общий контекст (The role of context in communication, culture being as the widest context)* (Leeds-Hurwitz, 1995; Griffin, 2003). Коммуникативные процессы протекают и приобретают для нас тот или иной смысл лишь в определенном контексте. Один и тот же разговор может иметь для нас разную значимость, наделяться разными смыслами в зависимости от того, рассматриваем ли мы его в контексте конкретного коммуникативного эпизода (где, когда, с кем, при каких условиях мы

вступили в разговор); либо в контексте отношенческом (кто мы друг для друга в этом разговоре, друзья, коллеги, соперники); либо в контексте той или иной культуры и культурных отношений (например, отношений полов, возрастов, отношений социально-экономических, профессиональных, этнических и т.д.).

4. *Восприятие своего "я" как социально-культурный конструкт (Identity as a social, culturally constructed view of the self).* Коммуникация - процесс, в котором мы конструируем не только свою социальную реальность (или реальности), но и свое собственное "я". Представители социальных подходов к коммуникации, продолжая традиции символического интеракционизма Джорджа Герберта Мида (George Herbert Mead) и Эрвина Гофмана (Erving Goffman), рассматривают "восприятие себя" не как фиксированное внутреннее образование, а как социально-культурный конструкт, постоянно модифицируемый в зависимости от того, с кем и как мы вступаем во взаимоотношения (Leeds-Hurwitz, 1995). Иначе говоря, с разными людьми, в том или ином кругу общения, я могу иметь разные представления о себе, разную самооценку как отражение мнений обо мне "обобщенного другого" (Mead, 1934).

5. *Символическая природа коммуникации (Symbolic nature of communication)* . **Коммуникация - процесс создания общих смыслов посредством использования символов, среди которых первоочередная роль принадлежит языку,** отсюда - признание значимости социолингвистических и семиотических исследований в социальных подходах к коммуникации (Leeds-Hurwitz, 1995; Griffin, 2003; Miller, 2002).

6. *Взаимовлияние исследователя и исследуемой им коммуникативной практики (Reflexivity between researcher and communication practice).* Сторонники социальных подходов разделяют положение о том, что коммуникативная теория - это "практическая теория" (Cronen, 1995) и что коммуникация - это "практическая дисциплина" (Craig, 1999). Исследователь, включенный в процесс исследования, испытывает на себе влияние исследуемой коммуникативной практики, и при этом он одновременно преобразует, формирует, "культивирует" эту коммуникативную практику (Pearce and Pearce, 2000; Pearce, 1995; Craig, 1995; Leeds-Hurwitz, 1995).

Данные положения, суммирующие современные социальные подходы к изучению коммуникации, важны, как концептуально, так и практически. Концептуально - поскольку дают представление о сложности и многомерности природы человеческой коммуникации как социального феномена. Практически - потому что ориентируют нас на то, что понять и объяснить эту многоплановость можно лишь интегрируя усилия и достижения всех социальных наук, всего знания о человеке.

В работе рассмотрены вопросы построения общих формализмов, математических моделей и алгоритмов вокально-знаковой межличностной коммуникации. При этом опущены описания специализированных и технических средств их реализации, так как они, по мнению автора, относятся не к общим вопросам, рассматриваемым в теории коммуникаций, а к конкретным предметным коммуникационным система и сетям.

1. Аксиоматико-множественный подход формирования основ теории коммуникациий

Существуют семантический и синтаксический способы задания логической теории, к которой будем относить и теорию коммуникаций.
Теория коммуникаций является основополагающей наукой существования и развития человеческого сообщества. Коммуникации подлежат материальные объекты (предметы) и/или энергия материальных объектов.

1.1. Базовые понятия теории коммуникаций

Семантические коммуникационные предписания являются шагом на пути к построению коммуникационных систем — систем, базирующихся на знании процессов коммуникаций. Семантические коммуникационные предписания являются основой большинства коммуникационных систем зрительного восприятия, понимания естественного языка и непрерывной речи, т. е. систем, осуществляющих связь с внешним миром — одним из главных источников знания коммуникационных систем (другим является сама коммуникационная система). Возможно, что именно характер развития и разноплановость использования семантических коммуникационных предписания послужит основной причиной разработки многочисленных вариаций этих коммуникационных предписания, базирующихся на нескольких общих идеях и методах реализации. Мы рассмотрим ряд основных принципов построения и характеристик семантических коммуникационных предписаний.
Основой всех вариантов семантических коммуникационных предписаний является *формализация структур семантического знания в виде направленного графа с помеченными вершинами и дугами*, при этом вершинам соответствуют некоторые объекты и/или энергии (предметы), а дугам — семантические отношения (связи) между этими предметами. Метки, приписываемые вершинам, носят ссылочный характер, представляя собой некоторые мнемонические имена, в частности, понятия естественного языка. В этом частном случае понятия дают ссылку на систему понятий, входам которой

соответствуют некоторые смысловые понятийные эквиваленты, или лексические значения (причем это соответствие может быть неоднозначным в обе стороны). Метки, приписанные дугам, соответствуют элементам множества отношений, заданных на графе, причем этими элементами могут быть как семантические свойства (признаки), так и семантические выводы (доказательства). Описанный выше граф мы будем называть *семантической коммуникационной сетью.*

На семантической коммуникационной сети могут быть определены некоторые подграфы определенной структуры, котрые будем называть *предписаниями (утверждениями, определениями)*. Каждый такой подграф представляет собой граф, корнем которого является *понятийнопредикатная вершина*; остальные вершины будем называть *концептуальными*. Такое разделение вершин предписания можно четко реализовать, лишь рассматривая каждое предписание в изоляции. В структуре семантической коммуникационной сети одна и та же вершина может быть понятийнопредикатной относительно одного предписания и концептуальной относительно другого. Мы вводим понятие предписания лишь с целью подчеркнуть, что оно является *минимальной единицей информации*, вводимой и хранящейся в семантической коммуникационной сети.

В семантических коммуникационных предписаниях будем использовать три базовых типа объектов: **понятия, события и свойства.**

Понятия являются константами или параметрами окружающего мира, описываемого (отображаемого) семантической коммуникационной сетью. Понятия, как правило, указывают (определяют) предметы (объекты, процессы, явления) или абстракции.

События представляют собой результат действия, которые произошли или могут произойти в окружающем мире. Если мы определим ***ситуацию*** как описание (отображение) части окружающего мира, то можно сказать, что все, что изменяет данную ситуацию, является событием.

Один из методов описания (отображения) событий состоит в указании изменений, которые событие производит, будучи применено к структуре коммуникационной сети, отражающей данную ситуацию.

Результатом события является некоторая ситуация, которую мы можем определить как **эталон (образец)** в некоторой процедуре, описывающей последовательность действий, приводящую к этой ситуации.

Свойства (предметов) будем использовать для уточнения, расширения или модификации понятий, событий или других свойств. Когда свойства представимы понятиями, то они могут быть особенностями, признаками, характеристиками, присущими или приписанными понятию. В случае событий свойства описывают (отображают) определенные общие, универсадьные, постоянные характеристики, например, место, время, длительность и т.д.

Предполагаемая теория коммуникаций должна заниматься вопросами познания (анализа), созидания (синтеза), применения и использования комукационных систем и сетей. Построение теории коммуникаций начнем с формирования базовых научнообразованых понятий и их определений .

К базовым понятиям в теории коммуникаций отнесем следующие: коммуникация, информация, понятие, объект, процесс, действие, система понятий.

Прежде чем сформулировать определение основного понятия теории коммуникаций, которым является понятие «коммуникация», приведем ряд отличительных признаков, характеризующих коммуникационный процесс.

Коммуникационный процесс должен:

- реализовывать заданные цели;
- решать поставленные задачи;
- выполнять присущие (свойственные) ему функции;
- состоять из заданных элементов (процедур, операций) и связей;
- иметь структуру;
- быть организованным согласно поставленных целей;
- выполнять управляющие функции (воздействия).

Исходя из перечня приведенных отличительных признаков коммуникационного процесса сформулируем научно-образованое определение понятия «коммуникация».

Коммуникация – процесс перемещения материальных объектов (вещей) и/или энергии по заданному (определенному) закону в заданном (определенном) направлении по заданной (определенной) траектории.

Трасса в Энциклопедическом словаре: Трасса - (от нем. Trasse - направление линии - пути)...1) линия,определяющая путь движения или продольную ось дороги, трубопровода и т. п.сооружения большой протяженности...

Маршрут (нем. *Marschroute*, от франц. *marche* — ход, движение вперёд и *route* — дорога, путь) — направление движения объекта, относительно определённых географических ориентиров и координат, с указанием основных пунктов.

Содержит начальный пункт старта и конечный. Обычно заранее намечается перед началом перемещения. Трасса маршрута является уже линией по которой пройдёт намеченное передвижение, установленный путь следования.

Будем различать внутрисистемные коммуникации (коммуникационные системы) и межсистемные коммуникации (коммуникационные сети).

Утверждение 1.

Коммуникационные процессы присущи (свойственны) только и только субъектам живой природы (животному и растительному миру).

Утверждение 2.

Процесс коммуникации может быть реализован только и только при наличии **отправителя** предмета коммуникации и **получателя (адресата)** предмета коммуникации

Утверждение 3.

Процесс коммуникации может быть реализован только и только между **отправителем** предмета коммуникации и **получателем** предмета коммуникации.

Декарт утверждал, что все ***вещи*** делятся на ***протяженные*** («res exstensa») и ***мыслимые*** («res cogitans»), а следовательно, изучение (и использование) их возможно либо в модусе физики, либо в модусе психологии.

Предметы (протяженные вещи) и коммуникационные процессы характеризуются (описываются) различного рода величинами, их значениями и интервалами значений.

Слово – замкнутая последовательность знаков (символов). ***Смысловое слово*** – замкнутая последовательность знаков (символов), несущая семантическую нагрузку (понятийный смысл).

Объект – сущность (вещь), обладающая формой (геометрической), характеризующаяся отличительными свойствами (признаками) и находящаяся в состоянии.

Процесс – последовательность действий.

Действие – приложенная одним субъектом (объектом), изменяющаяся во времени по величине и/или направлению, сила к другому субъекту (объекту).

Воздействие. Под *воздействием* будем понимать целенаправленное (ограниченное, интервальное, сосредоточенное, сконцентрированное) действие, выполняемое в (за) заданный промежуток времени.

Процесс функционирования коммукационной системы (сети) - есть последовательность изменения ее состояний, упорядоченных по времени.

Формальное описание процесса функционирования коммукационной системы (сети) есть четверка:

$$Z=<T, S, F, \alpha>$$

где: T - время;

S - пространство состояний;

F – траектория процесса ($F:T\rightarrow S$);

α – отношение линейного порядка на множестве T.

Математическое моделирование функционирования коммукационной системы (сети) осуществляется путем выполнения упорядоченной во времени последовательности логически взаимосвязанных событий (действий, процедур, операций).

Коммуникационные процессы в теории коммуникаций представляется лингвистическим описанием и математическими моделями научно-образованных понятий и понятийных выражений.

Коммуникационное поле – ограниченная область пространства в котором существуют и функционируют коммуникационные сети.

Коммуникационная сеть – вещественно-энергетический объект, в котором осуществляется коммуникация.

Коммуникационные процессы представимы методологиями, методами, способами, правилами, операторами, технологиями, алгоритмами, процедурами, операциями, приемами, формализованное описание которых будем осуществлять средствами математики и формальной логики, в частности, методами исчисления высказываний и исчисления предикатов.

Коммуникационная среда – вещественное пространство, в котором возможна реализация коммукационных процессов.

Методология – это учение о методах и процедурах научной деятельности, а также раздел общей теории познания, в особенности теории научного познания (эпистемологии) и философии науки. **Методология**, в прикладном смысле, — это система (комплекс, взаимосвязанная совокупность) принципов и подходов исследовательской деятельности, на которые опирается исследователь (учёный) в ходе получения и разработки знаний в рамках конкретной дисциплины — физики, химии, биологии и других научных дисциплин.

Метод – совокупность приемов (процедур) или операций для получения искомого результата.

Процесс - (от лат. processus — продвижение), последовательная смена (действий) состояний стадий развития.

Способ (как объект коммуникации) – процесс выполнения взаимосвязанных действий, необходимых для достижения поставленной цели.
Способ отличается от принципа действия объекта тем, что в нем отсутствует причинно-следственная связь между опреациями и приемами, которые объединены лишь общей задачей. В этом плане можно сказать, что способ - это совокупность последовательно осуществляемых операций, между которыми отсутствует причинно-следственная связь и которые объединены лишь общей решаемой задачей

Правило – требование для исполнения неких условий (норма на поведение) всеми участниками какого-либо действия (игры, правописания, коммуникационного процесса, организации, учреждения),

Оператор – какие-то особые (для данной области знаний) отображения, например, в функциональном анализе под операторами понимают отображение (правило), ставящее в соответствие функции другую функцию («оператор на пространстве функций» звучит лучше, чем «функция от функции»).

Технология – (от др.-греч. τέχνη — искусство, мастерство, умение; λόγος — мысль, причина; методика, способ производства) — совокупность методов, процессов и материалов, используемых в какой-либо отрасли деятельности, а также научное описание способов коммуникации.

Технологический процесс – совокупность и последовательность процедур и/или опреаций

Алгоритм - предписание, позволяющем по каждому *исходному данному,* или *аргументу, или* некоторой совокупности *возможных* (для данного алгоритма) *исходных данных* (аргументов) получить *результат* в случае, если таковой существует, или не получить ничего в случае, если для рассматриваемого исходного данного не существует результата (подчеркнем, что возможные исходные данные — это не те данные, в применении к которым алгоритм дает результат, а те, к которым можно его применять (возможно, безрезультатно).) Если для выбранного исходного данного результат существует, будем говорить, что алгоритм *применим* к этому исходному данному и *перерабатывает* его в этот результат.
Процедура - взаимосвязанная последовательность действий.
Операция – совокупность целенаправленных действий, совершаемые для достижения поставленной (определенной) цели.

Прием – целенаправленное действие, совершаемое для достижения поставленной (определенной) цели.

Цель коммуникации – обеспечение субъектов живой природы (флоры и фауны) выполнение (реализация) процессов обмена объектами (субъектами) и/или энергией.
Будем различать:
- внутрисистемные комуникационные процессы;
- межсистемные (сетевые) комуникационные процессы

Правда – завершенный процесс отображения истины.
Ложь – осознанная неправда.
Заблуждение – неосознанная неправда.
Представимость понятия:
- звуком,
- мимикой,
- жестами,
- словами на естественном и искусственном языках ,
- знаками.

Отображения межличностной коммуникации:
- средствами речи (речевая коммуникация),
- средствами аккустики (аудио коммуникация),
- средствами отображения на материальных носителях.
- визуальными средствами (формой, светом, цветом).
- тактильными средствами.
- средствами обаняния (газоанализационная коммуникация).
- органолептическими средствами (вкусовая коммуникация).

Модели понятий, используемые в теории коммуникаций:
- лингвистическая (текстовая),
- формализованная (аналитико-формульная)

1.2. Базовые аксиомы теории коммуникаций

В теории коммуникаций при синтаксическом способе формирования моделей коммуникационных систем (сетей) и их фрагментов кроме алфавита и формул, определяемых в семантическм способе формирования моделей коммуникационных систем (сетей) и их фрагментов, задаются **аксиомы и правила вывода.**

Аксиомами называют некоторое выделенное множество формул теории. Обычно существует возможность эффективно выяснить, является ли данная формула теории А аксиомой. В таком случае А называется *аксиоматической теорией.*

В основе теории коммуникаций лежат понятия «сущность» и «связь». Руководствуясь аксиоматической теорией (подходом), формально

построим совокупность базовых понятий теории коммуникаций на основании следующих аксиом.

Аксиомы существования понятий.

1. Существует по крайней мере одно универсальное понятие описания окружающего мира – сущность, собирательное понятие, некоторая абстракция реально существующего объекта предметной области, процесса или явления.
2. Существует по крайней мере одно множество сущностей. Сущности это начальные, простейшие объекты теории коммуникаций.
3. Существует по крайней мере одно базовое понятие – объект.
4. Существует по крайней мере одно множество объектов.
5. Существует по крайней мере одно базовое понятие – процесс.
6. Существует по крайней мере одно множество процессов.
7. Существует по крайней мере одно базовое понятие – явление.
8. Существует по крайней мере одно множество явлений.
9. Существует по крайней мере одно базовое понятие – коммуникация,
10. Существует по крайней мере одно универсальное понятие - связь, посредством которого осуществляется фиксирование взаимотношений между сущностими.
11. Существует по крайней мере одно множество связей, задающееся множеством отношений.
12. Существует такое множество понятий $\varnothing$, что ни одно понятие ему не принадлежит (это аксиома существования пустого множества).
13. Существет по крайней мере одно универсальное понятие – **теория**, (греч. $\theta\varepsilon\omega\rho\acute{\iota}\alpha$ — рассмотрение, исследование) — учение, система идей или принципов. (Совокупность научных идей взятых в их единстве как результат изучения отдельной проблемной области)
14. Существует по крайней мере одно множество теорий, в том числе, теория коммуникаций.

Аксиома объемности.

Если множества понятий M_a и M_b состалены из одних и тех же понятий, то они совпадают (равны): $M_a = M_b$

Аксиматический подход формирования аксиом **операций** над множествами понятий.

Аксиома объединения. Для поизвольных множеств понятий M_a и M_b существует множество понятий, понятиями которого являются все понятия множества M_a и все понятия множества M_b и которое никаких других понятий не содержит.

Из аксиом объемности и объединения понятий следует, что для поизвольных множеств понятий M_a и M_b множество понятий, удовлетворяющее условиям аксиомы объединения единственно. Действительно, если были бы два таких множества понятий M_{c_1} и M_{c_2}, то они содержали бы одни и те же понятия (все понятия, принадлежащие множеству понятий M_a и все понятия, принадлежащие множеству понятий M_b) и поэтому, согласно аксиоме объемности, $M_{c_1} = M_{c_2} = M_c$. Будем называть это единственное множество понятий M_c *объединением* множеств понятий M_a и M_b и будем писать

$$M_c = M_a \cup M_b$$

Аксиома разности. Для произвольных множеств понятий M_a и M_b существует множество понятий, понятиями которого являются те и только те понятия множества M_a, которые не являются понятиями множества понятий M_b.

Из аксиом объемности и разности заключаем, что для произвольных множеств понятий M_a и M_b существует в точности одно множество понятий, содержищее понятия множества понятий M_a, не принадлежащие множеству понятий M_b. Назовем это множество понятий M_c *разностью* множеств понятий M_a и M_b

$$M_c = M_a \setminus M_b$$

Для каждого множества понятий M существует множество понятий, элементами которого являются подмножества понятий множества понятий M и только они. Такое множество будем называть *семейством множества понятий* M или *булеаном* этого множества понятий и обозначать $B(M)$, а множество понятий M - *универсальным, универсумом* или *пространством* и обозначать U.

Аксиома степени. Для каждого множества понятий M существует семейство множеств понятий $B(M)$ (булеан), понятиями которого являются все подмножества понятий M_i, $M_i \subset M$, и только они.

Аксиоматический подход позволяет формально на основаниии введенных аксиом определить понятия и операции алгебры множеств понятий в теории коммуникаций.
С помощью операций объединения и разности, используя введенные аксиомы, определим еще три операции на множествах понятий.

Рассмотрим алгебру множеств понятий, которая будет использоваться нами при формировании (описании) моделей коммуникационных систем (сетей). Эта алгебра имеет вид

$$A_k = < B(U), \bigcup, \bigcap, \bar{} >$$

носителем которой является *булеан* универсального множества понятий U, *сигнатурой* - операции объединения $\bigcup$, пересечения $\bigcap$ и дополнения $\bar{}$ понятий. Для операций алгебры понятий выполняются следующие законы:

коммутативности объединения и пересечения

$M_a \bigcup M_b = M_b \bigcup M_a$, $M_a \bigcap M_b = M_b \bigcap M_a$;

ассоциативности объединения и пересечения

$M_a \bigcup (M_b \bigcup M_c) = (M_a \bigcup M_b) \bigcup M_c$,

$M_a \bigcap (M_b \bigcap M_c) = (M_a \bigcap M_b) \bigcap M_c$;

дистрибутивности пересечения относительно объединения и объединения относительно пересечения

$M_a \bigcap (M_b \bigcap M_c) = M_a \bigcap M_b \bigcup M_a \bigcap M_c$

$M_a \bigcup (M_b \bigcap M_c) = (M_a \bigcup M_b) \bigcap (M_a \bigcup M_c)$;

идемпотентности объединения и пересечения

$M_a \bigcup M_a = M_a$, $M_a \bigcap M_a = M_a$;

действия с универсальным U и пустым $\varnothing$ множествами

$M \bigcup \varnothing = M$, $M \bigcap \varnothing = \varnothing$, $M \bigcup U = U$, $M \bigcap U = M$, $M \bigcup \bar{M} = U$,

$M \bigcap \bar{M} = \varnothing$;

двойного дополнения

$\bar{\bar{M}} = M$.

С помощью операций объединения и разности, используя введенные аксиомы, определим еще три операции на множествах понятий.
Пересечение множеств понятий M_a и M_b определяется формулой
$M_a \bigcap M_b = M_a \backslash (M_a \backslash M_b)$.

Можно показать, что понятиями пересечения $M_a \bigcap M_b$ являются те и только те понятия, которые принадлежат как множеству понятий M_a, так и множеству понятий M_b.

Дополнение $\overline{M}$ множества понятий M определяется формулой
$\overline{M} = U \backslash M$.

Симметрическая разность множеств понятий M_a и M_b определяется формулой
$M_a \backslash \cdot M_b = (M_a \backslash M_b) \bigcup (M_b \backslash M_a)$.

На основании введенной аксиоматики можно доказать справедливость следующих законов:

закон дистрибутивности пересечения относительно разности
$M_a \bigcap (M_b \setminus M_c) = M_a \bigcap M_b \backslash M_a \bigcap M_c$;

закон коммутативности симметрической разности

$M_a \backslash^{\cdot} M_b = M_b \backslash^{\cdot} M_a$;

закон ассоциативности симметрической разности

$M_a \backslash (^{\cdot} M_b \backslash^{\cdot} M_c) = (M_a \backslash^{\cdot} M_b) \backslash^{\cdot} M_c$;

закон дистрибутивности пересечения относительно симметрической разности

$M_a \bigcap (M_b \backslash^{\cdot} M_c) = M_a \bigcap M_b \backslash^{\cdot} M_a \bigcap M_c$;

законы склеивания
$M_a \bigcap M_b \bigcup M_a \bigcap \overline{M}_b = M_a$, $(M_a \bigcup M_b) \bigcap (M_a \bigcup \overline{M}_b = M_a$;

законы поглощения
$M_a \bigcup M_a \bigcap M_b = M_a, M_a \bigcap (M_a \bigcup M_b) = M_a$.

1.3. Минимизации представления множества понятий в теории коммуникаций

Используя приведенные выше законы, рассмотрим задачу минимизации представления множества понятий M с помощью операций $\cup, \cap, \bar{}$.

Под *сложностью представления множества понятий M* будем понимать число понятий M_i, $\bar{M}_i$ в задающем его выражении (определении). Пусть в понятийном пространстве $U = \{M_1, M_2, M_3\}$ задано множество понятий вида

$M(M_1, M_2, M_3) =$

$= \bar{M}_1 \cap \bar{M}_2 \cap \bar{M}_3 \cup \bar{M}_1 \cap \bar{M}_2 \cap M_3 \cup \bar{M}_1 \cap M_2 \cap \bar{M}_3 \cup$

$\cup M_1 \cap \bar{M}_2 \cap \bar{M}_3 \cup M_1 \cap M_2 \cap \bar{M}_3 \cup M_1 \cap M_2 \cap M_3.$

Используя законы идемпотентности, коммутативности и ассоциативности объединения получаем

$M(M_1, M_2, M_3) =$

$= (\bar{M}_1 \cap \bar{M}_2 \cap \bar{M}_3 \cup \bar{M}_1 \cap \bar{M}_2 \cap M_3) \cup (\bar{M}_1 \cap M_2 \cap \bar{M}_3 \cup$

$\cup \bar{M}_1 \cap M_2 \cap \bar{M}_3) \cup (\bar{M} \cap \bar{M}_2 \cap \bar{M}_3 \cup M_1 \bar{M} M_2 \cap \bar{M}_3 \cup$

$\cup (M_1 \cap \bar{M}_2 \cap \bar{M}_3 \cup M_1 \cap M_2 \cap \bar{M}_3) \cup (M_1 \cap M_2 \cap \bar{M}_3 \cup M_1 \cap$

$\cap M_2 \cap M_3$

Используя законы коммутативности пересечения и склеивания, имеем

$M(M_1, M_2, M_3) =$

$= \bar{M}_1 \cap \bar{M}_2 \cup \bar{M}_1 \cap \bar{M}_3 \cup \bar{M}_2 \cap \bar{M}_3 \cup M_1 \cap \bar{M}_3 \cup M_1 \cap M_2.$

Используя законы коммутативности объединения и пересечения и закон склеивания, получаем

$M(M_1, M_2, M_3) = \bar{M}_1 \cap \bar{M}_2 \cup \bar{M}_3 \cup \bar{M}_2 \cap \bar{M}_3 \cup M_2 \cap M_3.$

Используя законы коммутативности пересечения и поглощения, имеем $M(M_1, M_2, M_3) = \bar{M}_1 \cap \bar{M}_2 \cup \bar{M}_3 \cup M_1 \cap M_2.$

Сложность представления заданного множества понятий уменьшилась от 21 до 5. Последовательность применения законов будем называть *стратегией преобразований множества понятий*. Сложность представления множества понятий, получаемого в результате применения этих законов (каждый из которых определяет эквивалентное преобразование), зависит от используемой стратегии.

Найдем стратегию, всегда порождающую минимальное предписание заданного множества понятий. Рассмотрим алгебру понятий $A_k=< B(U), \bigcup, \bigcap, \bar{} >$ и определим множества понятий, которые могут быть порождены (образованы) из произвольных подмножеств понятий $M_1, M_2,...,M_n$, называемых порождающими или образующими пространства U посредством выполнения операций $\bigcup, \bigcap, \bar{}$.

Множество понятий

$$M_i^{\sigma_i} = \begin{cases} M_i & при \quad \sigma_i = 1, \\ \bar{M}_i & при \quad \sigma_i = 0, \end{cases} \qquad i = 1, 2, ..., n.$$

в дальнейшем будем называть *первичным понятийным термом.*
Множество понятий вида

$$\bigcap_{i=1}^{n} M_i^{\sigma_i} = M_1^{\sigma_1} \cap M_2^{\sigma_2} \cap ... \cap M_n^{\sigma_n}, \quad \sigma_i = 0, 1,$$

назовем *конституентой*.

Известно, что общее число различных конституент не превышает 2^n. Каждой конституенте можно сопоставить двоичный набор длины n, число этих наборов равно 2^n. Если некоторые конституенты равны $\varnothing$, то общее количество конституент меньше 2^n, при этом среди подмножеств понятий найдутся хотя бы два такие, которые можно выразить одно через другое, т.е. зависимые. Например, если $n = 2$ и $M_2 = \overline{M_1}$, то существуют только две отличные от $\varnothing$ конституенты

$$\varnothing = M_1^0 \bigcap M_2^0 = M_1^1 \bigcap M_2^1,$$
$$C_1 = M_1^0 \bigcap M_2^1, C_2 = M_1^1 \bigcap M_2^0.$$

Утверждение 1. *Пересечение двух различных конституент пусто.*
Действительно, если конституенты

$$C_a = \bigcap_{i=1}^{n} M_i^{\sigma_i} \quad и \quad C_b = \bigcap_{i=1}^{n} M_i^{\sigma_i^*}$$

различны, то $\sigma_k \neq \sigma_k^*$ по крайней мере для одного k, $k \leq n$. Но тогда $M_k^{\sigma_k} \bigcap M_k^{\sigma_k^*} = \varnothing$ и, следовательно, $C_a \bigcap C_b = \varnothing$.

Утверждение 2. *Объединение всех конституент равно U.*
Представим U в виде

$$U = \bigcap_{i=1}^{n} \left(M_i^{\,0} \cup M_i^{\,1} \right)$$

и, раскрыв скобки, в правой части равенства получим объединение всех конституент.

Утверждение 3. *Множество понятий M_i - равно объединению конституент, каждая из которых содержит M_i^1.*

Согласно утверждению 2,

$$U = C_1 \cup C_2 \cup ... \cup C_i = \bigcup_{i=1}^{l} C_i,$$

где C_i, i=l, 2,...,l, - конституента. Определим пересечение левой и правой частей этого выражения с M_i. Имеем

$$M_i = (M_i \cap C_1) \cup (M_i \cap C_2) \cup ... \cup (M_i \cap C_i),$$

Если C_j содержит в качестве аргумента пересечения M_i^0, то $M_i \cap C_j = \emptyset$. Если же C_j содержит M_i^1, то $C_j \cap M_i = C_j$. Следовательно, M_i - объединение тех конституент, которые содержат M_i^1 в качестве сомножителя.

Теорема 1. *Каждое непустое множество понятий, которое образовано из множеств понятий $M_1, M_2,...,M_n$ с помощью операций $\cup, \cap, \bar{}$, является объединением некоторого числа конституент.*

Согласно утверждению 3, теорема справедлива для множеств понятий $M_1, M_2,...,M_n$. Следовательно, достаточно доказать, что если произвольные множества понятий M_a и M_b представимы в виде объединения некоторого числа конституент, то и множества понятий $M_a \cup M_b$, $M_a \cap M_b$ и $\overline{M}$, если они непустые, также можно представить в виде объединения конституент.

Пусть множества понятий M_a и M_b представимы в виде объединения конституент $M_a = C_{a_1} \cup C_{a_2} \cup ... \cup C_{a_k}$ и $M_b = C_{b_1} \cup C_{b_2} \cup ... \cup C_{b_s}$ ... Тогда множество понятий $M_a \cup M_b$ можно представить в виде объединения конституент.

Согласно закону дистрибутивности,

$$M_a \cap M_b = (C_{a_1} \cap C_{b_1}) \cup ... \cup (C_{a_k} \cap C_{b_s}),$$

при этом если $C_{a_\alpha} \neq C_{b_\beta}$ то, согласно утверждению 1, $C_{a_\alpha} \cap C_{b_\beta} = \emptyset$, в противном случае $C_{a_\alpha} = C_{b_\beta}$. Следовательно, пересечение $M_a \cap M_b$ либо пусто, либо представимо в виде объединения конституент.

Докажем, что множество понятий $\overline{M}$ также представимо в виде объединения конституент, если

$$M = C_1 \cap C_2 \cap \ldots C_k.$$

Согласно закону де-Моргана,

$$\overline{M} = \overline{C_1 \cup C_2 \cup \ldots \cup C_k} = \overline{C_1} \cap \overline{C_2} \cap \ldots \cap \overline{C_k} =$$
$$\overline{M_1^{\sigma_{11}} \cap M_2^{\sigma_{12}} \cap \ldots \cap M_n^{\sigma_{1n}}} \cap$$
$$\overline{M_1^{\sigma_{21}} \cap M_2^{\sigma_{22}} \cap \ldots \cap M_n^{\sigma_{2n}}} \cap \ldots$$
$$\cap \overline{M_1^{\sigma_{k1}} \cap M_2^{\sigma_{k2}} \cap \ldots \cap M_n^{\sigma_{kn}}} =$$
$$(\overline{M_1^{\sigma_{11}}} \cup \overline{M_2^{\sigma_{12}}} \cup \ldots \cup \overline{M_n^{\sigma_{1n}}} \cap$$
$$(\overline{M_1^{\sigma_{21}}} \cup \overline{M_2^{\sigma_{22}}} \cup \ldots \cup \overline{M_n^{\sigma_{2n}}} \cap \ldots$$
$$\cap (\overline{M}_1^{\sigma_{k1}} \cup \overline{M}_2^{\sigma_{k2}} \cup \ldots \cup \overline{M}_n^{\sigma_{kn}}).$$

Раскрывая скобки и используя соотношения $M_\alpha \cap \overline{M}_\alpha = \varnothing$, $M_\alpha \cup \overline{M}_\alpha = U$, а также добавляя в те пересечения, в которых отсутствует нижний индекс β, сомножитель $M_\beta \cup \overline{M}_\beta$, получаем, что множество $\overline{M}$ также представимо в виде объединения конституент.

Теорема 2. *Из n множеств понятий в алгебре* $A=<B(U), \cup, \cap, \bar{}, >$ *можно образовать не более чем* 2^{2^n} *множеств понятий.*

Каждое множество понятий M, согласно теореме 1, является объединением конституент, число которых не превышает 2^n; следовательно, число различных объединений не превышает 2^{2^n}. При этом если множества понятий $M_1, M_2, \ldots, M_n$ независимы, т.е. все конституенты отличны от пустого множества, то число различных конституент равно 2^n и число множеств понятий, которые образованы с этих конституент в виде их объединения, равно 2^{2^n} (с учетом пустого множества понятий).

Введение понятия конституенты позволяет задавать множество понятий M при фиксированных независимых подмножествах понятий $M_1, M_2, \ldots, M_n$ универсального множества понятий U в виде объединения конституент:

$$M = \bigcup_{j} \bigcap_{i=1}^{n} M_i^{\sigma_i}.$$

Каждое фиксированное множество понятий $M_i \in U$ разбивает пространство понятий на две частей: на собственно M_i и на $\overline{M}_i$. При независимых множествах понятий $M_i \in \{M_i / i=1,...,n\}$ пространство понятий разбивается на $\underbrace{2 \times 2 \times ... \times 2}_{n\ раз} = 2^n$ областей. Каждая область является пересечением n множеств понятий M_i или $\overline{M}_i$, i=l,...,n. Сопоставим этой области двоичный вектор (σ_1,σ_2, ..., σ_n), в котором σ_i =1, если в пересечение $C = \bigcap_{i} M_i^{\sigma_i}$ входит M_i , и σ_i=0, если входит $\overline{M}_i$, а также десятичный эквивалент

$$d(C) = \sum_{i=1}^{n} \sigma_i \cdot 2^{i-1}.$$

Любое множество понятий M в пространстве U можно задать в виде объединения этих областей. Сопоставим множеству понятий M двоичный вектор длины 2^n, в котором i-му разряду отвечает область с десятичным эквивалентом, равным i. Вектор, который определяет множество понятий, представим в виде десятичного эквивалента:

$$d(M) = \sum_{i=0}^{2^n-1} c_i \cdot 2^i, \quad c_i = 0,1$$

Следовательно, множество понятий M в пространстве может быть задано в виде соответствующего десятичного эквивалента.

Рассмотрим, например, в трехмерном пространстве U={M_1,M_2,M_3} множество $M(M_1,M_2,M_3)$ с десятичным эквивалентом $d(M)$=217. Имеем

$$217 = 1 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0.$$

Множеству понятий M отвечает двоичный вектор (1,1,0,1,1,0,0,1), который определяет включение областей в множество понятий M (см. рис.1).

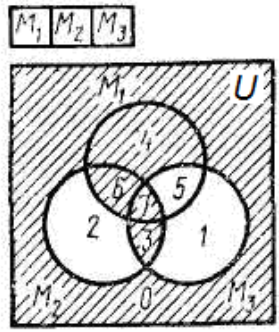


Рис. 1. Множественное пространство U, заданное кругами Эйлера

Кроме диаграммы Эйлера пространство может быть задано в виде гиперкуба или *n*-мерного куба (*n*-размерность пространства, равная числу фиксированных множеств).

Гиперкубом (n-мерным кубом) называется граф *H,* каждая вершина которого взаимно однозначно соответствует области пространства, и две вершины соединены ребром, если они соответствуют соседним областям (имеющим общую границу). Сопоставленные этим областям двоичные векторы отличаются в одном и только одном разряде.

Гиперкуб для рассматриваемого примера изображен на рис. 2 (вершины, соответствующие конституентам множества *M,* заштрихованы).

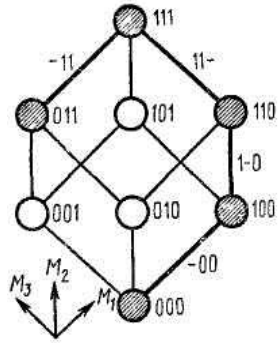


Рис. 2. Гиперкуб, представленный графом

Множество *M* можно задавать в виде двоичной таблицы, каждой строке которой взаимно однозначно соответствует конституента. Множество строк таблицы линейно упорядочено по возрастанию десятичного эквивалента соответствующего двоичного набора.

Столбцам соответствуют множества, образующие пространство, последний столбец сопоставляется множеству *M,* и единица указывает на вхождение соответствующей конституенты в множество *M*.
В данном случае имеем табл. 1.

Таблица 1

$d(C)$	M_1	M_2	M_3	M
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

Аналитически множество *M* задается в виде

$$M = \bar{M}_1 \cap \bar{M}_2 \cap \bar{M}_3 \cup \bar{M}_1 \cap M_2 \cap M_3 \cup M_1 \cap \bar{M}_2 \cap \bar{M}_3 \cup M_1 \cap M_2 \cap \bar{M}_3 \cup M_1 \cap M_2 \cap M_3$$

или в виде мографа

$G^M = < V, S_3 >$,

$V = \{M_i, \bar{M}_i / i = 1, 2, 3\}$, $S_3 \subset V^3$

$$S_3 = \{\underbrace{\{\bar{M}_1, \bar{M}_2, \bar{M}_3\}}_{1}, \underbrace{\{\bar{M}_1, M_2, M_3\}}_{2},$$

$$\underbrace{\{M_1, \bar{M}_2, \bar{M}_3\}}_{3}, \underbrace{\{M_1, M_2, \bar{M}_3\}}_{4}, \underbrace{\{M_1, M_2, M_3\}}_{5}\}$$ (рис. 3).

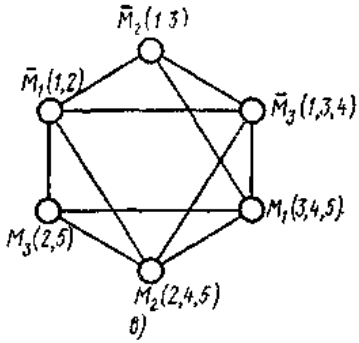


Рис. 3.

В рассматриваемой алгебре понятий $A=<B(U), \cup, \cap, \bar{} >$ операции являются зависимыми. Действительно, согласно закону де-Моргана, любое множество из 2^{2^n} множеств может быть построено и с помощью алгебры $A=<B(U), \cup, \bar{} >$. Равносильными в смысле порождения любого множества из 2^{2^n} множеств являются алгебры $A=<B(U), \cup, \bar{} >$, $A=<B(U), \cap, \bar{} >$, которые могут быть заменены соответственно алгебрами $A=<B(U), \cup, \setminus, U >$, $A=<B(U), \cap, \setminus, U >$ согласно формуле $\bar{M} = U \setminus M$, где универсум U рассматривается как нуль-местная операция.

Алгебра $A=<B(U), \cup, \setminus, U >$ в силу равенства

$$M_a \cup M_b = M_a \setminus^{\cdot} M_b \setminus^{\cdot} (M_a \cap M_b)$$

$$M_a \setminus M_b = M_a \setminus^{\cdot} (M_a \cap M_b)$$

может быть заменена алгеброй вида $<B(U), \cap, \setminus^{\cdot}, U >$.

Рассмотрим задачу минимизации представления множеств понятий в алгебре $A=<B(U), \cup, \cap, \bar{}, >$. Пересечение попарно различных множеств понятий $\cap M_i^{\sigma_i}$ будем называть *элементарным*. Выражение, задающее множество понятий M^i в виде объединения различных элементарных пересечений, будем называть *нормальной формой* (НФ) *множества понятий М*. Объединение конституент множества понятий *М* будем называть *совершенной* НФ *множества понятий М*.

Минимальной НФ *множества понятий М* будем называть НФ этого множества понятий, имеющая минимальную сложность.

Рассмотрим метод, который будем использовать для получения минимальной НФ множества понятий *М*. Этот метод заключается в последовательном выполнении таких этапов.

1. Выделение максимальных интервалов. *Интервалом множества понятий М* называется множество конституент множества понятий *М,* образующих гиперкуб (некоторой размерности).

Мощность интервала равна степени 2 (т. е. 2^0, 2^1 и т. д.).

Запишем, например, множество интервалов для рассмотренного выше примера: {000, 100, 110, 011, 111, -00, 1-0, 11-, -11}. Здесь и далее « —» означает, что множество понятий, соответствующее этому разряду, в пересечении отсутствует, т. е. по этому множеству после объединения соответствующих конституент произошло склеивание. Например, интервал -00, соответствующий множеству конституент 000 и 100, получается в результате преобразования

$$\bar{M}_1 \cap \bar{M}_2 \cap \bar{M}_3 \bigcup M_1 \cap \bar{M}_2 \cap \bar{M}_3 = M_2 \cap M_3.$$

Интервал I_α называется *максимальным интервалом* $I_{\text{макс}}$ *множества* M, если не найдется другою интервала I_β этого множества, содержащего интервал I_α, $I_\alpha \not\subset I_\beta$.

В данном случае имеется четыре максимальных интервала:
— 00, 1—0, 11—, —11; каждый из них образует гиперкуб размерности 1 (ребро).

Пересечение $\bigcap_i M_i^{\sigma_i}$ соответствующее максимальному интервалу множества M, называется *простой импликантой* этого множества.

Объединение простых импликант множества M называется *сокращенной* НФ *множества* M.

Количество первичных понятийных термов, образующих простую импликанту, называется *рангом простой импликанты,* а элементарное пересечение — *рангом соответствующего интервала*.

При выделении максимальных интервалов множество интервалов, имеющих один и тот же ранг, будем разбивать на пояса, причем i-й пояс содержит интервалы, которым соответствуют наборы с i единицами в каждом. Тогда выделение максимальных интервалов сводится к сравнению элементов только соседних поясов, номера которых отличаются на единицу. Если построенные интервалы не являются максимальными, то процесс сравнения продолжают.

Результаты сравнения для рассматриваемого случая приведены на рис. 4.

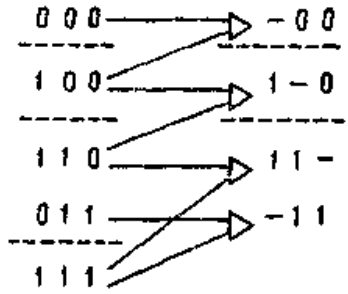


Рис. 4

Сокращенная НФ множества понятий $M(M_1, M_2, M_3)$ имеет вид

$$M(M_1, M_2, M_3) = \bar{M}_2 \cap \bar{M}_3 \bigcup M_1 \cap \bar{M}_3 \bigcup M_1 \cap M_2 \bigcup M_2 \cap M_3.$$

Построением сокращенной НФ множества M заканчивается первый этап метода.

Тупиковой НФ *множества М* называется такая НФ этого множества, которая при вычеркивании хотя бы одного первичного понятийного терма не определяет *М*.

Утверждение 4. *Минимальная* НФ *множества понятий М является тупиковой*.

Сложность минимальной НФ множества понятий *М* нельзя уменьшить вычеркиванием первичного понятийного терма. Следовательно, эта форма является тупиковой.

Утверждение 5. *Тупиковая* НФ *множества понятий М состоит из простых импликант этого множества*.

Если хотя бы одно пересечение соответствует интервалу множества *М,* не являющемуся максимальным, то это пересечение можно заменить простой импликантой вычеркиванием соответствующих первичных понятийных термов, не выходя из класса эквивалентных НФ (задающих одно и то же множество понятий) множества *М,* что противоречит определению тупиковой НФ.

Теорема 3. *Тупиковая* НФ *множества понятий М, в том числе и минимальная* НФ, *содержится в сокращенной* НФ *этого множества понятий*.

Тупиковая НФ множества понятий *М,* в том числе и минимальная НФ, состоит, согласно утверждению 5, из простых импликант. Сокращенная НФ множества *М* включает все простые импликанты. Следовательно, тупиковая (минимальная) НФ множества *М* содержится в сокращенной НФ этого множества.

Согласно теореме 3, построение тупиковой НФ множества *М* сводится к покрытию двумерной таблицы.

Покрытием столбцов строками в двумерной таблице называется такое множество строк, при котором для каждого столбца найдется хотя бы одна строка из этого множества, на пересечении с которой этот столбец имеет единицу, причем при вычеркивании хотя бы одного элемента из этого множества строк указанное свойство не выполняется.

2. Построение и покрытие таблицы Квайна. *Таблица Квайна* — двумерная таблица, каждой строке которой взаимно однозначно соответствует максимальный интервал, столбцу — конституента, а на пересечении *i*-й строки и *j*-го столбца находится единица, если *j*-я конституента входит в *i*-й максимальный интервал, в противном случае клетку *(i, j)* не заполняют или ставят в ней 0.

Для рассматриваемого примера таблица Квайна имеет вид:

Таблица 2

Максимальный интервал	Конституента				
	000	100	110	011	111
–00	1	1			
1–0		1	1		
11–			1		1
–11				1	1

Максимальный интервал называется *обязательным,* если найдется конституента, принадлежащая ему и только ему. Множество обязательных интервалов образует *ядро покрытия.*

В данном случае ядром покрытия является множество {-00, -11}, которое покрывает первый, второй, четвертый и пятый столбцы. Для образования покрытия можно взять либо вторую, либо третью строку. В результате получаем два покрытия: {-00, -11, 1-0}, {-00,-11, 11-}; каждое из них является минимальным и имеет сложность 6. Для определенности выберем первое из покрытий, которое соответствует минимальной НФ, задающей множество

$$M(M_1, M_2, M_3) = \bar{M}_2 \cap \bar{M}_3 \cup M_2 \cap M_3 \cup M_1 \cap \bar{M}_3.$$

В результате упрощения сложность *LM)* уменьшилась от 15 до 6.

Минимальная НФ находится в результате перебора всех покрытий, осуществляемого с помощью преобразования мультипликативно-аддитивной формы в аддитивно-мультипликативную форму.

Для рассматриваемого примера идентифицируем четыре строки табл. 2 соответственно буквами *a, b, c, d*. Запишем множество строк, каждый элемент которого покрывает *j-й* столбец:

$j = 1 \rightarrow A_1 = \{a\}$, $j = 2 \rightarrow A_2 = \{a, b\}$,

$j = 3 \rightarrow A_3 = \{b, c\}$, $j = 4 \rightarrow A_4 = \{d\}$, $j = 5 \rightarrow A_5 = \{c, d\}$,

Покрытием столбцов строками этой таблицы является множество строк, покрывающее все столбцы таблицы, и при удалении хотя бы одной из этих строк найдется непокрытый столбец. Следовательно, если каждое множество A_j представить в виде объединения ее элементов и найти пересечение всех множеств A_j, $\bigcap_j A_j$, то каждое пересечение в полученной аддитивной форме соответствует покрытию, а число всех покрытий равно числу различных пересечений в полученной аддитивно-мультипликативной форме:

$$\bigcap_j A_j = a \cap (a \cup b) \cap (b \cup c) \cap d \cap (c \cup d) = a \cap (b \cup c) \cap d =$$

$$= a \cap b \cap d \cup a \cap c \cap d.$$

Полученные пересечения $a \cap b \cap d$ и $a \cap c \cap d$ порождают два покрытия: {-00, 1-0, -11} и {-00, 11-, -11}; каждое из них соответствует минимальной НФ заданного множества *M*.

Дальнейшее уменьшение сложности выражения, определяющего заданное множество, возможно, если из класса НФ перейти в класс скобочных форм (СФ). Выражение, определяющее множество *M,* называется *скобочной формой,* если кроме первичных термов и знаков операций объединения и пересечения в него входят скобки (,).

В рассматриваемом примере сложность представления множества, равная 6, понижается до 5 в результате применения закона дистрибутивности пересечения относительно объединения

$M(M_1, M_2, M_3) = \bar{M}_3 \cap (M_1 \cup \bar{M}_2) \cup M_2 \cap M_3$.

Преобразование мультипликативно-аддитивной формы в аддитивно-мультипликативную называется *методом Петрика,* который может быть определен соответствующим *алгоритмом.*

Интуитивное (наивное) определение коммуникации. Совокупность правил, обладающих свойствами *массовости* (инвариантность относительно входной информации), *детерминированности* (однозначность применения этих правил на каждом шаге), *результативности* (получение после применения этих правил информации, являющейся результатом) и *элементарности* (отсутствует необходимость дальнейшего уточнения правил), называется ***коммуникацией***.

Отображения межличностной (социальной) коммуникации :

- средствами речи (речевознаковая коммуникация),
- средствами звука,
- средствами написания на материальных носителях.
- визуальными средствами (формой, светом, цветом).
- тактильными средствами.
- средствами обаяния,
- средствами, определяющие вкус предмета.

2. Методы представления и решения коммуникационных задач в коммуникационных системах и сетях

2.1. Исходные понятия

Выделяем два аспекта, учитывемых при формировании коммуникационных задач, — способность хранить, накапливать, извлекать, обобщать, анализировать и корректировать знания (познания) о коммуникациоых процессах *(эпистемологический аспект (подход))* и способность использовать знания о коммуникациоых процессах вместе с поставленной целью для нахождения эффективных решений коммуникационных задач *(эвристический аспект (подход)*.

Таким образом, знание фактов относительно различных сторон функционирования коммуникациоых процессов, с которыми работает коммуникационня система, является ключевым фактором для построения коммуникационных систем и сетей.

Общая система знаний может рассматриваться как состоящая из знания о внешнем мире (модель внешнего мира), абстрактного знания (мир философии, математики и т. д.) и знания о знаниях активных источников действия в мире (силы природы или сама коммуникационня система). Существенно отметить, что мы включаем в знание набор *универсальных и специализированных методов решения коммуникационня задач* (РКЗ). Таким образом, система создания коммуникационных систем должна обладать способностью хранить, извлекать, обобщать, анализировать и корректировать методы РКЗ как часть знания. Система знаний, организованная соответствующим образом, составляет *внутренний мир (внутреннее содержание)* коммуникационной системы.

Форма совокупного выражения коммуникационных процессов и условий решаемой коммуникационной задачи в системе создания коммуникационных систем называется *предписанием*. Перед нами стоит задача формировать такие предписания, которые, с одной стороны, были бы достаточно общими, т. е. допускали описание широкого класса коммуникаций и задач, решаемых этими коммуникациями. С другой стороны, предписания должны допускать использование мощных методов в отношении как качества решения

коммуникационных задач, так и потребных для решения ресурсов. В общем случае требования общности предписания и мощности методов РКЗ являются противоречивыми. Существование обратной зависимости между общностью и мощностью приводит к тому, что в попытках построить общий решатель коммуникационных задач мы будем вынуждены снабдить его общими и потому относительно слабыми методами решения.

В рамках указанной качественной зависимости формирование предписания является весьма важным фактором, определяющим как простоту описания коммуникационной задачи, так и эффективность ее решения. В силу такого двойственного характера предписания имеет смысл описать эпистемологические и эвристические свойства предписания.

Эпистемологически полным предписанием, или предписанием в широком смысле, назовем совокупность формализмов для описания всех фактов о мире, необходимых для выполнения определенного класса коммуникационных задач.

Пример эпистемологически неполного предписания: естественный язык эпистемологически неполон для описания хранения знаний в человеческом мозгу или для описания сложных визуальных образов.

Эпистемологически адекватным предписанием называется предписание, которое можно практически использовать для выражения фактов относительно какого-то аспекта знаний о коммуникационных системах (сетях).

Пример эпистемологически полного, но неадекватного предписания: предписание осуществления коммуникационных процессов с использованием 40-разрядного двоичного регистра в виде конечного автомата.

Эвристически адекватным предписанием называется предписание, которое допускает лингвистическое выражение последовательности заданных директив, приведшей к решению коммуникационной задачи.

Пример эпистемологически полного и адекватного, но эвристически неадекватного предписания представление зрительных образов путем двоичного кодирования «черного» и «белого» на рецептивном поле эпистемологически полно и адекватно, но эвристически неадекватно, т. е. не может быть использовано для распознавания образов.

Общий план использования методов для решения поставленной коммуникационной задачи будем называть *стратегией*.

Эвристически эффективной стратегией называется стратегия, направленная на оптимизацию необходимых для решения коммуникационной задачи ресурсов.

Предписание и стратегия составляют две основные и взаимосвязанные компоненты процесса РКЗ, причем эвристически эффективная стратегия определяется относительно заданного эвристически адекватного предписания, а формирование эвристически адекватного предписания должно осуществляться с ориентацией на построение эвристически эффективных стратегий.

Обратная зависимость общности и мощности приводит к качественному заключению о том, что чем более специализированным является эвристически адекватное предписание, тем с большей вероятностью мы сможем построить эвристически эффективную стратегию.

Поэтому, если рассматривать эпистемологически полное предписание главным образом как инструмент для описания фактов о знании коммуникационных систем (сетей) и для постановки коммуникационной задачи, процесс решения должен включать в себя последовательное преобразование предписаний, начиная от эпистемологически полного предписания и кончая таким специализированным эпистемологически и эвристически адекватным предписанием, в котором может быть определена эвристически эффективная стратегия.

Под *предписанием в узком смысле,* или собственно предписанием, будем понимать эпистемологически и эвристически адекватное предписание, описанное в рамках единого формализма и ориентированное на построение эвристически эффективных стратегий решения подкласса коммуникационных задач относительно класса, определяемого эпистемологически полным предписанием.

2.2. Проблема формирования предписаний

2.2.1. Общий подход

Несмотря на то, что проблема формирования предписаний — одна из важнейших для построения по-настоящему «разумных» и эффективных решателей коммуникационных задач, до сих пор не предложено сколько-нибудь удовлетворительной теории, способствующей решению этой проблемы.

Основным подходом к проблеме формирования предписаний при решении коммуникационных задач является разработка средств *глобального исследования пространства поиска решений коммуникационных задач,* результатом которого явилось бы преобразование этого пространства в другое или меньшего размера

и/или обладающего специфическими свойствами, полезными для решения данной коммуникационной задачи. Мы употребляем термин «глобальное исследование», чтобы отличить его от «локального исследования» пространства поиска решения, естественно происходящего в процессе РКЗ.

Среди стандартных целей такого глобального исследования следует выделить:

1) Обнаружение свойств симметрии, избыточности или подобных обобщенных отношений в пространстве поиска решений коммуникационных задач, ведущих к сокращению пространства.

2) Переформулировка коммуникационной задачи путем обобщения элементов начального предписания и отождествления полученных макроэлементов с элементами (директивами) в новом предписании.

3) Разделение общей топологии пространства на «легкопроходимые», критические и запрещенные области с последующим переопределением элементов (директив) предписания.

Успех глобального исследования пространства поиска решений коммуникационных задач определяется в основном опытом, накопленным решателем задач при попытках решения задачи в исходном представлении. Другими словами, этот подход к решению задачи преобразования предписпний следует рассматривать как *пошаговый процесс,* эволюционирующий от каждого текущего предписания к более эффективному, исходя из информации о задаче (или классе задач), которая появляется при попытках решить задачу в текущем представлении. Мы рассмотрим этот подход на примере, а затем укажем на некоторые общие методы глобального исследования пространств поиска решений коммуникационных задач

2.2.2. Пример

Рассмотрим задачу коммуникации миссионеров и людоедов (МЛ). Словесная формулировка задачи (эпистемологически полное предписание) выглядит следующим образом.

Элементарная постановка. Отправитель получил заказ на осуществление (выполнение) коммуникации получателю трех миссионеров и трех людоедов с левого берега на правый берег реки. Для осуществление (выполнение) этой коммуникации имеется лодка, вмещающая не более двух человек (в любом сочетании миссионеров и людоедов). Если число людоедов на любом берегу превысит число миссионеров, то миссионеры будут съедены. Найти простейший план

коммуникаций, безопасный для миссионеров и такой, что три миссионера и три людоеда окажутся на правом берегу у адресата.

Обобщенная постановка отличается от элементарной тем, что имеется N миссионеров и N людоедов, а лодка вмещает k ($k \geq 2$) человек. Число людоедов не должно превышать числа миссионеров на любом берегу и в лодке.

Мы начнем с представления задачи МЛ в элементарной *системе продукций*. Общая постановка задачи в этой системе выглядит следующим образом: даны начальная ситуация, конечная ситуация, множество возможных предписаний (директив) в пространстве ситуаций и условия, определяющие применимость предписаний (директив) в той или иной ситуации. Ситуация в системе продукций описывается перечнем ее основных признаков, называемым *N-состоянием*. Требуется найти наилучшую коммуникацию в виде последовательности допустимых преобразований предписаний (директив) из начального состояния в конечное. Пусть S — множество всех возможных N-состояний, $\{A\}$ — конечное множество правил преобразования предписаний (директив). Множество $\{A\}$ задает *отношение непосредственной достижимости* T между элементами S. Для $s_x, s_y \in S$ $s_x T s_v$ тогда и только тогда, когда существует допустимое $A \in \{A\}$, преобразующее s_x в s_v, где под допустимым понимается предписание, удовлетворяющее условиям применимости в N-состоянии s_x. Путь из s_a в s_b есть конечная последовательность $s_1, s_2, \ldots, s_m$, $s_1 = s_a$, $s_2 = s_b$ такая, что для всех i, $1 < i \leq m$, $s_{i-1} T s_i$. Состояние s_b *достижимо* из s_a ($s_a \Rightarrow s_b$) тогда и только тогда, когда или $s_a = s_b$, или существует путь (коммуникация) из s_a в s_b. Наконец, множество всех N-состояний, частично упорядоченных отношением T, называется *пространством N-состояний* σ. В этом пространстве мы и осуществляем поиск решающего пути.

Переходя к задаче МЛ, введем следующие обозначения: $\{m_i / i = 1, 2, \ldots, N\}$ —множество миссионеров, $\{c_i / i = 1, 2, \ldots, N\}$ — множество людоедов, b_k — лодка с максимальной емкостью k, $p_л$, $p_п$ — левый и правый берег реки соответственно, $M_л$, $M_п$, $M_в$ — количество миссионеров на левом берегу, правом берегу и в лодке соответственно, $C_л$, $C_п$, $C_в$ — количество людоедов на левом берегу, правом берегу и в лодке соответственно. Определим следующие отношения:

At(x_i, p) указывает, что объект x_i, имеющий область значений $(m_i, 1 \leq i \leq N; c_i, 1 \leq i \leq N; b_k)$, находится *на берегу* p, p принимает значения $p_л$ и $p_п$;

On (y_i, b_k) указывает, что объект y_i, имеющий область значений $(m_i, 1 \leq i \leq N; c_i, 1 \leq i \leq N)$, находится *в лодке* b_k.

В первом предписании задачи МЛ ситуация отождествляется с описанием мест всех y_i (миссионеров и людоедов) и лодки. Например, начальное состояние

s_o==At(b_k, $p_л$), At(m_1; $p_л$), ...,

At (m_N, $p_л$), At (c_1, $p_л$), . . ., At (c_N, $p_л$). (2.1)

Конечное состояние определяется тем же предписанием, с подстановкой всюду $p_п$ вместо $p_л$.

Множество возможных предписаний (директив) A_1 записывается следующим образом:

$$
\left.
\begin{array}{l}
а)\ \textit{Загрузить лодку на левом берегу}\ (ЗЛЛ): \\
\textit{для любого}\ y_i \\
\alpha, \mathrm{At}(b_k, p_л), \mathrm{At}(y_i, p_л); (M_в + C_в \le k-1) \to \\
\to \alpha, \mathrm{At}(b_k, p_л), \mathrm{On}(y_i, b_k); \Lambda. \\
б)\ \textit{Перевезти лодку с левого берега на правый}\ (ПЛЛП): \\
\alpha, \mathrm{At}(b_k, p_л); (M_в + C_в > 0) \to \alpha, \mathrm{At}(b_k, p_п); \Lambda. \\
в)\ \textit{Выгрузить лодку на правом берегу}\ (ВЛП): \\
\textit{для любого}\ y_i \\
\alpha, \mathrm{At}(b_k, p_п), \mathrm{On}(y_i, b_k); \Lambda \to \\
\to \alpha, \mathrm{At}(b_k, p_п), \mathrm{At}(y_i, p_п); \Lambda.
\end{array}
\right\} \quad (2.2)
$$

Аналогично определяются предписания для осуществления процесса коммуникации y_i с правого берега на левый (ЗЛП, ПЛПЛ, ВЛЛ). В выражениях (2.2) отдельные предписания (директивы), входящие в описание состояния, отделяются запятыми; ограничения, накладываемые на применимость предписаний, отделяются от описания состояния точкой с запятой, α — произвольная конфигурация, делающая описание состояния полным, Λ — пустое множество ограничений.

Заметим, что множество возможных предписаний (директив) A_1, определяемое (2.2), не учитывает ограничений на относительное число миссионеров и людоедов на берегах реки и в лодке.

Построим теперь множество правил предписания, которое, кроме учета этого ограничения, включает в себя некоторые макропредписания, определяемые как последовательности элементарных предписаний (директив) из A_1 *(обобщение элементов начального предписания,* а именно предписаний (директив); понятие состояния пока остается неизменным). Назовем это множество A_2:

$$\left.\begin{array}{l}
\textit{а) Загрузить r объектов } y_i \textit{ в лодку на левом берегу } (З^r ЛЛ): \\
\textit{для множества } \{y_i / i = 1,2,...,r, 1 \le r \le k\} \\
\alpha, \mathrm{At}(b_k, p_л), \mathrm{At}(y_1, p_л), ..., \mathrm{At}(y_r, p_л); (M_в + C_в = 0) \to \\
\to \alpha, \mathrm{At}(b_k, p_л), \mathrm{On}(y_1, b_k), ..., \mathrm{On}(y_r, b_k); \\
((M_л = 0) \vee (M_л \ge C_л)), ((M_в = 0) \vee (M_в \ge C_в)). \\
\textit{б) Перевезти r объектов } y_i \textit{ в лодке на правый берег и} \\
\textit{выгрузить всех их на правом берегу } (ПЛЛП + B^r ЛП): \\
\textit{для множества } \{y_i / i = 1,2,...,r, 1 \le r \le k\} \\
\alpha[e], \mathrm{At}(b_k, p_л), \mathrm{On}(y_1, b_k), ..., \mathrm{On}(y_r, b_k); \Lambda \to \alpha[e], \\
\mathrm{At}(b_k, p_п); \mathrm{At}(y_1, p_п), ..., \mathrm{At}(y_r, p_п); ((M_п = 0) \vee (M_п \ge C_п)).
\end{array}\right\} \quad (2.3)$$

Здесь α[*e*] — конфигурация α, ограниченная условием *e*: «ни одно выражение формы On *(y, bk)* не включено в α»; другими словами, все, кто сел на левом берегу, должны высадиться на правый берег.

По-прежнему мы аналогично определяем предписания для коммуникации $\{y_i\}$ с правого берега на левый, заменяя в предписаниях для $З^r$ЛЛ и ПЛЛП+B^rЛП $p_л$ на $p_п$ и $p_п$ на $p_л$ ($З^r$ЛП и ПЛПЛ+B^rЛЛ соответственно).

Преобразованием элементов определения $A_1 \to A_2$ мы делаем первый шаг на пути уменьшения размера пространства поиска решений σ, так как

1) количество промежуточных состояний уменьшается;

2) количество запрещенных состояний, т. е. состояний, в которых неприменимо ни одно из предписаний, увеличивается.

Исследуем на избыточность ограничения на применимость предписаний A_2, а именно покажем, что если в начале и в конце коммуникации с одного берега на другой условия $((M_л = 0) \vee (M_л \ge C_л))$ и $((M_п = 0) \vee (M_п \ge C_п))$ удовлетворяются, то $((M_в = 0) \vee (M_в \ge C_в))$ также удовлетворяется. По предположению удовлетворяются

$$\left.\begin{array}{l}((M_л = 0) \vee (M_л = C_л) \vee (M_л > C_л)) \\ ((M_п = 0) \vee (M_п = C_п) \vee (M_п > C_п))\end{array}\right\} \quad (2.4)$$

Легко видеть, что конъюнкция условий (2.4) эквивалентна

$$(M_л = 0) \vee (M_л = N) \vee (M_л = C_л), \quad (2.5)$$

так как $M_л + M_п = N, C_л + C_g = N$.

Для удовлетворения (2.5) необходимо, чтобы в лодке ехали либо только людоеды (сохранение $M_л$=0 и $M_л$=N), либо чтобы число миссионеров и людоедов в лодке было одинаковым (сохранение $M_л$=$C_л$), либо чтобы число миссионеров в лодке превышало число людоедов в лодке (переход от $M_л$ =N к $M_л$=$C_л$ или $M_л$=0, или переход от $M_л$=$C_л$ к $M_л$=0). Итак, удовлетворяется условие

$$(M_B = 0) \vee (M_B > C_B). \qquad (2.6)$$

Поскольку условие (2.6) является избыточным, мы можем провести еще одно обобщение предписания, переходя от множества предписаний F_2 к множеству A_3:

$$\left.\begin{array}{l} а)\ \textit{Перевезти } r \textit{ объектов } y_i \textit{ с левого берега на правый} \\ (П^r ЛП): \textit{для множества } \{y_i / i = 1,2,...,r, 1 \le r \le k\} \\ \alpha, \mathrm{At}(b_k, p_л), \mathrm{At}(y_1, p_л),..., \mathrm{At}(y_r, p_л); (M_в + C_в = 0) \to \\ \to \alpha, \mathrm{At}(b_k, p_л), \mathrm{At}(b_k, p_п); \mathrm{At}(y_1, p_п),..., \mathrm{At}(y_r, p_п); \\ (M_в + C_в = 0), ((M_л = 0) \vee (M_л \ge C_л)), ((M_п = 0) \vee (M_п \ge C_п)). \\ б)\ \textit{Перевезти } r \textit{ объектов } y_i \textit{ с правого берега на левый} \\ (П^r ПЛ)\ (\textit{получается, как и ранее, из выражения для } П^r ЛП \\ \textit{заменой } p_л \textit{ на } p_п \textit{ и } p_п \textit{ на } p_л. \end{array}\right\} \qquad (2.7)$$

Важно подчеркнуть, что исключение избыточных условий привело к обобщению предписаний A_2, т. е. к дальнейшему исключению промежуточных состояний и, в конечном итоге, к сокращению пространства поиска решений.

До сих пор мы не касались другого элемента предписания — N-состояния, которое выражалось в форме вида (2.1). Однако очевидно, что возможен переход от вида (2.1), вытекающего непосредственно из словесного описания задачи, к виду $(M_л, C_л, B_л)$, где $M_л$ и $C_л$ — целые числа, $0 \le M_л, C_л \le N$, а $B_л$ — булевская переменная, принимающая значение 1, если лодка находится на левом берегу, и 0, если на правом. Возможность такого перехода вытекает из условия e и очевидного соотношения

$$M_Л + M_П = C_Л + C_П = N.$$

Заметим, что выражение (2.1) для s_0 приобретает вид $(N, N, 1)$, а для конечного состояния s_t — (0, 0, 0). Переформулировка понятия состояния приводит к новому множеству предписаний A_4:

$$
\begin{array}{l}
а)\textit{Перевезти пару } (M_в, C_в) \textit{ с левого берега на правый} \\
(ПЛП,\ M_в,\ C_в)\text{: для всех пар}(M_в, C_в), \\
1 \le M_B + C_B \le k, \\
(M_л, C_л,\ 1);\ \Lambda \to (M_Л - M_B, C_л - C_в, 0); \\
\big((M_Л - M_B = 0) \vee (M_Л - M_B \ge C_Л - C_B)\big), \\
(((N - (M_л - M_B) = 0) \vee (N - (M_л - M_B) \ge N - (C_л - C_в))). \\
б)\textit{Перевезти пару } (M_в, C_в) \textit{ с правого берега на левый} \\
(ППЛ,\ M_в,\ C_в)\text{: для всех пар } (M_в, C_в), \\
1 \le M_B + C_B \le k, \\
(M_л, C_л,\ 0);\ \Lambda \to (M_Л + M_B, C_л + C_в, 1); \\
\big((M_Л + M_B = 0) \vee (M_Л + M_B \ge C_Л + C_B)\big), \\
(((N - (M_л - M_B) = 0) \vee (N - (M_л + M_B) \ge N - (C_л + C_в))).
\end{array}
\qquad (2.8)
$$

Переход $A_3 \to A_4$ избавляет нас от необходимости рассматривать отдельно каждого миссионера и людоеда, а представляет задачу в понятиях последовательности сложения и вычитания векторов, удовлетворяющей специальным условиям и преобразующей начальный вектор в конечный. Общее количество различимых состояний в пространстве σ от такого перехода не меняется, однако существенное упрощение вычислений, необходимых для идентификации каждого состояния, приводит к повышению эффективности РКЗ.

Рассмотрим предписание решения задачи о миссионерах и людоедах в так называемой *системе редукций*. В этой системе состояния (называемые далее *Р-состояниями)* отождествляются с предписаниями вида $S_i = (s_a \Rightarrow s_b)$, где s_a, s_b — *N*-состояния, а $\Rightarrow$ имеет тот же смысл достижимости, что и в системе продукций. *Нетерминальный ход* соответствует применению допустимого предписания к левому *N*-состоянию *Р*-состояния. Таким образом, например, к *Р*-состоянию $S_i = (s_a \Rightarrow s_b)$ применимо предписание s_a в s_c, и его применение соответствует ходу, редуцирующему S_i к $S_j = (s_c \Rightarrow s_b)$. Применение этого хода интерпретируется как «если s_b достижимо из s_c, то оно достижимо из s_a» (поскольку известно, что s_c достижимо из s_a). *Терминальный ход* в системе редукций просто распознает, что $(s_t \Rightarrow s_t)$.

Решение в системе редукций — это последовательность *Р*-состояний, достижимых последовательным применением нетерминальных ходов, начиная с начального состояния и кончая состоянием, где применяется терминальный ход.

Таким образом, в этой системе решение получается последовательностью разбиений текущей исходной задачи на пару задач, решение одной из которых известно, а решение другой предстоит найти следующим разбиением в последовательности. Можно видеть, что последовательность левых *N*-состояний из решающей последовательности *P*-состояний представляет собой решающий путь в системе продукций.

В задаче МЛ начальное *P*-состояние $s_o=((N, N, 1) \Rightarrow (0, 0, 0,))$, конечное состояние s_t =((0, 0, 0,) $\Rightarrow$ (0, 0, 0,)), ходы (ПЛП, $M_в$, $C_в$) и (ППЛ, $M_в$, $C_в$) определяются (2.8), терминальный ход обозначим A_t.

Часть графа решения задачи МЛ для случая N=3, k=2 приведена на рис. 2.1. Здесь *P*-состояния для простоты представлены своими левыми *N*-состояниями, ветви графа соответствуют ходам, стрелки указывают направления действия ходов. Решающий путь выделен жирной линией. Из рис. 2.1 видно, что решающий путь в графе *симметричен* относительно хода ППЛ, 1, 1 (*единственного* хода, дающего решение).

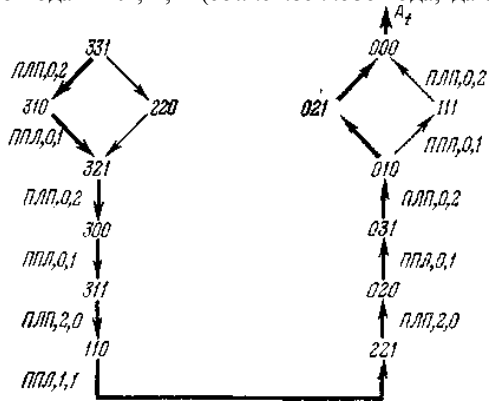


Рис. 2.1. Часть решающего графа для элементарной задачи о миссионерах и людоедах.

Введем отношение $\tilde{T}$, обратное отношению T непосредственной достижимости $(s_a T s_b \leftrightarrow s_b \tilde{T} s_a)$. Определим отображение θ между *N*-состояниями:

$$\theta: (M_л, C_л, B_л) \rightarrow (N\text{-}M_л, N\text{-}C_л, 1\text{-}B_л), \qquad (2.9)$$

или, в терминах векторной операции вычитания,

$$\theta\,(s)=(N, N, 1)\text{-}s. \qquad (2.10)$$

Следующее определение устанавливает отношение антиизоморфизма относительно θ между σ, пространством *N*-состояний, частично

упорядоченных отношением T, и $\tilde{\sigma}$, пространством N-состояний, частично упорядоченных отношением $\tilde{T}$.

Утверждение 2.1. *Для любой пары N -состояний* s_a, s_b,

$s_a T s_b \leftrightarrow \theta(s_a)\,\tilde{T}\,\theta(s_b)$, *или* $s_a T s_b \leftrightarrow \theta(s_b) T \theta(s_a)$.

Следствие 1 из утверждения 2.1. $(s_a \Rightarrow s_b) \leftrightarrow (\theta(s_b) \Rightarrow \theta(s_a))$.

Следствие 2 из утверждения 2.1. *Ход, порождающий переход из* s_a *в* s_b, *идентичен ходу, порождающему переход из* $\theta(s_b)$ *в* $\theta(s_a)$.

Установленные свойства пространств σ и $\tilde{\sigma}$ позволяют решать проблему *одновременно* в двух пространствах, т. е. «с начала» и «с конца», осуществляя поиск только с одной стороны (в силу следствия 2).

Мы вновь переформулируем понятие состояния, введя обобщенное P-состояние

$$\sum_i = (\{ s_i \} \Rightarrow \{\theta(s_i)\}), \quad i=0, 1,2, \ldots, j, \qquad (2.11)$$

i — число переходов между начальным или конечным N-состоянием и текущим обобщенным P-состоянием.

Нетерминальный ход в этом новом предписании (A_5) осуществляет переход в пространстве σ прямым поиском и параллельно вычисляемый на основе свойства симметрии переход в $\tilde{\sigma}$. Терминальный ход распознает, что $s_k T s_l$, $s_k \in \{s_j\}$, $s_l \in \{\theta(s_j)\}$. Решение имеет форму последовательности обобщенных P-состояний, начинающейся с $\sum_0 = (s_o \Rightarrow s_t)$ и заканчивающейся обобщенным P-состоянием, из которого осуществляется терминальный ход. Решающий коммуникационный путь в этой последовательности представляется последовательностью элементов (директив), выбираемых по одному из множеств

$$\{s_0\} = s_0, \{s_1\}, \ldots, \{s_j\}, \{\theta(s_j)\}, \ldots . \{\theta(s_2)\}, \{\theta(s_1)\}, \{\theta(s_o)\} = s_t.$$

Следующий предпринимаемый нами шаг по пути преобразования предписания — поиск некоторых *характеристических образов* в пространстве поиска решений. Поскольку к настоящему моменту количество возможных N-состоянии равно $2(N+1)^2$ нам необходим глобальный взгляд на пространство σ, позволяющий

1) отождествить с состояниями целые области этого пространства,

2) соответствующим образом обобщить предписания.

Мы представим пространство поиска решений в виде квадратного массива точек $(N+1)\times(N+1)$ (пример для элементарной задачи МЛ приведен на рис. 2.2).

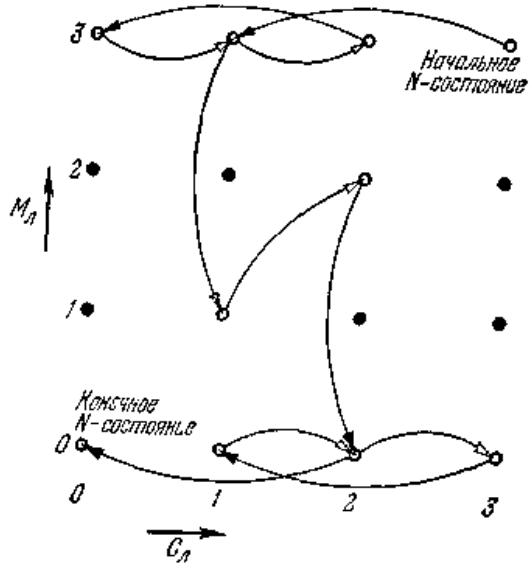


Рис. 2.2. Пространство *N*-состояний для элементарной задачи о миссионерах и людоедах.

Здесь стрелки с черным концом соответствуют коммуникации с левого берега на правый (ПЛП), с белым концом — коммуникации с правого берега на левый (ППЛ). Точка представляет собой в зависимости от цвета стрелки, с которой мы в нее вошли, одно из двух *N* -состоянии — ($M_л$, $C_л$, 1) (белая) или ($M_л$, $C_л$, 0) (черная). Решением в этом предписании является последовательность стрелок с чередующимся цветом концов, начинающаяся в точке *(N, N)* и кончающаяся в точке (0, 0). Предписание сформулировано в системе продукций, исходя из множества предписаний A_4.

Заметим, что зачерненные точки в пространстве обозначают запрещенные состояния, а остальные точки образуют зигзагообразную ломаную Z, состоящую из трех отрезков: $M_л=N$, $M_л=C_л$, $M_л=0$ в полном соответствии с (2. 5). Анализ Z-образа показывает (рис 2. 3), что

1) Любая точка *(N, x,* 1), $1 \le x \le N$, может быть достигнута из любой другой точки *(N, y,* 1), $1 \le y \le N$, некоторой горизонтальной последовательностью переходов

2) Любая точка *(N—x, N—x,* 0), $0 \le x \le k$, $k \ge 2$, на диагонали Z может быть достигнута из любой точки *(N, N—x,* 1), $0 < x \le k$, одним переходом (ПЛП, *x,* 0). В свою очередь из этой точки одним переходом (ППЛ, 1, 1) может быть достигнута точка *(N—x*+1, *N-x*+1, 1) на диагонали Z.

Эта пара представляет собой составной «стабильный прыжок» (необходимый для дальнейшего перехода на линию $M_л$=0). Итак, наиболее удаленная от начальной точка диагонали, которая может быть достигнута этой парой переходов, $(N—k+1, N—k+1, 1)$.

3) Точка на линии $M_л$=0 может быть достигнута из точки на диагонали, если расстояние от диагонали до линии не превышает k. Отсюда вытекает условие (с учетом симметрии Z)

$$k \geqslant \frac{N+1}{2}. \qquad (2.12)$$

4) Любая точка (0, x, 0), $0 \leq x < N$, может быть достигнута из любой другой точки (0, y, 0), $0 \leq y < N$, горизонтальной последовательностью переходов.

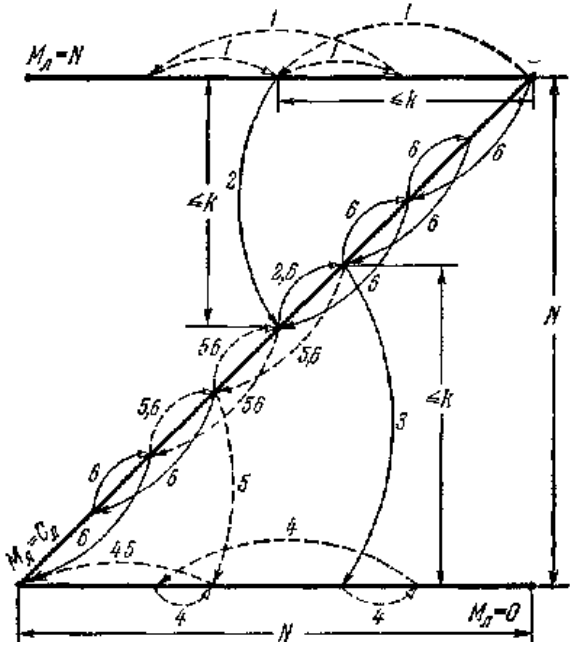


Рис. 2.3. Схемы решений обобщенной задачи о миссионерах и людоедах в Z области. Цифры у дуг соответствуют пунктам в тексте.

5) Условие (2.12) работает для k<4. В случае $k \geq 4$ разрешима любая задача МЛ (т. е. для любого N). Разрешимость вытекает из возможности построения диагональной последовательности типа (ПЛП, 2, 2), (ППЛ, 1, 1), (ПЛЛ, 2, 2), (ППЛ, 1, 1), . . ., обеспечивающей «спуск» по диагонали до точки $(k, k, 1)$.

б) Из изложенного в п. 5) ясно, что можно заменить Z-образ диагональным образом (для $k \geq 4$), т е. двигаться из начальной точки вниз по диагонали с помощью последовательных пар (ПЛП, $k/2$, $k/2$), (ППЛ, 1, 1) (в случае нечетного k - (ПЛП, $\frac{k-1}{2}$, $\frac{k-1}{2}$) , (ППЛ, 1,1)).

Изложенные результаты анализа позволяют сформулировать предписания в системе продукций, в котором множество предписаний A_6 представляет собой специфическое (и потому более мощное) множество макропереходов:

а) Горизонтальные переходы (Г):
Г1: (N, $C_л$, 1); $0<C_л< N$, $k \geq 2 \rightarrow (N,N, 1)$.
Г2: (0, $C_л$, 0); $0 \leq C_л < N, k \geq 2 \rightarrow$
$\rightarrow (0, C'_л, 0)$; $0 \leq C'_л \leq N$, $C_л \neq C'_л$.
б) Диагональные переходы (Д):
Д: ($M_л$, $C_л$, 1); $0 < M_Л = C_Л \leq N$, $k \geq 4 \rightarrow (0,0,0)$.
в) Прыжки (П): (2.13)
П: ($M_л$, $C_л$, 1); $0 < M_Л = C_Л \leq k \rightarrow (0, C_л , 0)$.
г) Составные переходы
(прыжки с горизонтали и диагонали):
ГП: (N, $C_л$, 1); $0<C_л \leq N, k \geq 2 \rightarrow$
$\rightarrow (N-k+1, N-k+1, 1)$.
ДП: ($M_л$,$C_л$, 1); $M_Л = C_Л > k \geq 4 \rightarrow (0, k, 0)$.

Множество A_6 обеспечивает решение задачи МЛ с помощью Z-образа, диагонального образа, а также смешанного образа, начиная из любого и кончая любым допустимым состоянием.

Введением множества предписаний A_6 мы существенно уменьшили размер пространства поиска решений, исключив значительное число запрещенных и промежуточных состояний. Новое пространство содержит начальное и конечное состояния, 4 состояния $(N, N, 1)$, $(N-k+1, N-k+1, 1)$, (0, 0, 0) и (0, k, 0), а также множество N-состояний $\{s/M_л=0, B_л=0, 0<C_л< k\}$, т. е. максимум $5+k$ состояний (вместо $2(N+1)^2$). Путем фиксации одного из трех используемых образов это пространство может быть еще сокращено.

Следует отметить, что новое пространство может быть разделено на три области ($M_л$=-0, $M_л = N$, $M_л = C_л$), которые «легко проходимы». Критическими местами являются точки перехода из одной области в другую.

Представляется весьма перспективным нахождение в пространстве поиска решений таких точек путем глобального исследования пространства. Это могло бы существенно повысить эффективность РКЗ, особенно в системе редукции.

Заканчивая исследование задачи МЛ, следует подчеркнуть, что оно, к сожалению, носит, как и большое число других исследований в области теории коммуникаций, характер *case study* (изучение конкретного опыта), т. е. исследования конкретного опыта, на базе которого сделаны некоторые обобщения.

В частности, становится ясным, что решение проблемы преобразования предписаний требует решения следующих основных вопросов:

1) Выбор основных элементов предписания (директив) в исходной коммуникационной системе (интерпретация состояний, ограничений и предписаний по заданному словесному или иному эпистемологически полному описанию).

2) Поиск полезных свойств, уменьшающих размер пространства поиска решений задачи или выделяющих в нем характерные критические точки.

3) Использование опыта исследования задачи для улучшения процесса РКЗ.

2.2.3. Методы исследований свойств пространства поиска решений коммуникационных задач

В предыдущем параграфе мы продемонстрировали на примере некоторые приемы, позволяющие путем глобального исследования пространства поиска решений коммуникационных задач существенно повысить эффективность решения задачи. В большинстве случаев, как уже отмечалось, такое исследование может быть проведено только для конкретной коммуникационной задачи, причем в общем случае оно еще не поддается полной автоматизации. Тем не менее можно указать некоторые общие методы, которые представляются перспективными с точки зрения их использования в решателях коммуникационных задач.

В настоящем параграфе мы рассмотрим два таких метода.

Первым методом является так называемый *метод образов* с последующим построением на основе образов приближенного плана решения коммуникационных задач.

Пусть задано, в духе предыдущего примера, множество состояний S и множество предписаний $A = \{A_j\}$, $A_j : S \rightarrow 2^S$.

Предположим, что нам заданы некоторые, хотя и не все, характеристики состояний. **Это могут быть системы признаков, характеризующих состояния, меры близости между состояниями или любая другая частичная информация о состояниях.** Таким образом, мы предполагаем, что на множестве состояний задана некоторая выделяющая функция $h: S \to I$, сопоставляющая каждому состоянию образ $h(s)$, и получаем множество образов I, причем каждый $i_k \in I$ представляет подмножество состояний $S(i_k) \subseteq S$. В идеальном случае можно было бы точно сгруппировать состояния в пространстве, т. е. $i_k, i_l \in I$ представляли бы такие подмножества состояний, что

$$S = \bigcup_{i \in I} S(i), \quad S(i_k) \cap S(i_l) = \varnothing.$$

Очевидно, что это соответствовало бы полному знанию свойств состояний в пространстве. В реальных случаях необходимо связать с функцией h некоторую меру неопределенности в виде вероятностных оценок или в духе теории размытых (нечетких) множеств. Можно рассматривать образы как состояния в некотором вспомогательном пространстве и использовать это вспомогательное пространство вместе с множеством предписаний A и выделяющей функцией h для составления планов решения. Если задать меру неопределенности перехода образа $i'=h(s_m)$ в образ $i=h(A_j(s_m))$ в виде $\mu(i, A_j, i')$, где s_m — некоторое состояние, $A_j \in A$ — предписание, то условия вида $\mu(i, A_j, i') \approx 1$ могут служить основой для стратегии, состоящей в построении плана в пространстве образов, затем решения в пространстве состояний в соответствии с планом, сверки их в точках, удовлетворяющих указанным выше условиям и соответствующей корректировке плана (а не решения) в случае неудачи.

Другой метод - это метод поиска запрещенных областей в пространстве путем *наследственных разбиений*. Идея этого метода состоит в том, чтобы показать, что из некоторого множества состояний целевое состояние недостижимо. Тогда эти состояния могут быть выделены по некоторым признакам (например, объединением в образы) в запрещенное для данного класса целевых состояний множество.

Построение такой опровергающей стратегии может быть проиллюстрировано на примере известной задачи Маккарти: задана шахматная доска, из которой удалены две противоположные клетки по диагонали. Доказать, что такая доска не может быть покрыта домино. Опровержение для этой задачи может быть получено следующим образом. Заметим, что когда мы кладем очередное домино на доску, число белых и черных квадратов, покрытых домино, остается равным.

Это свойство является *наследственным,* так как оно не меняется ни при одном допустимом ходе. Поскольку первоначальное число белых и черных квадратов не равно, то это неравенство не изменится.

Итак, для построения опровергающей стратегии необходимо показать, что

1) Для всех $S_i \subseteq S$, S — множество состояний, истинно свойство $P(S_i)$
2) Для всех s таких что $S_i \Rightarrow s$, также истинно $P(s)$.
3) Для конечного состояния s_t истинно $\sim P(s_t)$. Тогда s_t недостижимо из S_t и $S_i \bigcup \{s\}$ — запрещенное подмножество состояний.

Наследственные разбиения являются обобщением наследственного свойства. Стратегия этих разбиений получается следующим образом. Пусть $S_i \subseteq S$ может быть разбито на непересекающиеся подмножества, $S_i = \bigcup_j S_{ij}$ $S_{ik} \cap S_{il} = \emptyset$, $k \neq l$. Пусть также для всех j S_{ij} обладают свойствами $P_j(S_{ij})$. Эти же свойства удовлетворяются для всех множеств состояний $\sum_{ij}$ таких, что $S_{ij} \Rightarrow \sigma_{ij}$, $\sigma_{ij} \in \sum_{ij}$. В тоже время ни одно из целевых состояний $s_t \in S_t$ не обладает ни одним из свойств P_j, j=1, 2, . Тогда $S_i = \bigcup \left(\bigcup_j \Sigma_{ij} \right)$ — запрещенное множество состояний.

Заметим, что и в этом методе может оказаться необходимым связать с разбиением некоторую меру неопределенности, поскольку в реальной ситуации полная информация о свойствах всех состояний может оказаться недоступной.

Изложенные методы подчеркивают общий тезис о необходимости знания общих свойств пространства поиска решений для эффективного преобразования предписаний.

Возвращаясь к концу п. 2.2.2, отметим, чго наиболее принципиальным с точки зрения построения эффективных решателей коммуникационной задачи является выбор основных элементов предписания (директив) в исходном предписании, поскольку этот выбор определяет, в том числе, возможность эффективного преобразования предписаний. Это приводит к необходимости определения основных методов предписания и решения коммуникационных задач.

2.3. Краткая характеристика основных классов предписаний

Основные методы формирования коммуникационных предписаний будут основываться на идее понятийной предикации, т. е. **утверждения истинности или ложности тех или иных свойств в той или иной стадии процесса РКЗ**. В рамках этой основной идеи различные методы могут отличаться описательным аппаратом, уровнем определения основных объектов и отношений, формальными моделями решения коммуникационных задач. Рассмотрим три основных класса методов формирования коммуникационных предписаний: *декларативные методы формирования предписаний*, *процедуральные методы формирования предписаний* и *семантические методы формирования предписаний*.

2.3.1. Декларативные методы формирования предписаний

Этот класс уже был частично описан в п. 2.2.2. Здесь система формирования предписаний представляется полным описанием состояния и множеством преобразований, или операторов. Полное описание состояния представляется в виде множества предписаний безотносительно к тому, как их использовать. К классу декларативных методов формирования предписаний, кроме систем продукций и редукций, введенных в п. 2.2.2, относятся системы доказательства предписаний в исчислении предикатов. В этом методе решение задачи формирования предписаний сводится к доказательству того, что целевая формула предписания логически следует из системы начальных аксиом. Поскольку фактический процесс доказательства предписаний включает в себя применение некоторых правил вывода к начальным аксиомам, затем этих же или других правил вывода к аксиомам и выведенным на первом шаге предписания и т. д., вплоть до получения **целевой формулы предписания**, в системе доказательства предписаний легко обнаружить общие свойства декларативных методов формирования предписаний. Система формирования предписаний представляется полным описанием состояния, которое интерпретируется множеством аксиом и всех сформированных выведенных (сформированных) к данному моменту предписаний, и **множеством операторов, т. е. правил вывода предписаний.**

Основными и тесно связанными друг с другом характеристиками декларативных методов формирования предписаний являются:

— четкое разделение организации поиска *(механизм генерации),* представляющего собой полный перебор, и сокращения количества альтернатив, т. е. придания поиску узкой направленности *(механизм управления);*

— *универсальный,* т. е. проблемно-независимый *характер механизма генерации;*

— необходимость работы с полными описаниями состояний на каждой стадии процесса РКЗ.

Указанные характеристики будут описаны при рассмотрении конкретных декларативных методов формирования предписаний (п.п. 2.4—2.7) и методов решения коммуникационных задач с использованием этих предписаний (раз. 3, 4). Сейчас мы заметим, что эти характеристики предопределяют ряд особенностей декларативных методов формирования предписаний.

1. В силу универсальности механизма генерации декларативные предписаний могут в принципе служить основой для создания универсального РКЗ.

2. Эвристическая эффективность РКЗ в декларативных предписаниях полностью определяется свойствами механизма управления. Механизм управления поиском может быть задан в виде ограничений на допустимые состояния, в виде взаимосвязей между образами подмножеств состояний и операторами или соответствующим образом определенных эвристических функций. Чем точнее определен для каждой конкретной задачи механизм управления, т. е. чем выше знания системы о конкретной проблемной среде формирования предписаний, тем эффективнее осуществляется процесс РКЗ. Однако, в отличие от упомянутого выше знания, представленного описаниями состояний и операторов — структурного, или *синтаксического знания,*— знание о проблемной среде формирования предписаний отражает *понимание* системой глобальных свойств пространства поиска решения, или, что то же самое, законов, действующих в мире (области), в котором работает система формирования предписаний. Это *знание* мы называем *семантическим.* Итак,

а) эвристическая эффективность РКЗ формирования предписаний определяется количеством семантического знания, накопленного системой формирования предписаний;

б) структурное и семантическое знание в декларативных предписаниях полностью отделены друг от друга.

3. В силу характера указанных выше способов задания механизма управления декларативные методы формирования предписаний должны работать хорошо в легко **метризуемых пространствах** поиска решений формирования предписаний, т. е. в таких мирах (областях),

где семантическое знание может быть естественно выражено в **численном или логическом виде** (линейные неравенства, аддитивные функции, булевские матрицы и т. д.).

4. Попытки непосредственного введения семантического знания в описания состояний и операторов, т. е. в структурное знание, приводят к потере универсальности механизма генерации, что находится в полном соответствии с законом обратной зависимости общности и мощности.

2.3.2. Процедуральные методы формирования предписаний

В то время как основой описания объектов и отношений между ними в декларативных методах формирования предписаний является множество логических предписаний и общих правил вывода, процедуральные методы формирования предписаний основаны на описании в виде директив всех возможных манипуляций с объектами и отношениями. Таким образом, текущее знание системы формирования предписаний представляется в виде специально организованной базы данных и набора более или менее специализированных директив, обрабатывающих соответствующие области базы данных. Формирование в этом классе предписаний сводится к построению целенаправленной последовательности директив формирования предписаний, как правило, имеющей сложную рекурсивную организацию. Важно отметить, что в процедуральном методе формирования предписаний, в отличие от декларативного, на каждой стадии процесса РКЗ обрабатываются *локальные области базы данных*, причем необходимая для РКЗ информация представлена в императивной форме. Носителями процедуральных предписаний являются специальные *проблемно-ориентированные языки* (ПОЯ).

Процедуральные методы формирования предписаний наиболее естественно и эффективно реализуются в таких языках, которые содержат необходимые встроенные механизмы, обеспечивающие автоматический поиск решения на основе знания и поставленной цели. Помимо обычных выразительных средств, используемых алгоритмическими языками (структуры данных, управляющие структуры, операторы, процедуры), ПОЯ должны обладать встроенными *целенаправленными механизмами,* которые позволили бы им эпистемологически и эвристически адекватно представлять широкий класс систем формирования предписаний и коммуникационных задач, решаемых ими. В сущности, ПОЯ для задач формирования предписаний должен сочетать в себе свойства

современного алгоритмического языка высокого уровня и возможности решателя коммуникационных задач, являющегося программой, написанной в ПОЯ.

Отсюда вытекает двойственность функций ПОЯ. С одной стороны, он, как и обычный алгоритмический язык, должен упростить работу по программированию путем рационального выбора библиотеки часто встречающихся подпрограмм и макроопераций. С другой стороны, ПОЯ должен обеспечить такое наложение своей структуры и выразительных средств на решатель коммуникационных задач, запрограммированный пользователем, которое обеспечило бы построение эффективных стратегий решения, т. е. «вынуждало» бы пользователя применять именно те предписания, которые приводили бы к предпочтительным с точки зрения эффективности стратегиям.

Естественно, что при создании и использовании ПОЯ следует считаться с противоречием между его универсальностью и эффективностью при решении коммуникационной задачи (это уже известная нам зависимость между общностью и мощностью), а также с противоречием между мощностью встроенных механизмов (т. е. объемом спецификаций, задаваемых пользователем) и управляемостью в ходе решения со стороны пользователя (возможностью определить в каждый момент времени состояние процесса решения).

Перечислим теперь основные характеристики процедуральных методов формирования предписаний:

— наличие большого количества специализированных стратегий и правил, основанных на ряде унифицированных средств и механизмов, встроенных в ПОЯ;

— введение семантической информации в выражения предписаний, хранящиеся в базе данных (в виде свойств этих предписаний);

— использование на каждой стадии процесса РКЗ только тех предписаний из базы данных, которые необходимы активированной в данный момент процедуре и описаны в ней;

— возможность представления мира (области) на более высоком уровне описания, что в конечном счете сокращает длину решающей последовательности процедур формирования предписаний.

Указанные характеристики обеспечивают ряд преимуществ процедуральных методов формирования предписаний перед декларативными.

1. Введение семантической информации в описание элементов базы данных позволяет легко формализовать некоторые отношения, трудно выразимые в формальных декларативных методах формирования предписаний (например, равенства, отношения высшего порядка и т. д.).

2. Семантическое знание может быть выражено не только в числовом или логическом виде, но и в виде произвольных символических предписаний.
3. Отсутствие трудностей, связанных с обработкой полных описаний предписаний (п.п. 2.7, 2.8.7).

Указанные преимущества достигаются ценой потери общности по сравнению с декларативными методами формирования предписаний. Кроме того, *механизм недетерминистичного выбора* (п. 2.8), необходимый в ПОЯ для автоматического поиска решения, в совокупности с весьма слабо развитыми методами накопления семантического знания в процедуральных методах формирования предписаний обуславливает значительную неэффективность ПОЯ при решении задач в сложных обстоятельствах. Решение этой проблемы обычно находят в более тесном общении системы формирования предписаний с человеком, на долю которого выпадает обязанность ввода ограничивающего перебор семантического знания (например, путем рекомендаций).

2.3.3. Семантические методы формирования предписаний

Как уже указывалось выше, одним из недостатков как процедурального, так и в большей степени декларативного методов формирования предписаний является разделение структурного и семантического знания, что затрудняет их использование, особенно в реальных областях. Именно в таких областях семантика, являющаяся инструментом повышения эффективности решения задач формирования предписаний, играет особо важную роль. Поэтому естественно потребовать, чтобы семантика непосредственно отражалась в самом формализме формирования предписаний. Другими словами, нам нужен такой уровень сформированного предписания, элементами которого явились бы директивы и семантические отношения между ними. Теоретико-графическое представление такого уровня называется *семантической сетью.* Существует довольно много вариантов реализации семантических сетей, что позволяет говорить о целом *классе семантических методов формирования предписаний*. Общие характеристики этого класса сводятся к следующему:

— описание предметов коммунипации и/или директив выводится на уровень естественного языка;

— все знания, включая вновь поступающие факты, а также некоторые специализированные методы решения формирования

предписаний, накапливаются в относительно однородной структуре памяти;

— на сетях определяется ряд более или менее унифицированных семантических отношений между предметами коммунипации и/или директивами, которым соответствуют унифицированные методы вывода;

— методы вывода предписаний и/или директив в совокупности с целями (запросами) определяют участки семантического знания, имеющего отношение к поставленной задаче, формулируя акт понимания запроса и некоторую цепь выводов и неполных выводов, соответствующих решению задачи.

Следует отметить следующие особенности семантических методов формирования предписаний и/или директив:

1. В семантической сети формирования предписаний и/или директив могут быть представлены такие виды объектов, как понятия, события, специализированные методы решения; следует, однако, учесть, что увеличение номенклатуры объектов снижает однородность сети и приводит к необходимости увеличения арсенала методов вывода.
2. Многомерность семантических сетей позволяет представлять в них многочисленные семантические отношения, связывающие отдельные понятия, понятия и события в предписаниях и/или директивах, а также предложения в текстах; кроме того, в семантической сети может быть отражена семантическая иерархия специализированных методов решения, определяющая их взаимоподчиненность.
3. Формализация, или структурное представление семантических знаний, позволяет наложить на эти знания некоторую *суперсемантику,* отражающую относительную «силу» семантических отношений, что способствует повышению эффективности вывода в семантических сетях.
4. На каждой стадии РКЗ можно четко разделить *полное знание* системы формирования предписаний и/или директив (полная семантичесткая сеть) и *текущее знание* — возбужденный участок семантической сети, в котором производятся некоторые операции (процесс понимания, вывода, процесс коммуникации и т. д.).

Необходимо учесть, что, обладая преимуществом структурирования общего семантического знания, семантические методы формирования предписаний часто проигрывают в представлении чисто структурных отношений, легко реализуемых в исчислении предикатов (логические связки, кванторы общности и существования) или процедуральном методе формирования предписаний (параллельные процессы, гипотетические области, динамические события).

Поэтому ряд исследований в области формирования предписаний должен осуществляться путем вложения в семантические методы формирования предписаний некоторых фрагментов процедуральных и декларативных методов формирования предписаний с целью объединения их преимуществ в новом, смешанном методе формирования предписаний.

Последующие параграфы настоящего раздела посвящены описанию методов формирования предписаний и решения задач формирования предписаний. Поскольку процедуральные и семантические методы формирования предписаний допускают лишь достаточно специфические, проблемно-зависимые методы решения, мы излагаем лишь некоторые общие механизмы и алгоритмы. В то же время, учитывая универсальность механизмов генерации и независимость механизмов управления в декларативных методах формирования предписаний, мы сочли необходимым выделить этот материал в отдельные разделы (раз. 3 — для эвристических методов формирования предписаний, раз. 3 — для доказательства предписаний и/или директив в исчислении предикатов).

2.4. Эвристические методы формирования предписаний на основе декларативных методов формирования предписаний

2.4.1. Общая постановка задачи

Одна из многочисленных, но близких друг к другу постановок задачи эвристического метода формирования коммукационных предписаний заключается в следующем: заданы *начальная коммукационная ситуация* (объект, состояние, директива), конечная, или *целевая коммукационная ситуация* (объект, состояние, директива); задано также *множество коммукационных операторов (директив)*, преобразующих одну коммукационную ситуацию (предписание) в другую. Требуется найти такую последовательность коммукационных операторов (директив), которая преобразует начальную коммукационную ситуацию (предписание) в конечную ситуацию коммукационную (предписание).

На эту базовую постановку часто булем накладывать ограничения. Так, иногда требуется найти последовательность операторов (директив), оптимальную в некотором определенном смысле. Часто рассматривается обобщенная постановка, где задается множество начальных и (или) конечных состояний.

Формально задача, поставленная как задача формирования коммукационных предписаний эвристическим методом, представляет из себя четверку *(S_o, S, F, T),* где S — множество состояний, $S_0 \subseteq S$ — множество начальных состояний, $T \subseteq S$ — множество конечных состояний, F — множество операторов. Каждый оператор $f \in F$ является функцией, отображающей S_f в S, где $S_f \subseteq S$ — область определения f. Если $s \in S_f$, то *f применим* к s.

Решением задачи является последовательность операторов $f_1, f_2, ..., f_n$ такая, что $f_i \in F$, $i=1, 2,..., n$, $f_1° f_2°. . .° f_n (s) \in T$, где $f_1° f_2°. . .° f_n (s)$ обозначает *композицию функций*

$$f_n (...(f_2 (f_1 (s))...) , \quad s \in S_0, \qquad (2.14)$$

причем эта композиция определена, если

$$s \in S_{f_1}, f_1(s) \in S_{f_2}, ..., f_n(...(f_2(f_1(s)))...) \in S_{f_{n-1}}.$$

Метод решения задачи (S_o, S, F, T) будем называть *эвристическим методом формирования коммукационных предписаний*, если он на каждом шаге находит все возможные применения операторов к данному текущему состоянию, а порядок рассмотрения состояний и порядок применения операторов управляется свойствами уже рассмотренных до этого шага состояний

Будем использовать два основных подхода к решению определенной выше задачи формирования коммукационных предписаний эвристическим методом — основанный на *продукционных системах формирования коммукационных предписаний* (т.е. использующих методы формирования коммукационных предписаний в *пространстве состояний)* и основанный на *редукционных системах формирования коммукационных предписаний* (т. е. использующих сведение решения задачи к решению ее подзадач). Оба эти подхода достаточно наглядно продемонстрированы в п. 2.2.2 на примере решения задачи о миссионерах и людоедах. Поэтому в настоящем параграфе мы лишь дадим общие постановки решения задач в этих системах и подчеркнем связь между продукционными и редукционными системами решения задач.

2.4.2. Формирование коммукационных предписаний в пространстве состояний *(система продукций).*

В системе продукций будем представлять пространство поиска решений формирования коммукационных предписаний в виде *локально-конечного направленного графа G=(X,* Г), где

$X=\{x_0, x_1 \ldots\}$ — множество, в общем случае бесконечное, вершин графа, каждая из которых отождествляется с одним из состояний $s \in S$;

$E=\{(x_i, x_j)/x_i, \quad x_j \in X, \quad x_j \in \Gamma(x_i)\}$ — множество дуг, или ребер графа, бесконечное, если бесконечно множество X;

$\Gamma: X \to 2^X$ — конечное отображение, т. е. для всех $x \in X |\Gamma(x)| \in N$; N — целое число;

$|\Gamma(x)|$ — количество *дочерних вершин x,* т. е. вершин, соединенных с x дугой.

В множестве вершин X мы выделяем подмножества вершин $X_0 \subseteq X$, соответствующее множеству начальных состояний $S_0 \subseteq S$, и $X_t \subseteq X$, соответствующее множеству конечных состояний $T \subseteq S$.

Определим *путь в графе G* как $\mu=(x_1, x_2, \ldots, x_k), (x_i, x_{i+1}) \in E, i=1, 2, \ldots, k-1$. Если $x_1 \in X_0$, $x_k \in X_t$, то очевидно, что решение задачи эвристического поиска в пространстве состояний (т. е. нахождение последовательности операторов, преобразующей начальное состояние в конечное) сводится к *задаче поиска пути* μ *на графе G. Путь* из $x_0 \in X_0$ в $x_t \in X_t$ называется *решающим.*

Для локально-конечного графа G целесообразно *неявное задание,* т. е. определение множества X_0 и множества операторов, которые, будучи применимы к вершине графа, дают все ее дочерние вершины (естественно, что выполнение условия применимости $\Gamma_i \subseteq \Gamma$ к вершине x_i обязательно, хотя возможно, что для некоторой x_i $\Gamma_i = \varnothing$). При таком задании механизм генерации решений должен строить в явной форме некоторый *подграф неявно заданного графа G, содержащий по крайней мере одну конечную вершину.*

Представление пространства поиска решений в виде графа G обеспечивает решение, начиная от начального состояния. Однако в тех случаях, когда целевое состояние явно задано, может оказаться целесообразным проводить поиск пути в графе, начиная от конечной вершины к начальной. Более того, в этом случае можно скомбинировать эти два поиска в единый *двунаправленный поиск* в графе. Интуитивно очевидно, что поскольку поисковые деревья растут экспоненциально, то два поиска с меньшей глубиной могут оказаться эффективнее одного поиска с суммарной глубиной. Алгоритм двунаправленного поиска будет рассмотрен в раз. 3. Следует отметить, что дополнительно к рассмотренному выше представлению в системе продукций необходимо добавить отображение предшествования $\Gamma^{-1}: X \to 2^X$, где

$$\Gamma^{-1}(x_j) = \{x_i / x_i, x_j \in X, x_j \in \Gamma(x_i)\}. \qquad (2.15)$$

2.4.3. Решение коммуникационных задач в системе редукций. Пропозициональные графы

Если решение задач в системе продукций сводится к поиску решающего пути, то основной идеей редукционной системы формирования коммуникационных предписаний является *поиск доказательства (обоснования) того, что решение задачи формирования коммуникационного предписания выводится из решения совокупности ее подзадач.*

Другими словами, решение задачи в этой системе сводится к нахождению множества альтернативных совокупностей подзадач, каждая из которых дает решение задачи, затем множества альтернативных совокупностей подзадач этих подзадач и т. д. до тех пор, пока задача формирования коммуникационного предписания не станет *разрешимой,* т. е. решение всех ее подзадач не станет очевидным, или пока не будет доказано (обосновано), что задача *не имеет решения.* Очевидность решения подзадач определяется следующими возможностями:

1) Подзадача формирования коммуникационного предписания носит характер общеизвестного утверждения (аксиома).
2) Подзадача формирования коммуникационного предписания легко может быть решена в системе продукций формирования коммуникационных предписаний (например, за один шаг).
3) Подзадача формирования коммуникационного предписания хотя и сложна, но ее решение известно системе формирования коммуникационных предписаний на основе предыдущего опыта формирования коммуникационных предписаний.

Подход с использованием редукционной системы формирования коммуникационных предписаний является в некотором роде обобщением подхода с использованием пространства состояний. Действительно, в процессе сведения задачи к совокупности подзадач могут возникнуть различные возможности такого сведения (альтернативные совокупности); в то же время применение оператора в продукционной системе формирования коммуникационных предписаний сводит задачу к более простой подзадаче, но эта возможность для данного оператора единственна. С другой стороны, редукцию можно рассматривать и как вспомогательный процесс разбиения задачи поиска решающего пути в пространстве состояний на подзадачи поиска подпутей этого пути с последующей их композицией в окончательном решении задачи.

Как продукционный, так и редукционный подход формирования коммуникационных предписаний требуют для решения задачи

использования процессов поиска с той только разницей, что в первом случае поиск осуществляется в *пространстве состояний,* а во втором — в *пространстве описаний множеств подзадач.*

Пространство описаний множеств подзадач представляется в виде специального направленного графа *G,* называемого *«И/ИЛИ-графом,* или *пропозициональным графом.*

С каждой вершиной этого графа связывается описание определенной подзадачи. Дуги этого графа соответствуют операторам сведения задачи к подзадачам. В графе выделяются два типа вершин: *конъюнктивные вершины,* или вершины типа «И», которые вместе со своими дочерними вершинами интерпретируются высказыванием «чтобы решить задачу, *необходимо* решить все ее подзадачи», и *дизъюнктивные вершины,* или вершины типа «ИЛИ», которые вместе со своими дочерними вершинами интерпретируются высказыванием «чтобы решить задачу, *достаточно* решить одну из ее подзадач». Дуги, исходящие из конъюнктивной вершины, связаны дужкой при этой вершине. Пример пропозиционального графа при веден на рис. 2.4.

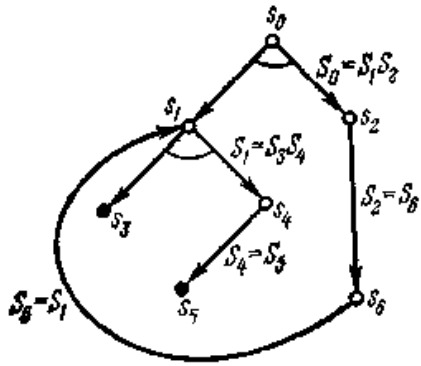


Рис. 2.4. Пример пропозионального графа. Конечные вершины представлены зачерненными точками.

Здесь s_0 — первоначальная задача, для решения которой *обходимо* решить подзадачи s_1 и s_2, для решения s_1 *необходимо* решить подзадачи s_3 и s_4, для решения s_2 *достаточно* решить s_6, для решения s_4 и s_6 *достаточно* решить s_5 и s_1 соответственно. Решение задач s_3 и s_6 предполагается известным.

В множестве вершин пропозиционального графа выделяются подмножество *начальных вершин,* т. е. задач, которые следует решить, и подмножество *конечных вершин,* т. е. заведомо разрешимых задач. Это завершает формулировку задачи как задачи эвристического метода

формирования коммуникационных предписаний, причем решение ее сводится к нахождению в пропозициональном графе *решающего графа,* формальное определение которого будет дано несколько ниже.

С каждой вершиной пропозиционального графа мы связываем высказывание в виде *булевской функции,* выраженной в дизъюнктивной нормальной форме и образующейся по следующим правилам: для вершины s, имеющей дочерние вершины $s_1, s_2, \ldots, s_k$,

а) если s — конъюнктивная вершина, то соотнесенная с ней булевская функция

$$S = \bigwedge_{i=1}^{k} S_i,$$

где S_i — булевская функция, соотнесенная вершине s_i;

б) если s — дизъюнктивная вершина, то соотнесенная с ней булевская функция

$$S = \bigvee_{i=1}^{k} S_i;$$

в) если s — конечная вершина, то соотнесенная с ней булевская функция *тождественно истинна* (из нашего определения конечной вершины пропозиционального графа следует, что вершины, не являющиеся конечными и не имеющие дочерних вершин, соответствуют заведомо неразрешимым задачам; для этих вершин булевская функция *тождественно ложна);*

г) если $s_1, s_2, \ldots, s_m$ — начальные вершины, то с ними соотносится булевская функция

$$S = \bigwedge_{i=1}^{m} S_i.$$

На рис. 2.4 рядом с вершинами показаны соотнесенные с ними булевские функции.

Введем ряд формальных определений.

Пусть s — дизъюнктивная вершина, т. е. $S = \bigvee_{i=1}^{k} S_i$, s_i — дочерние вершины s. Тогда каждая из S_i, $i = 1, 2, \ldots, k$, называется *непосредственной импликантой S.*

Пусть s — конъюнктивная вершина, т. е. $S = \bigwedge_{i=1}^{k} S_i$, s_i — дочерние вершины s. Тогда *непосредственной импликантой* S называется булевская функция, получаемая из S заменой S_i, $i=1,2, \ldots, k$, одной из ее непосредственных импликант.

Пусть s_1, s_n — конъюнктивные вершины. Тогда S_n является *импликантой* S_1, если имеется последовательность конъюнктивных

вершин s_1, s_2.....s_n такая, что S_i является непосредственной импликантой S_{i-1}, i=2, 3, . . ., n.

Пусть s — конъюнктивная вершина с дочерними вершинами s_1, s_2, . . ., s_k. Назовем *путевым графом, начинающимся в вершинах* s_1, s_2, . , ., s_k *и заканчивающимся в вершинах* t_1, t_2, . . ., t_m, где s_i, $t_l \in G$, i=l, 2, . . ., k; l=1, 2, ... m, такой конечный подграф G' пропозиционального графа G, что

а) s_i, $t_l \in G'$, i=1,2, . . .,k; l=1,2, . . ., m.

б) Только s_i, i=l, 2, . . ., k, не имеют входящих дуг, а t_l, l=1, 2, . . ., m. — исходящих.

в) Для всех вершин $x \in G'$, кроме t_l, l=1, 2, . . ., m, имеются такие дочерние вершины $x_j \in G'$, j=1, 2, . . ., p, что X_1, X_2, . . ., X_p являются единственными непосредственными импликангами X в G'.

г) $T = \bigwedge_{l=1}^{m} T_l$ — импликанта S в G'.

Если t_l, l=1, 2, . . ., m,— конечные вершины графа G, то путевой граф называется *решающий графом, начинающимся в вершинах* s_i, i=1, 2, . . .,k. Решающий граф, начинающийся в *начальных* вершинах G, называется *решающим графом.* Граф, представленный на рис. 2.4, является решающим, поскольку, как видно, S_3S_5 является импликантой S_0.

Итак, решение задачи в системе редукций может быть сведено к поиску *решающего графа исходного пропозиционального графа,* поскольку, как вытекает из предыдущего изложения, просмотр решающего графа от конечных вершин к начальным точно задает множество подзадач, которые необходимо решить для решения исходной задачи, и порядок их решения. Необходимость поиска решающего графа определяется наличием более чем одной дочерней вершины у дизъюнктивных вершин, или, другими словами, наличием альтернативных совокупностей подзадач, решение которых необходимо для решения исходной задачи.

Поскольку во всех предыдущих рассуждениях не накладывалось никаких ограничений на конечность пропозиционального графа, необходимо его *неявное задание,* т. е. задание множества начальных вершин и оператора Г, генерирующего для данной вершины дочерние вершины и указывающего для нее булевскую функцию в виде конъюнкции дочерних вершин (мы предполагаем, чго оператор, сводящий решение задачи к решению ее подзадач, применяется к конъюнктивным вершинам, в то время как процесс выбора оператора определяет альтернативные совокупности подзадач, т. е. образует вершины, дочерние для дизъюнктивной вершины).

Следует заметить, что любой конечный пропозициональный граф с разделимыми конъюнктивными и дизъюнктивными вершинами может быть отображен в *контекстно-свободную грамматику*. При этом конъюнктивные вершины соответствуют *продукциям,* дизъюнктивные вершины — *вспомогательным символам,* конечные вершины — *основным символам,* дуги из конъюнктивных вершин к дизъюнктивным вершинам определяют *выбор подстановки для переменных,* а дуги из дизъюнктивных вершин к конъюнктивным показывают *действительную подстановку*. На рис. 2.5 показан пропозициональный граф, соответствующий контекстно-свободной грамматике

$$G = (\{a, b\}, \{S, A\}, S, P),$$
$$P = \{p_1\colon S \to aAS,\ p_2\colon S \to a,\ p_3\colon A \to$$
$$\to Sba,\ p_4\colon A \to ba,\ p_5\colon A \to SS\}, \quad (2.16)$$

где A — вспомогательный символ, S — начальный символ, a, b — основные символы, P — множество продукций.

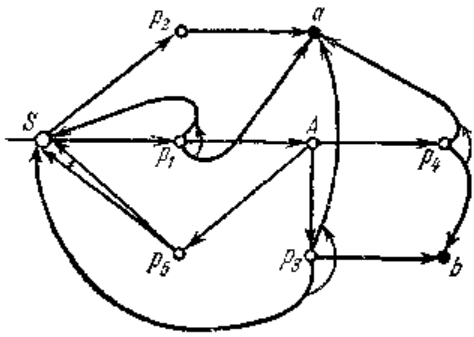


Рис. 2.5. Представление контекстно-свободной грамматики в виде пропозиционального графа.

Такое представление пропозиционального графа предопределяет упорядочение, накладываемое на порядок генерации вершин, следующих за конъюнктивными (на рис. 2.5 указано дужками со стрелками).

Таким образом, к поиску решающего графа в пропозициональном графе могут быть приложены методы теории формальных грамматик. В частности, решающий граф в пропозициональном графе соответствует *дереву вывода* некоторого высказывания в соответствующей контекстно-свободной грамматике, а поиск такого

графа эквивалентен поиску высказывания в языке, соответствующем контекстно-свободной грамматике, вместе с его деревом вывода.

2.4.4. Механизмы сведения задач к подзадачам

На первый взгляд формирование коммуникационных предписаний в виде пропозиционального графа кажется многообещающей основой для построения универсального решателя задач. Однако будем помнить, что использование теоретико-графической модели позволяет формализацию лишь одного из элементов декларативного метода формирования коммуникационных предписаний, а именно пространства описания множеств подзадач формирования коммуникационных предписаний. Что же касается коммуникационных предписаний, определенных на множестве вершин графа, то как в настоящем параграфе, так и в следующем разделе, где будут рассмотрены соответствующие алгоритмы, мы лишь считаем, что задано некоторое, как правило, фиксированное множество операторов, позволяющее на каждом шаге порождать все вершины, дочерние по отношению к рассматриваемой. Иначе говоря, теоретико-графическая модель формирования коммуникационных предписаний сама по себе не дает сколько-нибудь систематического подхода к решению в общем виде следующей важной задачи: *«каким образом осуществить акт разбиения задачи на подзадачи?»* Она лишь отвечает на вопрос, *как* решить задачу, если такой подход *существует,* т. е. представляет собой скелет, на который можно наложить решаемую задачу с соответствующей ей специализированной семантикой описания вершин и дуг графа, т. е. конкретной интерпретацией описания задачи и ее подзадач, а также допустимых операторов разбиения задач на подзадачи. В этой связи представляют интерес независимые или хотя бы частично независимые от задачи механизмы сведения задач к подзадачам *(механизм редукции)*. Шагом на пути к построению такого рода механизмов применительно к представлению в виде пропозиционального графа является использование понятий ключевых состояний и ключевых операторов.

Рассмотрим метод сведения задачи к совокупности подзадач, последовательно упрощающий задачи формирования коммуникационных предписаний в пространстве состояний, т. е. накладывающий механизм редукции на решение задачи в системе продукций.

Представим задачу формирования коммуникационного предписания в пространстве состояний в виде (S_0, F, T), S_0 — множество начальных состояний, T —множество целевых состояний, F — множество

операторов, отображающих состояния в состояния. Пусть также заданы множества *ключевых состояний* $T_1, T_2, \ldots, T_N$, т. е. множества тех состояний, через которые, наиболее вероятно, пройдет решающий путь в графе. Тогда можно использовать механизм редукции для сведения задачи (S_0, F, T) к совокупности задач (S_0, F, T_1), $(\{t_1\}, F, T_2)$, ..., $(\{t_N\}, F, T)$, эквивалентной исходной задаче. Здесь $t_1 \in T_1, t_2 \in T_2, \ldots, t_N \in T_N$ — конкретно выбранные ключевые состояния.

Одним из приемов нахождения множеств ключевых состояний является выделение *ключевых операторов,* т. е. операторов, применение которых необходимо для решения задачи (таков, например, оператор ППЛ, 1,1 на рис. 2.1 в задаче о миссионерах и людоедах). Пусть $f \in F$ — ключевой оператор для задачи (S_0, F, T). Тогда задача может быть разбита на три подзадачи:

1) Поиск пути к состоянию $t \in T_f$, T_f — область определения f, т.е. множество состояний, к которым f применим,— подзадача (S_0, F, T_f).
2) Применение оператора f — подзадача $(\{t\}, F, \{f(t)\})$.
3) Оставшаяся часть задачи — подзадача $(\{f(t)\}, F, T)$.

Заметим, что если бы нам было задано множество операторов $F_k \subset F$ такое, что $f \in F_h$, то это привело бы к необходимости построения пропозиционального графа для получения альтернативных совокупностей трех подзадач указанного выше вида.

Недостаток описанного метода заключается в том, что, за исключением тривиальных случаев, ключевые состояния или операторы могут быть найдены на основе анализа пространства состояний, а это, как указывалось в п. 2.2, едва ли не самая сложная проблема в области автоматизированного формирования коммуникационных предписаний.

В эвристическом методе формирования коммукационных предписаний основными понятиями являются состояния и операторы. Поэтому формальная и содержательная постановка задачи в эвристическом методе полностью совпадает с изложенными в начале этого параграфа.

В процессе выполнения эвристического метода находят *различия* между текущим и целевым состояниями. На основе этих различий выбирается оператор, который применяется к текущему состоянию, вырабатывая новое состояние. Далее производится сравнение этого состояния с целевым, и цикл повторяется. В случае неприменимости выбранного оператора к текущему состоянию определяются различия, суммирующие причину неприменимости. На основе этих различий выбирается оператор, *пригодный* для их устранения. Если он применим и устраняет их, то применяется предыдущий оператор. Однако он может быть *неприменим* или *непригоден,* поэтому изложенная схема работы метода рекурсивна.

Основной механизм редукции использует три стандартных метода (рис. 2.6): *преобразование состояния A* в состояние *B, уменьшение различия D* между состояниями *A* и *B* и *применение оператора f* к состоянию *A*.

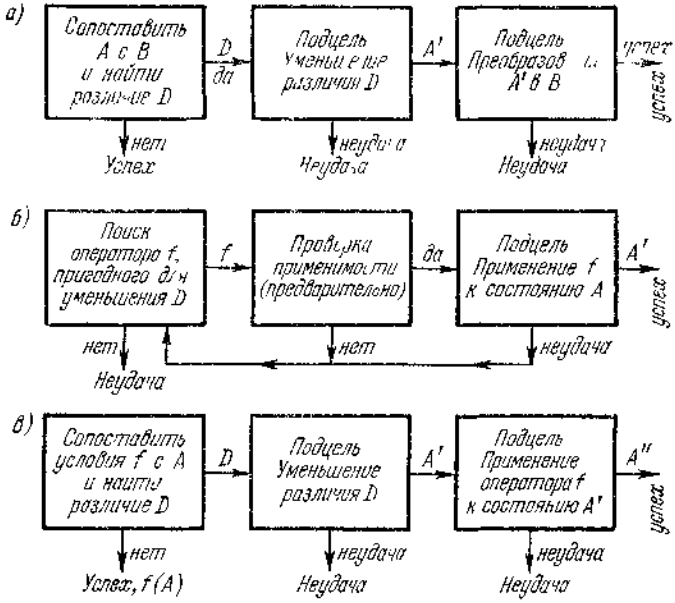


Рис. 2.6. Основные методы механизма редукции: *а)* метод преобразования состояния *A* в состояние *B, б)* метод уменьшения различия *D* между состояниями *A* и *B, в)* метод применения оператора *f* к состоянию *A*.

Преобразование состояния. Генерируется выведенная (т. е. полученная путем последовательного применения операторов к *A* и следующим состояниям) последовательность состояний, оканчивающаяся состоянием, идентичным *B*.

Уменьшение различия. Вырабатывается новое состояние *A'*, выведенное из *A* с измененным различием *D*.

Применение оператора. Генерируется новое состояние применением *f* к *A* или состоянию, выведенному из *A*.

Пример работы механизма редукции приведен на рис. 2.7, где изображено дерево методов для преобразования объекта *A* в объект *B* (пример необходимо прослеживать по рекурсивной схеме рис. 2.6).

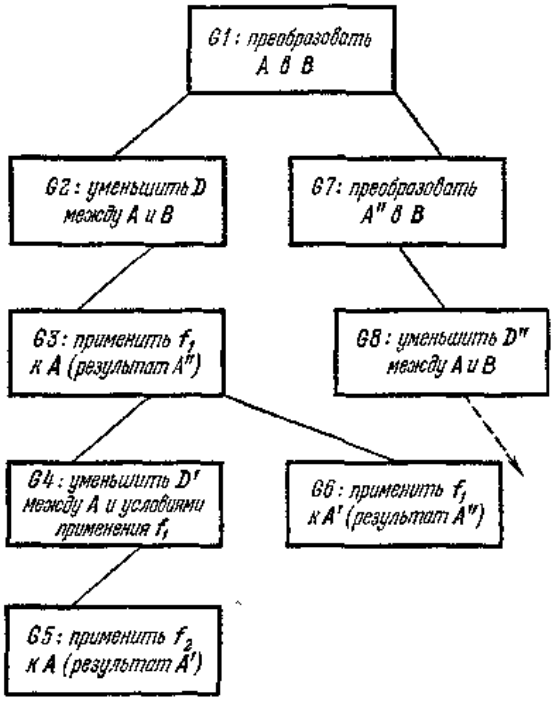


Рис. 2.7. Пример работы механизма редукции.

Пытаясь преобразовать *A* в *B,* механизм находит различие *D* между *A* и *B* и переходит к его уменьшению (G2), находит оператор f_1, пригодный для уменьшения, и пытается применить его к *A* (G3). Однако оператор f_1 неприменим, и механизм находит различие *D'* и пытается его уменьшить (G4). Предположим, что оператор f_2 пригоден для уменьшения *D'* и применим к *A* (G5). Тогда вырабатывается новое состояние *A'*. Теперь механизм записывает *A*' как результат G5 и G4 и переходит к применению *f*i к *A*'. Поскольку различие *D*' устранено, f_1 применяется к *A*', вырабатывается результат *A*" (G6). Этот результат записывается в G3 и G2. Поскольку различне *D* устранено, производится переход к преобразованию *A*" в *B* (G7). К этому моменту механизм выработал последовательность операторов $f_2 o f_1(A)=f_1(f_2(A))$, преобразующую *A* в *A",* и очередную подзадачу преобразования *A"* в *B.*

Представим процесс редукции (рис. 2.7) с помощью пропозиционального графа (рис. 2.8).

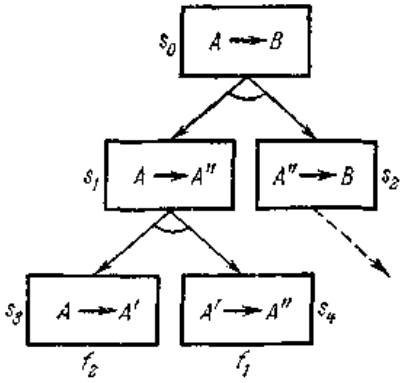


Рис. 2.8. Пропозициональный граф, соответствующий процессу редукции.

В вершинах графа записаны формулировки исходной задачи и ее подзадач. Граф содержит только конъюнктивные вершины, так как мы предполагали для простоты, что механизм метода обладает способностью выбирать один пригодный оператор. Вершины s_3 и s_4 являются конечными, так как им соотносятся операторы f_2 и f_1 соответственно, непосредственно преобразующие A в A' (f_2) и A' в A'' (f_1).

Из сопоставления двух формализмов — механизма редукции и представления с помощью пропозиционального графа — можно сделать следующие выводы:

1) Механизм редукции метода осуществляет разбиение задачи на подзадачи с помощью метода уменьшения различия, не требуя специального набора операторов для этой цели; однако он должен обладать эффективными методами определения различия между двумя состояниями и выбора оператора, пригодного для уменьшения или устранения этого различия. Решение первой задачи при заданном формализме описания коммуникационных предписаний не представляет принципиальных трудностей, чего нельзя сказать о второй задаче.

Что касается представления коммуникационных предписаний в виде пропозиционального графа, то без информации о конкретном содержании задачи и о свойствах пространства описаний множества подзадач мы не смогли бы определить множество операторов, преобразующих вершины графа в дочерние вершины, т. е. осуществить разбиение задачи, представленной на рис. 2.7, на подзадачи.

2) Как будет показано в следующем разделе, представление коммуникационных предписаний в виде пропознционального графа

допускает построение допустимых алгоритмов, т. е. всегда находящих решение задачи, если оно есть, а при определенных условиях — алгоритмов, находящих оптимальные решения. В то же время использование механизма редукции даже при наличии специализированной системы формирования коммуникационных предписаний, не всегда гарантирует нахождение решения.

Рассмотрим условия, при которых механизм редукции находит решение задачи. Рассмотрим произвольное отображение $S \times S \rightarrow D$, где D — множество различий. Это отображение ставит в соответствие каждой паре $(s_1, s_2), s_1, s_2 \in S$, различия $\{d_j\} \subseteq D$.

Введем линейное упорядочение $>$ на множестве D, так что $d_1 > d_2$, $d_1, d_2 \in D$, означает, что d_1 — *более трудное* для уменьшения *различие,* чем d_2. Это определение не допускает равнотрудных различий, и в этом случае необходимо объединять их в одно различие $d = d_1 \cup d_2$.

Для заданных множества операторов F и множества различий D построим функцию W: $D \times H \rightarrow \{0, 1\}$, где H — разбиение, заданное на F. Для $d \in D$, $h \in H$, $f \in h$ $W(d, h) = 1$ означает, что f пригоден для уменьшения различия d, $W(d, h) = 0$ означает, что f непригоден для уменьшения различия d.

Функция W, выраженная в табличной форме, носит название *таблицы связок* и показывает, какие из групп операторов являются пригодными для уменьшения тех или иных различий.

Построим специальный вид таблицы связок — *треугольную таблицу*. Каждому d_i присваивается $h_i \in H$ так, чю $W(d_i, h_i) = 1$, а $W(d_k, h_i) = 0$ для всех $d_k > d_i$ (индексы присваиваются так, что если $d_k > d_i$, то $k > i$).

Таким образом, мы строим треугольную таблицу, производя такие разбиения множества операторов, чтобы каждая группа операторов была пригодной для уменьшения определенного различия, но не уменьшала бы различия большей трудности. По отношению к различиям меньшей сложности она может быть как пригодной, так и непригодной.

Мы рассматриваем далее класс задач эвристического метода формирования коммуникационных предписаний — *А-задачи,*— который описывается пятеркой (S_0, S, F, T, W), W — треугольная таблица связок. В процессе разбиения задачи на подзадачи образуется два типа подзадач: непосредственно решаемые и подлежащие дальнейшему разбиению. Их результаты обозначим через $R_0(\sigma)$ и $R_1(\sigma)$ соответственно, σ — некоторое промежуточное состояние. Определим также *максимальное различие между* s_1 *и* s_2, $s_1, s_2 \in S$, с помощью функции M: $S \times S \rightarrow D$. $M(s_1, s_2) = d_i$ тогда и только тогда, когда $d_i > d_j$ для всех $d_j \in \{ d_j \}$. Заметим, что в случае $s_1 = s_2$ $M(s_1, s_2)$ не определена.

Максимальное различие между $s_1 \in S$ *и* $X \subseteq S$, $MM(s_1, X) = \min_{s_2 \in X} M(s_1, s_2)$, причем *MM* не определена, если $S_i \in X$.

Если $f \in h_i$, $MM(\sigma, T) = d_i$ и, следовательно, $W(d_i, h_i) = 1$, то $R_0(\sigma) = \{f(\sigma)/\sigma \in S_f\}$, $R_1(\sigma) = \{f(\tau)/\tau \in S_f$ и существует решение подзадачи $(\sigma, S, \bigcup_{k=1}^{i-1} h_k, \{\tau\}, W)\}$.

Δ*-схема* Δ-задачи (S_0, S, F, T, W) есть последовательность $(s_0, s_1, \ldots s_n)$, $s_0 \in S_0$, $s_n \in T$, такая, что $s_i \in R_0(s_{i-1}) \cup R_1(s_{i-1})$, $i = 1, 2, \ldots, n$. Теперь становится ясной важность введенной треугольной таблицы связок. Действительно, пусть σ — элемент Δ-схемы и $MM(\sigma, T) = d_i$. Тогда мы используем $f \in h_i$ для уменьшения d_i. Если $\sigma \in S_f$, т.е. f применим, то $f(\sigma) \in R_0(\sigma)$. В противном случае ставится подзадача преобразования σ в S_f. Однако для решения этой подзадачи используются только операторы из $\bigcup_{k=1}^{i-1} h_k$. Если результатом решения этой подзадачи является $\tau \in S_f$, то $f(\tau) \in R_1(\sigma)$.

Пусть $f_1 \circ f_2 \circ \ldots \circ f_n(s_0) = t$, $t \in T$. Решение Δ-задачи *упорядочено* тогда и только тогда, если $MM(s_0, T) = M(s_0, t) \geq MM(f_1(s_0), T) = M(f_1, (s_0), t)$ и $M(f_1 \circ f_2 \circ \ldots \circ f_i(s_0), t) \geq MM(f_1 \circ f_2 \circ \ldots \circ f_{i+1}(s_0), T) = M(f_1 \circ f_2 \circ \ldots \circ f_{i+1}(s_0), t)$, $i = 1, 2, \ldots, n-2$.

Таким образом, вырабатывая упорядоченное решение, механизм редукции последовательно уменьшает монотонно невозрастающую последовательность различий. На каждом шаге из множества конечных объектов рассматривается тот, различие которого с текущим объектом является минимальным, т. е. $M(\sigma, t) = MM(q, T)$ для любых $\sigma \in S$, $t \in T$.

Можно показать, что *наличие упорядоченного решения* Δ*-задачи и всех ее подзадач дает достаточные условия того, что механизм редукции найдет решение задачи, если оно есть*.

Преимущества этого подхода заключаются в том, что

1) На каждом шаге решения задачи механизм рассматривает лишь подмножества множества операторов, причем в ходе решения задачи эти подмножества последовательно сокращаются.

2) Механизм рассматривает лишь те подзадачи, которые легче (опять в смысле используемого подмножества операторов), чем образующая их задача.

Однако пользователь системы формирования коммуникационных предписаний должен задать ей множество различий, их упорядочение и таблицу связок.

2.5. Методы доказательства (обоснования) коммуникационных предписаний на основе декларативных методов

В данном параграфе мы изложим еще один вид декларативного метода формирования коммуникационных предписаний, используемого как для представления существующих коммуникационных предписаний, так и для сведения процесса решения задачи формирования коммуникационных предписаний или ее части к автоматическому логическому анализу. Постановка задачи при указанном подходе заключается в следующем. Задача записывается в виде директив некоторого формального языка. При этом часть директив, соответствующая исходным данным, рассматривается как аксиомы, а цель задачи рассматривается как коммуникационное предписание, справедливость которого следует установить или опровергнуть на основании аксиом и правил вывода формальной системы формирования коммуникационных предписаний.

Существуют различные логические формализмы, пригодные для записи в них коммуникационных предписаний, относящихся к широкому кругу коммуникационных задач.

Мы будем далее рассматривать только исчисления предикатов первого порядка с равенством и без равенства, так как для этих исчислений разработаны универсальные и эффективные процедуры, обладающие *полнотой,* т. е. всегда устанавливающие наличие некоторого факта, если он выводим из аксиом (более подротно об использовании исчислений предикатов для формирования коммуникационных предписаний см. в следующих разделах) .

Любая логическая система (теория) формирования коммуникационных предписаний и исчисление предикатов, в частности, может быть построена на базе как синтаксических, так и семантических концепций. Теорию формирования коммуникационных предписаний, построенную на базе семантических концепций, будем называть *теорией моделей коммуникационных предписаний*, а теорию, построенную на базе синтаксических концепций,— *аксиоматической теорией коммуникационных предписаний*. При обоих способах построения некоторой теории А необходимо определить понятие алфавита и формулы предписания (директивы).

Алфавитом называется некоторое счетное множество символов теории. Произвольные конечные последовательности символов алфавита называются *выражениями директив* теории А.

Формулой теории А будем называть некоторое выделенное подмножество директив теории А. Алфавит исчисления предикатов состоит из следующего множества символов:

1. Знаков пунктуации ( ), .
2. Пропозициональных связок ~, $\vee, \wedge, \rightarrow$.
3. Знаков кванторов $\forall, \exists$.
4. Символов переменных x_k, $k = 1, 2, \ldots$
5. n-местных (размерных) функциональных букв f^n_k, $k \geq 1$, $n \geq 0$, f^0_k называют константными буквами.
6. n-местных предикатных букв (символов) p^n_k, $k \geq 1$, $n \geq 1$.

В дальнейшем в примерах для удобства употребления будем вместо x_k писать u, v, w, x, y, z; вместо f^0_k — a, b, c, d; вместо f^n_k, $n \neq 0$, — f, g, h, φ, а вместо p^n_k — P, Q, R, S, T, V, W.

Из символов алфавита можно строить различные выражения директив. Выделим среди них те, которые представляют для нас интерес.

1. *Термы.*

а) Каждый символ переменной или константной буквы является термом.

б) Если $t_1, \ldots, t_n$, $n \geq 1$,— термы, то и $f^n_k(t_1, \ldots, t_n)$ является термом.

в) Выражение директивы является термом только в том случае, если это следует из правил а) и б).

2. *Элементарные формулы директив (атомы).*

Если p^n_k —предикатная буква, а $t_1, \ldots, t_n$ —термы, то $p^n_k(t_1, \ldots, t_n)$ — элементарная формула директивы (атом).

3. Формулы директив, или *правильно построенные формулы директив (ппфд).*

а) Всякая элементарная формула директивы есть формула.

б) Если D и B — формулы директив и x — переменная, то каждое из выражений (~D), $(D \vee B)$, $(D \wedge B)$, $(D \rightarrow B)$, $(\forall xD)$, $(\exists xD)$ есть формула директивы.

в) Выражение является формулой директивы только в том случае, если это следует из правил а) и б).

В выражениях $(\forall yD)$ и $(\exists yD)$ D называется *областью действия квантора всеобщности (общности) и квантора существования соответственно.* При этом переменная y называется связанной квантором (несвободной). (Для указания области действия кванторов будем также использовать нотацию $(\forall y)$ (D) и $(\exists y)$ (D), эквивалентную введенной выше.)

Формула называется *замкнутой,* если она не содержит свободных переменных. Нас будут интересовать именно такие формулы директив.

При определении логической системы с семантической точки зрения вводят понятия интерпретации, общезначимости и выполнимости.

Для того чтобы придать формуле директивы содержание, ее интерпретируют как предписание, касающееся рассматриваемой коммуникационной области.

Под *интерпретацией формулы директивы* будем понимать всякую систему, состоящую из непустого множества E, называемого *областью интерпретации,* и какого-либо соответствия, относящего каждой предикатной букве p^n_k некоторое *n*-местное отношение в E, каждой функциональной букве f^n_k — некоторую *n*-местную функцию в E (т. е. функцию, отображающую E^n в E) и каждой константной букве f^0_k — некоторый элемент из E. Предметные переменные мыслятся пробегающими область E интерпретации. При заданной интерпретации всякой элементарной формуле директивы приписывается значение «истинно» *(T)* или «ложно» *(F)*. Приписывание значения элементарной формуле директивы $p^n_k\ (t_1, \ldots, t_n)$ осуществляется по следующему правилу: если термы предикатной буквы соответствуют элементам из E, удовлетворяющим отношению, определяемому данной интерпретацией, то значением элементарной формулы директивы будет истина *T,* в противном случае — ложь *F*.

Значение неэлементарной формулы директивы вычисляется рекуррентно, исходя из значений составляющих ее формул. При этом, если *D* и *B* — формулы, то значения формул $\sim D$, $D \vee B$, $D \wedge B$, $D \rightarrow B$ определяются по следующей *таблице истинности:*

D	B	$\sim D$	$D \vee B$	$D \wedge B$	$D \rightarrow B$
T	T	F	T	T	T
F	T	T	T	F	T
T	F	F	T	F	F
F	F	T	F	F	T

Отметим, что формула $(\forall xD)$ обозначает утверждение: «для любого значения *x* из области E истинно (выполнено) *D»,* а формула $(\exists xD)$ обозначает утверждение: «существует такое значение *x* из области E, что истинно (выполнено) *D*».

Приведенные выше утверждения могут быть как истинны, так и ложны. В случае конечных областей E значения истинности таких формул можно установить с помощью таблиц истинности.

Очевидно, что некоторые формулы могут быть истинными или ложными в зависимости от выбранных интерпретаций.

Формула директивы *D* называется *выполнимой* тогда и только тогда, когда существует интерпретация *f* такая, что *D* принимает значение *T* в *I*. Если формула директивы *D* принимает значение *T* в интерпретации *I*, то будем говорить, что *I* есть *модель директивы D*, или *I удовлетворяет* формуле директивы *D*.

Если некоторая формула директивы принимает значение *T* при всех интерпретациях, то ее будем называть *общезначимой*. Так, например, формула $P(a)\rightarrow(P(a) \vee P(b))$ истинна при любой интерпретации (это можно установить по таблице истинности) и, следовательно, эта формула общезначима.

Если формула директивы *D* принимает значение *F* в интерпретации *I*, то будем говорить, что *I не удовлетворяет* формуле директивы *D*.

Формула директивы называется *невыполнимой (неудовлетворимой)*, если при всех интерпретациях она принимает значение *P*. Очевидно, что если формула директивы *D* общезначима, то формула директивы $(\sim D)$ невыполнима.

Введенные выше определения выполнимости, общезначимости, невыполнимости модели некоторой формулы директивы *D* переносятся на множество формул директив; при этом предполагается, что все формулы множества директив связаны знаком конъюнкции. Таким образом, некоторое множество формул директив $D_1, \ldots, D_n$ выполнено на данной интерпретации, если каждая формула директивы D_i этого множества имеет значение *T* на данной интерпретации.

Формула директивы *B логически следует* из некоторого множества формул директив $S=\{D_1, \ldots, D_n\}$, если каждая интерпретация, удовлетворяющая *S*, удовлетворяет также и *B*.

Задачей доказательства (обоснования) директив (предписаний) мы будем называть выяснение вопроса логического следования некоторой формулы *B* из заданного множества формул $\{D_1, \ldots, D_n\}$, т. е. выяснения общезначимости формулы $((D_1 \wedge \ldots \wedge D_n)\rightarrow B)$.

Однако, как показал Чёрч, не существует общего метода для установления общезначимости любых формул исчисления предикатов первого порядка. По этой причине исчисление предикатов называют *неразрешимым*. Тем не менее из теоремы Эрбрана следует, что если некоторая формула исчисления предикатов общезначима, то существует процедура для проверки ее общезначимости, т. е. исчисление предикатов можно назвать *полуразрешимым*.

При формировании формул директив оказывается более удобным определять невыполнимость, а не общезначимость. Поэтому рекомендуется рассматривать формулу $\sim((D_1 \wedge \ldots \wedge D_n)\rightarrow B)$,

являющуюся отрицанием исходной. Формула $\sim((D_1 \wedge \ldots \wedge D_n) \rightarrow B)$ эквивалентна формуле $(D_1 \wedge \ldots \wedge D_n \wedge \sim B)$, и именно невыполнимость этой последней формулы и следует доказывать (обосновывать). Для установления невыполнимости необходимо доказать (обосновать), что не существует такой интерпретации, при которой каждая из формул множества $D_1, \ldots, D_n, \sim B$ имеет значение T.

В связи с полуразрешимостью исчисления предикатов эта процедура (директива) будет приводить к успеху только в случае, если формула B следует из $D_1, \ldots, D_n$. В противном случае процедура (директива) может продолжаться бесконечно.

Процесс установления невыполнимости некоторого множества формул директив будем называть *процессом опровержения директивы*.

Как мы указали выше, кроме определенного нами семантического способа задания логической теории комуникационных предписаний (директив), существует синтаксический способ. При этом способе, кроме алфавита и формул директив, определяемых так же, как и раньше, задаются **аксиомы и правила вывода директив (предписаний).**

Аксиомами называют некоторое выделенное множество формул теории. Обычно существует возможность эффективно выяснить, является ли данная формула теории А аксиомой. В таком случае А называется *аксиоматической теорией*.

Правилами вывода формул директив (предписаний) будем называть конечное множество $R_1, \ldots, R_n$ отношений между формулами директив (предписаний). Для каждого отношения R_i существует такое целое положительное число j, что для каждого множества $D_1, \ldots, D_j$ формул и для каждой формулы B эффективно решается вопрос о том, находятся ли эти j формул в отношении R_i с формулой B, и если да, то B называется *непосредственным следствием* данных j формул по правилу R_i.

Выводом **в теории комуникационных предписаний (директив) называется такая последовательность формул $D_1, \ldots, D_n$, в которой для любого i формула D_i есть либо аксиома теории комуникационных предписаний (директив), либо непосредственное следствие каких-либо предыдущих формул по одному из правил вывода.**

Формулу B теории комуникационных предписаний (директив) будем называть *определением* теории комуникационных предписаний (директив), если существует вывод в этой теории, в котором последней формулой является B. Такой вывод будем назывть *выводом формулы директивы B*.

Теория комуканиционных предписаний (директив) называется *разрешимой,* если существует единая эффективная процедура (алгоритм), позволяющая узнать для любой данной формулы, существует ли ее вывод в теории комуникационных предписаний (директив).

Логическая теория комуникационных предписаний (директив) называется *непротиворечивой,* если не существует формулы *B* такой, чтобы *B* и (~*B*) были определениями в теории комуникационных предписаний (директив).

Известно, что всякое исчисление первого порядка непротиворечиво.

Теорема Гёделя о полноте устанавливает эквивалентность семантической и синтаксической точек зрения:

Во всяком исчислении предикатов первого порядка теоремами являются все те и только те формулы, которые логически общезначимы.

Итак, мы определили язык исчисления предикатов первого порядка для записи коммуникационных предписаний и директив, являющихся исходными данными { $D_1, \ldots, D_n$ }, и определения *B,* справедливость которого следует установить. Справедливость определения *B* сводится к доказательству того, что формула $((D_1 \wedge \ldots \wedge D_n) \rightarrow B)$ является общезначимой (т. е. является директивой).

Для определения невыполнимости и выводимости формулы ее удобно представить в виде **дизъюнктов (предписаний).** Всякую формулу предписания можно представить в виде дизъюнктов, применив к ней последовательность приведенных ниже простых операций.

1. Переименование переменных. Выполняется такая замена переменных, что все переменные, связанные кванторами, становятся различными. Например, $\forall x R(x) \vee \forall x S(x)$ переписывается в виде $\forall x R(x) \vee \forall y S(x)$.

2. Исключение знака импликации. Всякий раз, когда встречается $\rightarrow$, делается замена $(A \rightarrow B)$ на $((\sim A) \vee B)$.

3. Уменьшение области действия связки ~. Везде, где возможно, делаются замены:

Заменяется $\sim\sim A$ на A.

Заменяется $\sim(A \vee B)$ на $\sim A \wedge \sim B$.

Заменяется $\sim(A \wedge B)$ на $\sim A \vee \sim B$.

Заменяется $\sim(\forall x A)$ на $\exists x(\sim A)$.

Заменяется $\sim(\exists x A)$ на $\forall x(\sim A)$.

В конце концов получается формула, где связка встречается непосредственно перед атомной формулой.

4. Исключение кванторов существования. Вычеркиваются поочередно кванторы существования. При этом каждая переменная y, связанная квантором существования, заменяется на $g(x_1, ..., x_m)$, где g — символ новой (отличной от имеющихся в формуле) функции, а $x_1, ..., x_m$ — все переменные, встречающиеся в кванторах всеобщности, области действия которых содержат вычеркиваемый квантор существования. Если таких переменных нет, то y заменяется на новую константу.

5. Приведение к предваренной нормальной форме. Все кванторы общности переносятся влево в начало формулы, так что формула принимает вид $\forall x_1 \forall x_2 \ldots \forall x_n A$, где A не содержит кванторов.

6. Приведение к конъюнктивной нормальной форме. Приведение осуществляется заменой, пока это возможно, $(A \wedge B) \vee C$ на $(A \vee C) \wedge (B \vee C)$.

В результате применения шагов 1—6 получаем выражение

$$\forall x_1 \forall x_2 \ldots \forall x_n (A_1 \wedge \ldots \wedge A_n),$$

где A_i имеет вид $(l^i_1 \vee \ \ldots \ \vee \ l^i_r)$, а l^i_j есть атомная формула или ее отрицание. Атомную формулу или ее отрицание будем называть *литерой*. Если A — атомная формула, то литеры A и $\sim A$ будем называть *дополнительными (комплементарными) литерами*.

***7.* Исключение кванторов всеобщности**. Так как все переменные связаны кванторами всеобщности, а порядок расположения кванторов безразличен, то не будем указывать кванторы явным образом. Будем называть этот вид представления бескванторной нормальной формой.

8. Исключение связок $\wedge$. Исключаем связку $\wedge$, заменяя $A \wedge B$ на две формулы A, B. В результате многократной замены получим множество формул, каждая из которых представляет собой дизъюнкцию литер, называемую *предписанием (дизъюнктом)*.

Исходное множество формул A является невыполнимым тогда и только тогда, когда невыполнимо множество A', полученное из A применением указанных восьми операций.

Рассмотрим теперь процесс поиска доказательства (обоснования). Покажем, что он может быть представлен в виде поиска пути на графе специального вида, называемом *графом доказательства предписаний и/или директив*. Задача доказательства начинается с непустого исходного множества формул B_0 и множества правил вывода R. Если $\varphi \in R$, а B есть некоторое множество формул, то $\varphi(B)$ есть множество выводимых формул.

$\varphi(B) = \emptyset$, если φ не применимо к B. В частности, $\varphi(B) = \emptyset$, если B не является конечным. Пусть B^* будет множество всех формул, которые

могут быть выведены из B_0 повторным применением правил из R. Тогда каждое $\varphi \in R$ есть функция $\varphi: 2^{B^*} \to 2^{B^*}$, определенная на подмножествах B^* и принимающая в качестве значений подмножества B^*. Каждой формуле $C \in B^*$ может быть присвоен уровень: если $C \in B_0$, то C присваивается нулевой уровень, в противном случае $C \in \varphi(B)$ для некоторого $\varphi \in R$, и для некоторого $B \subseteq B^*$ уровень C на единицу больше, чем уровень некоторой формулы $D \in B$, имеющей максимальный уровень в B. Если B_i есть множество всех формул уровня i, то $B^* = \bigcup_i B_i$. Формула $C \in B^*$ может иметь несколько различных выводов, поэтому формула C может иметь несколько уровней. Так как $\varphi(B) \neq \emptyset$, только если B является конечным, то множество формул, которые встречаются в данном выводе формулы $C \in B^*$, всегда является конечным.

Итак, **задача доказательства предписаний (директив) для тройки (B_0, R, F), $F \subseteq B^*$ (где F — множество терминальных формул) состоит в генерировании с помощью некоторой стратегии формирования $\sum$ формулы предписания (директивы) $C^* \in F$ повторным применением правил из R, начиная с формул в B_0.**

Тройка (B_0, R, F) определяет направленный граф, чьи вершины являются формулами $C \in B^*$. C' является непосредственным преемником C (т. е. C' связано с С дутой, направленной из C в C'), если для некоторого $B \subseteq B^*$ и $\varphi \in R$ $C \in B$ и $C' \in \varphi(B)$. Как уже было отмечено выше, формула C может иметь несколько выводов, т. е. несколько путей в графе, определяемом тройкой (B_0, R, F). Удобно представлять единственную вершину d как различные вершины $n_2, \ldots, n_k$ в некотором графе (называемом квазидеревом), если вершина d может быть получена различными путями из начальной вершины a. Такое взаимно однозначное соответствие между вершинами и выводами (путями) позволяет рассматривать число вершин, генерированных стратегией $\sum$ в ходе вывода, как меру эффективности $\sum$ для данной задачи.

Определим теперь понятие *абстрактного графа доказательства предписаний (директив)* (G, s), который может быть интерпретирован как рассмотренная нами выше задача доказательства предписаний (директив) (B_0, R) пометкой вершин $n \in G$ метящей функцией $c: G \to R^*$ и рассмотрением каждого применения функции s к подмножеству $G' \subseteq G$ как применения функции $\varphi \in R$ к подмножеству $\{c(n) | n \in G'\}$.

Абстрактный граф доказательства предписаний (директив) есть пара (G, s), где G —множество вершин, а s: $2^G \to 2^G$ есть функция

преемствования, определенная на подмножествах G и принимающая в качестве значений подмножества G.

G и s удовлетворяют следующим условиям:

1. $s(\emptyset)=\emptyset$.

2. Если $s(G')\neq\emptyset$, то G' является конечным множеством.

3. Если $G'\neq G''$, то $s(G')\cap s(G'')=\emptyset$.

4. Пусть $G_0=\{n\in G \mid n\notin s(G') \text{ для любого } G'\subseteq G\}$, $G_{k+1}=\{n\in G \mid n\in s(G') \text{ для некоторого } G\subseteq \bigcup_{i\leq k} G_i, G'\cap G_k\neq\emptyset\}$.

Тогда

а) $G_0\neq\emptyset$.

б) $G=\bigcup_{0\leq i} G_i$.

в) $G_i\cap G_j\neq\emptyset$ для $i\neq j$.

Условие 3 утверждает, что различным множествам вершин соответствуют различные множества преемников. Именно это условие обеспечивает то, что граф (G, s) является квазидеревом. Условие 4 определяет, что в графе (G, s) каждой вершине $n\in G$ может быть присвоен единственный уровень i, т. е. $n\in G_i$ и $n\notin G_j$ для всех $i\neq j$. Если (B_0, R) является интерпретацией (G, s) с метящей функцией $c: G\to B^*$, то $B_i=\{c(n) \mid n\in G_i\}$ есть множество помеченных вершин уровня i. **Условие 3 гарантирует, что для каждой формулы $C\in B^*$ и для каждого вывода C из B_0 существует своя вершина $n\in G$ такая, что $C=c(n)$.**

Отметим, что не требуется конечность множеств G_0 и B_0. Это позволяет иметь дело со схемами аксиом и потенциально бесконечными множествами начальных вершин G_0.

Функции преемствования s графа (G, s) определяет частичное упорядочение вершин в графе G: n' есть непосредственный преемник n (а n есть непосредственный предшественник n'), если $n'\in s(G')$ и $n\in G'$ для некоторого $G'\subseteq G$. Вершина n' есть *преемник* n (*n — предшественник n'*) и обозначается $n'>n$, если n' есть непосредственный преемник n или если n' есть преемник непосредственного преемника n. Будем записывать $n\leq n'$, если $n<n'$ или $n=n'$. Определение графа (G, s) гарантирует, что для всех $n\in G$ множество $\{n' \mid n'\leq n\}$ является конечным, хотя множество $\{n' \mid n'\geq n\}$ может быть бесконечным. Заметим, что в интерпретации графа (G, s) вывод формулы $c(n)$ состоит из всех формул $c(n')$, где $n'\leq n$. Каждый такой вывод содержит конечное количество формул $c(n')$.

На рис. 2.9 приведен пример графа (G, s).

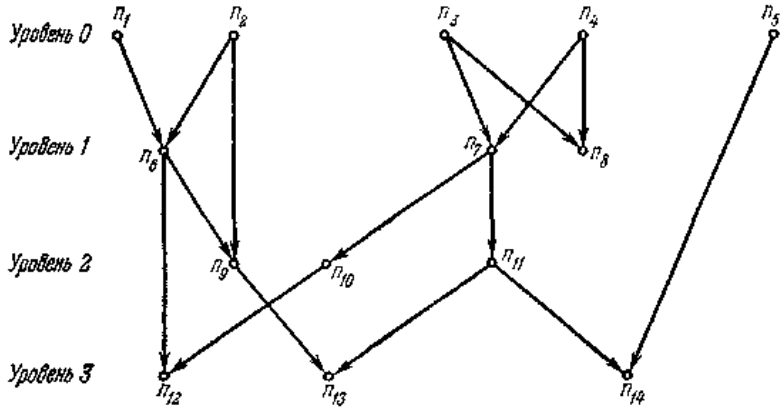


Рис. 2.9. Абстрактный граф доказательства директив.

Вершины графа представляются точками, а точки n и n' связываются дугой, направленной от n к n', если n' — непосредственный преемник n. Удобно изображать эти графы направленными вниз так, что n лежит выше n'. Для того чтобы определить на рис. 2.9, принадлежит ли n множеству $s(G')$, достаточно проверить, что G' есть множество всех тех вершин, которые связаны с n дугой, направленной к n'. Так, в приведенном примере

$$s(n_1, n_2) = \{n_6\},$$
$$s(n_2, n_6) = \{n_9\},$$
$$s(n_3, n_4) = \{n_7, n_8\},$$
$$s(n_7) = \{n_{10}, n_{11}\},$$
$$s(n_2) = s(n_5) = s(n_8) = s(n_1, n_2, n_6) = \Phi.$$

Абстрактная задача доказательства директив может быть представлена в виде четверки $P=(G, s, F, g)$, где $F \subseteq G$ является множеством терминальных (решающих) вершин для P, и g есть некоторая оценка, выражающая меру сложности вывода. Далее мы приведем конкретные алгоритмы поиска и правила вывода, интерпретирующие абстрактную задачу доказательства директив.

2.5.1. Применение метода доказательства предписаний (директив)

К задачам теории коммуникаций, которые могут интерпретированны в виде доказательства предписаний (директив) может быть сведен очень широкий круг задач коммуникаций. Это позволяет не только

отвечать на вопрос, следует ли логически директива (формула) C из некоторого множества $D_1, \ldots, D_n$ директив. Он позволяет отвечать на вопрос, следует ли из исходного множества директив директива $(\exists x C(x))$, и если следует, то каково то частное значение переменной x, при котором это имеет место. Умение строить удовлетворяющие частные случаи для переменной, относящейся к квантору существования, позволяет ставить довольно общие вопросы. Например, мы могли бы задаться вопросом: «Существует ли такая последовательность действий программы для случая игры в шахматы, которая приводит к ее победе?» Следует, однако, помнить, что сложные вопросы могут привести к доказательствам настолько сложным, что при быстродействиях существующих вычислительных машин и ограниченном времени решения эти доказательства не будут найдены. Кроме того, надо не забывать о полуразрешимостп исчисления предикатов.

Метод доказательства предписаний (директив) может быть использован в сочетании с другими подходами. Рассмотрим применение метода доказательства предписаний (директив) для решения задач в пространстве состояний. Будем описывать состояния в виде правильно построенных формул (ппф) исчисления предикатов. При этом операторами являются преобразования, заменяющие одно множество формул другим (например, «список вычеркиваний» и «список добавлений»). Множество состояний, к которым применим данный оператор, задается с помощью предусловий, также записанных в виде ппф. В такой системе методы доказательства предписаний (директив) можно использовать для проверки выполнения условий достижения цели и условий применимости операторов.

Возможна некоторая модификация описанного метода, используемая в вопросно-ответной системе. В предыдущем способе преобразования, выполняемые операторами, отображают одни множества формул в другие. Но эти преобразования совершаются независимо от системы логического вывода в исчислении предикатов. Включения действия оператора в рамки формализма исчисления предикатов можно добиться путем введения в каждый предикат терма состояния, указывающего состояние, к которому предикат применим. **При такой формулировке операторы рассматриваются как функции, отображающие одно состояние в другое, а их действия выражаются в виде дополнительных аксиом, которые можно объединить с формулами, описывающими начальное состояние.** Так значением оператора $f(s)$ будет новое состояние, возникающее в результате применения оператора f к состоянию s.

Если наша цель состоит в создании состояния *s*, удовлетворяющего некоторой целевой формуле *B*(*s*), то эту задачу можно решить формально, найдя доказательство для формулы $\exists s B(s)$) и определив частное значение переменной *s*. **Ответ будет содержать директиву для целевого состояния в форме композиции операторных функций.**

Приведем пример, поясняющий суть данного подхода. Пусть некоторое состояние *S* в мире объектных понятий *(R)* описывается следующим фактом:

*F*1*:* At*(R, A,s0)* — объект находится в точке *A* в состоянии s_0 и объект умеет выполнять следующее действие (оператор):

*f*1: объект перемещается из точки *x* в точку *y*.

Основной эффект применения оператора *f*1 можно описать с помощью формулы

$$\forall x \ \forall y \ \forall s(\mathrm{At}\,(R, x, s) \rightarrow \mathrm{At}(R, y, f1\ (x, y, s))),$$

означающей, что для всех *s, x* и *y,* если объект находится в точке *x* в состоянии *s,* то в состоянии, возникающем в результате применения оператора *f*1 к состоянию s, объект окажется в точке *y*.

Цель задачи состоит в определении последовательности действий, переводящих объект из точки *A* в точку *C,* т. е. в доказательстве истинности формулы: $(\exists s\ (\mathrm{At}\,(R, C, s)))$.

Очевидно, что решение получается непосредственно из *F*1 и *f*1, так что результирующее состояние $s = f1\ (A, C, s_0)$.

2.6. Обобщенные декларативные методы формирования предписаний (директив)

В предыдущих параграфах мы описали представления задач, сформулированных в виде эвристического поиска, и задач доказательства предписаний (директив). При этом задачи эвристического поиска рассматривались нами в продукционной и редукционной системах. В первой из них решение сводится к поиску решающего пути в графе пространства состояний, причем поиск может осуществляться от начальных состояний к конечным (однонаправленный поиск) или (в случае явного задания целевых состояний) одновременно от начальных состояний к конечным и от конечных к начальным с замыканием общего решающего пути в одном из промежуточных состояний (двунаправленный поиск).

В редукционной системе решение сводится к поиску решающего графа в пространстве описаний множества подзадач и последующей композиции решения задачи из решений образующих ее подзадач.

Наконец, в задаче доказательства предписаний (директив) решение сводится к выводу целевого предписания (директивы) из исходного множества аксиом на основе правил вывода.

Несмотря на столь различные внешне постановки задач, поиск решения, особенно при однотипном задании механизма управления поиском, осуществляется в графах, определяющих один и тот же класс пространств поиска. Выявление общих закономерностей пространств поиска представляет значительный интерес как с точки зрения более глубокого понимания процессов решения коммуникационных задач в декларативных предписаниях и взаимосвязей между различными формализациями, так и с точки зрения построения единого формализма решения коммуникационных задач, из которого могут быть выведены частные декларативные предписания.

В настоящем параграфе изложим ряд основных предпосылок и идей построения обобщенных коммуникационных предписаний и одну, кажущуюся нам наиболее целесообразной постановку коммуникационной задачи в таком предписании.

Прежде всего, процессы поиска в пропозициональных графах и графах доказательства директив (ГДД) работают в прямо противоположных направлениях. Действительно, в ГДД мы начинаем с задания множества аксиом (заведомо разрешимых задач) и на каждом шаге последовательно наращиваем это множество выведенными на этом шаге директивами до тех пор, пока не выведем целевое предписание.

В пропозициональных графах процесс происходит обратным образом. Мы начинаем с задачи, которую следует решить (целевое предписание), и на каждом шаге выводим из построенного к этому моменту множества подзадач альтернативные совокупности подзадач этих подзадач. Решение будет получено, когда множество подзадач будет полностью состоять из заведомо разрешимых задач (аксиом).

Пусть нам дана задача *T,* причем ее решение в системе редукций выглядит словесно следующим образом: «задача *T* может быть решена, если решены подзадачи *A* и *B* или подзадачи *B* и *C,* подзадача *A* решается, если решается подзадача *D,* подзадача *B* решается, если решаются подзадачи *E, F* или подзадача *G,* подзадача *C* решается, если решается подзадача *G, D, E, F, G* — разрешимые подзадачи».

Соответствующее представление в виде пропозиционального дерева приведено на рис. 2.10, *а*. Все различные вхождения одной и той же подзадачи представлены различными вершинами (по определению дерева).

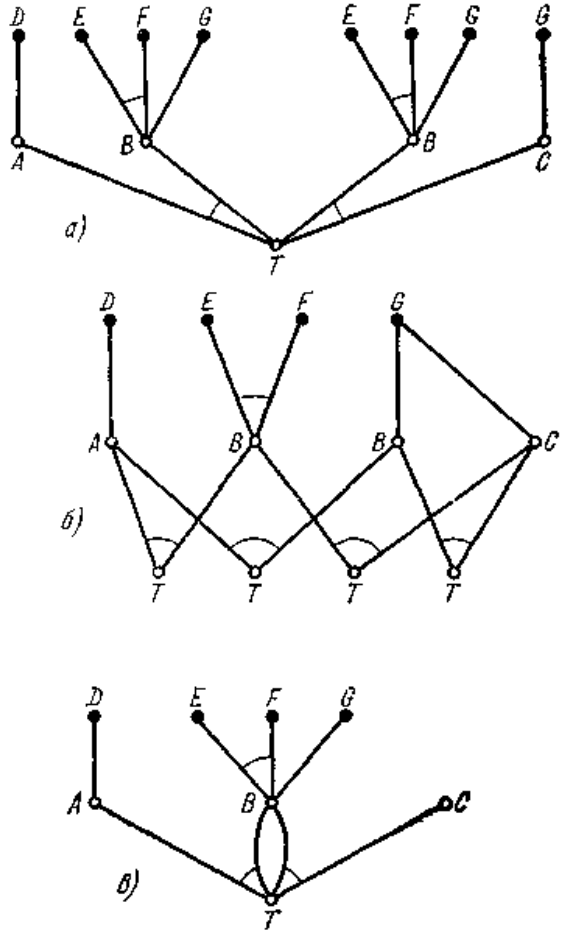


Рис. 2.10. Представление задачи *T* в виде пропозиционального дерева (а), графа доказательства директив *(б)* и графа вывода *(в)*.

Та же задача с помощью доказательства директив решалась бы следующим образом: «даны аксиомы *D, E, F* и *G*, из *D* выводится *A,* из *E* и *F* выводится *B,* из *G* выводится *B,* из *G* выводится *C*, из *A* и *B* выводится *T,* из *A* и *B* выводится *T,* из *B* и C выводится *T,* из *B* и *C* выводится *T*». Соответствующее представление в виде ГДД приведено

на рис. 2.10, *б*. И здесь различным вхождениям одной и той же директивы соответствуют различные вершины.

Директивы (рис 2.10, *а* и 2.10, *б)* одного и того же решения одной и той же задачи различны. Однако переинтерпретируем эти директивы таким образом, чтобы директиве соответствовала вершина графа, а различным способам ее вывода (образования ее как подзадачи) соответствовали различные дуги, инцидентные этой вершине. На рис. 2.10, *в* приведена такая директивы, являющаяся единной для обоих исходных директив.

Теперь решение задачи T сводится к поиску некоторого решающего подграфа в графе вида, предстявленного на рис. 2.10, *в*. Мы можем скомбинировать оба направления поиска (от аксиом к цели и от задачи к разрешимым подзадачам) в единый двунаправленный процесс с окончанием в «точке встречи». Далее, если разрешить выводы только из одной посылки (в направлении сверху вниз на рис. 2.10, б) или исключить использование конъюнктивных вершин (в направлении снизу вверх на рис. 2.10, б), то частным случаем этого графа будет граф в пространстве состояний. Поиск решающего подграфа в таком вырожденном графе сводится к поиску решающего пути. Таким образом, мы получаем граф в пространстве состояний как частный случай нашего графа. В этом вырожденном графе мы можем осуществить как однонаправленный, так и двунаправленный поиск.

Прежде чем перейти к формальному определению графа в обобщенном декларативном предписании, сделаем упоминание еще об одном обобщении, получаемом введением указанного графа. Как в редукционных, так и в продукционных системах предписаний мы отождествляли вершины графа с состояниями (описаниями подзадач), а его дуги с операторами, преобразующими одни состояния в другие. Построение графа в обобщенном предписании позволяет ввести несколько другую интерпретацию. Мы будем рассматривать коммуникационные системы, в которых задается начальное множество состояний $S_o \subseteq S$ и начальное множество операторов $F_0 \subseteq F$. Результатом акта вывода (S_i, F_j), $S_i \subseteq S$, $F_j \subseteq F$, если он определен, может быть либо $s \in S$, либо $f \in F$. Другими словами, мы допускаем, что некоторые из операторов могут отождествляться с вершинами графа, а множество операторов не является фиксированным, а наращивается в ходе решения задачи.

Назовем граф вида, представленного на рис. 2.10, *в, графом вывода директив* и введем некоторые формальные фрагменты определения, касающиеся этого графа.

Мы определяем граф вывода директив как пару (S, F), где S — множество вершин, отождествляемых с фрагментами директив,

$F: 2^S \rightarrow S$ — функция следования, отображающая множество вершин в одну вершину. Эту функцию мы назовем *оператором вывода директив*.

Пусть S — фрагмент директивы, следующей непосредственно из конечного числа n фрагментов директив $S_1, S_2, \ldots, S_n$, путем применения одного оператора вывода. Фрагмент директивы S связан с каждым из S_i, i=l, 2, . . ., n, дугой, образуя *конъюнктивный пучок*. $S_i, S_2, \ldots, S_n$ называются *посылками, S — заключением*. Посылки, заключение и связывающий их конъюнктивный пучок образуют один *акт вывода директив*.

В частности, мы рассматриваем каждую аксиому S_0 как один акт вывода директивы с пустым множеством посылок и заключением S_0.

Выводом директивы D назовем конечное множество фрагментов директив и связывающих их актов вывода такое, что

1) Каждое $S \in D$ принадлежит по крайней мере одному акту вывода в D.

2) Точно один фрагмент директивы $C \in D$ является заключенем акта вывода, не являясь посылкой какого-либо акта вывода.

3) Каждый фрагмент директивы $S \in D$ является заключением не более чем одного акта вывода в D.

4) Вывод D не содержит бесконечных ветвей вида $S_1, S_2, \ldots, S_n, S_{n+i}, \ldots$, где $S_1 \in D$, S_{n+1} — посылка акта вывода в D, имеющего заключением S_n (т. е. граф вывода не содержит циклов).

Посылками вывода директивы D называются те фрагменты директивы из D, которые не являются заключениями какого-либо из актов вывода, принадлежащих D. *Заключением вывода директивы D* является $C \in D$.

Мы назовем вывод *беспосылочным* (БВ), если у него нет посылок, и *редукционным* (РВ), если заключение этого вывода — данная целевая директива. *Решающим выводом директивы* будем называть беспосылочный редукционный вывод, полученный поиском одновременно в двух направлениях.

Итак, во введенном формализме решение задачи сводится к *поиску решающего вывода в графе вывода*. При этом последовательно образующиеся в процессе решения БВ порождают граф доказательства директив, а РВ — пропозициональный граф. Граф пространства состояний порождается выводами, акты которых содержали бы не более одной посылки.

2.7. Проблема коммуникационных границ в декларативно представляемых предписаниях (директивах)

Создание и организация информационного контекста, связанного с шагами процесса РКЗ формирования коммуникационных предписаний и директив, в значительной степени определяет эффективность этого процесса.

В коммуникационных системах на применение любых формализмов описания накладываются ограничения количественного порядка.

Предположим, что в поисках плана решения задачи формирования коммуникационных предписаний и директив разработчик имеет в своем распоряжении в среднем 6 операторов, применимых и эвристически обоснованных в каждом состоянии (формуле) директивы; предположим, что типичная задача формирования коммуникационных предписаний или директив решается последовательностью из 4 операторов. Тогда поисковое дерево будет иметь около $6^4 \approx$ 1300 вершин (мы включаем сюда возможность хранения альтернативных планов формирования коммуникационных предписаний или директив). Будем считать, что разработчик работает в среде умеренной сложности. Тогда для полного описания каждого состояния формируемой директивы может потребоваться хранение около 1000 элементарных фрагментов директив, **касающихся местоположения всех предметов и их признаков, указания всех отношений между ними и т. д.** Оказывается, что только для описания всех состояний в поисковом дереве требуется хранить свыше миллиона фрагментов директив! А поскольку каждый фрагмент директивы сам представляет из себя сложную списочную структуру, то становится ясным, что проблема сокращения числа обрабатываемых описаний состояний становится весьма острой. Очевидно, что каждое действие, совершаемое оператором, вызывает изменения лишь в небольшом подмножестве множества фрагментов директив. Казалось бы, что каждому состоянию или действию можно сопоставить список лишь изменяющихся фрагментов директив, а остальные фрагменты директив могут храниться в общей базе данных директив в течение всего процесса решения задач. Однако следующий элементарный пример покажет принципиальные трудности сокращения числа обрабатываемых в ходе РКЗ описаний и позволит нам сформулировать *проблему комуникационных границ*.

Пусть некоторое состояние S в пространстве отправителя описывается следующими фактами (фрагментами директив):

*F*1*:* отправитель находится в позиции *A;*
*F*2*:* ящик *B*1 находится в позиции *B*;
*F*3*:* ящик *B*2 находится на ящике *B*1;
*F*4*:* допустимые позиции — *{A, B, C, D},*
и отправитель умеет производить следующие действия (множество операторов (директив)):
*f*1: отправитель идет из *x* в *y;*
*f*2: отправитель толкает *B*1 из *x* в *y, x, y* ∈ *{A, B, C, D}.*
Рассмотрим две задачи:
*P*1*:* отправитель должен быть в *C*;
*P*2*:* ящик *B*1 должен быть в *C.*
Очевидно, что задача *P*1 решается с помощью оператора (директивы) *f*1, причем после решения этой задачи факты (фрагменты директив) *F*2, *F*3, *F*4 остаются неизменными, а *F*1 меняется на
*F*1*':* отправитель находится в позиции *C.*
Задача *P*2 решается с помощью *f*2, причем меняются факты (фрагменты директив) *F*1 и *F*2, в то время как *F*3 и *F*4 остаются неизменными.
*F*1*″:* отправитель находится в позиции С;
*F*2*″:* ящик *B*1 находится в позиции *C.*
Возникает вопрос, каким образом сократить описание новых состояний *S' = {F*1', *F*2, *F*3, *F*4*}, S″={F*1″, *F*2″, *F*3, *F*4*}.*
Казалось бы, что сокращение может быть достигнуто введением специальных директив типа
*A*1*:* определить меняющиеся факты (фрагменты директив) сопоставлением условий задачи описанию состояния *S.*
Такая директива могла бы определить, что в первой задаче должен измениться только тот факт, который относится к местоположению отправителя. Но в процессе состояния плана решения *P*1 отправитель может выяснить, что на кратчайшем пути в позицию *C* стоит ящик *B*1 и лучшим решением, чем обход ящика, является решение *f*2. При этом условия задачи выполняются, но директива *A*1 терпит неудачу.
Может показаться, что следует привязать изменяющиеся факты к описанию операторов введением, например, директивы
*A*2*:* указать факты, изменяющиеся каждым оператором.
Тогда, если задачу, будь то *P*1 или *P*2, решает оператор *f*2, то системе было бы указано, что надо менять факты, связанные с положением отправителя и *B*1, т. е. *F*1 и *F*2. Однако директива *A*2 потерпела бы неудачу, если бы в множестве фактов в состоянии S присутствовал хотя бы один факт, выведенный из других фактов. Например, из *F*2 и *F3* можно вывести
*F*5*:* ящик *B*2 в положении *B,*

и этот факт не изменится директивой *A*2, хотя после *f*2 *B2* будет в *C*.

До сих пор мы рассматривали однооператорные (однодериктивные) решения задач. Если же рассмотреть задачу

*P*3*:* отправитель должен быть в *D,* и *B*2 должен быть в *C,*

то легко видеть, что для решения этой задачи решатель должен на каждом шаге решения иметь доступ ко всему множеству фактов, включая выведенные следствия.

Можно привести более яркий своей кажущейся абсурдностью пример серьезности проблемы полных описаний состояний. Для того чтобы человек *p* вступил в телефонный разговор с человеком *q* (речевой комунникационный процесс) директивы, казалось бы, достаточно, чтобы *p* нашел номер телефона *q* в телефонной книге и набрал этот номер. Решатель задач, построивший такой план, потерпел бы неудачу в случае, если

— страница с номером телефона *q* вырвана,
— человек *p* слепой,
— кто-то залил чернилами нужный номер,
— телефонная компания сделала ошибки в коммутации,
— телефон *q* не значится в телефонной книге,
— телефонная линия в этот момент неисправна,
— человек р потерял голос и т. д.

Для учета всех этих возможностей формальной коммуникационной системе должны быть заданы дополнительные фрагменты директив или условия, исключающие неопределенность ситуации, однако предугадать все возможности практически невозможно.

Теперь мы можем сформулировать *проблему коммуникационных границ* в достаточно общем виде.

Необходимо, чтобы решатель задач (разработчик директив), имея полное описание состояний предметов, мог разграничить (отсюда название проблемы) фрагменты директив, которые должны изменяться в результате некоторого действия, от фрагментов директив, которые остаются неизменными в результате этого действия, и делал бы это эффективно с эвристической точки зрения.

В более общих понятиях, решатель задач должен, имея эпистемологически полное представление о фрагментах директив в коммуникационной системе, обладать эвристически эффективной способностью выделять минимально необходимое подмножество фрагментов директив, имеющих отношение к текущей стадии решения задачи, оставляя прочие фрагменты директив без внимания.

Формально проблема коммуникационных границ может быть представлена следующим образом.

Допустим, что s_1 и s_2 — различные состояния предметов, причем s_2=apply (f, s_1), функция apply означает «применить оператор f к состоянию s_1». Мы могли бы описать результат действия оператора (директивы) f как

$$A(s_1) \rightarrow B(\text{apply}(f, s_1)), \qquad (2.17)$$

где A и B — множества измененных и новых фрагментов директив соответственно, представляющие некоторые короткие директивы. Однако если мы должны указать все фрагменты директив, которые, будучи истинными в состоянии s_1, не изменились в состоянии s_2, и если множество таких фрагментов директив $F=\{F_i/i=1, 2, \ldots, n\}$, то результат apply ($f$, s_1) может быть записан в виде

$$\bigwedge_{i=1}^{n} F_i(s_1) \wedge A(s_1) \rightarrow \bigwedge_{i=1}^{n} F_i(\text{apply}(f, s_1)) \wedge \wedge B(\text{apply}(f, s_1)), \qquad (2.18)$$

где n может быть весьма большим.

Таким образом, мы получаем длинные выражения законов действия (директив) в пространстве коммуникаций S. Решение проблемы коммуникационных границ — замена (2.18) на (2.17).

Прежде чем описать основные подходы к решению проблемы коммуникационных границ, укажем некоторые ее особенности.

Отметим принципиальный и количественный аспекты проблемы коммуникационных границ. Принципиальный характер проблемы подтверждается следующими аргументами:

1. В любой коммуникационной системе всегда будет стоять задача приведения в соответствие *растущих знаний* системы предписаний и уменьшающейся из-за роста количества фрагментов директив *эвристической эффективности предписаний*. Мы должны решить задачу **разумной организации коммуникационной системы** или, другими словами, четко разграничить области предписаний, относящиеся к классам действий коммуникационной системы, а также пути воздействия последних на изменение сложного коммуникационного процесса. Проблема коммуникационных границ представляет собой частный случай этой общей задачи.

2. В любой коммуникационной системе запись действий (директив) в виде (2.17) не дает нам гарантии, что эта запись является адекватной раз и навсегда. Истинность того или иного фрагмента директивы может зависеть не только от ее связи с тем или иным действием (директивой) непосредственно, но и от более детального анализа обновляющейся совокупности фрагментов директив. Иными словами, часто истинность фрагмента директивы сможет быть установлена лишь в результате длинной цепи предписаний. Конечно, «лобовым»

решением этой задачи был бы полный вывод всех возможных следствий множества фрагментов директив в каждом из состояний предмета. Однако мы относим эту проблему к принципиальному аспекту хотя бы потому, что множество таких следствий может быть бесконечным.

3. Особую сложность проблема коммуникационных границ представляет для коммуникационных систем, работающих в *динамическом режиме,* т. е. в режиме, где директивы (предписания) могут не только исходить от коммуникационной системы, но и быть независимыми от нее. В этом случае результаты действия директив, записанные в виде (2.18), могут потерять свою истинность, как только на множестве S определяется новый предикат (в результате независимого действия директивы).

Существующие подходы к решению проблемы коммуникационных границ в основном касаются количественного аспекта этой проблемы, т. е. выделения минимального списка фрагментов директив, относящихся к тем или иным действиям.

Краткий обзор этих подходов мы начнем с изложения идеи *метода коммуникационных границ*. Коммуникационная граница представляет классификацию фрагментов директив, независимую в том смысле, что некоторое действие может изменять фрагменты директив, относящиеся только к одному классу, не меняя остальных. К сожалению, если такая классификация и может быть получена, то она будет весьма грубой для всех практически важных задач формирования предписаний. Идея такой классификации еще не получила развитие в теории коммуникаций. Каждому действию соответствует некоторое малое множество фрагментов директив, на которые это действие прямо влияет. Мы не можем предполагать, что все остальные фрагменты директив остаются неизменными, потому что они могут быть **соединены длинными причинно-следственными цепями с изменяющимися фрагментами директив**. Обозначим бинарное отношение причинной связи через $\mathcal{R}$. Тогда $a\mathcal{R}b$ означает, что некоторый фрагмент директивы a будет изменяться, если будет изменяться причинно-связанный с ними фрагмент директивы b. Если мы можем доказать, что $\sim(a\mathcal{R}b)$, то это означает, что никакие изменения b не вызывают изменений в a. Теперь при выполнении некоторого действия достаточно проверить, что a не связано причинной связью ни с одним из фрагментом директивы, изменяемых действием.

Этот подход предполагает, что изменения в коммуникационных процессах не происходят спонтанно и что имеется только **один источник (<u>отправитель</u>) действия (директивы)**. Это обстоятельство вызывает сомнение в том, что метод коммуникационных границ без

привлечения новых средств сможет разрешить принципиальные аспекты проблемы коммуникационных границ. Что касается количественного аспекта, то этот метод вряд ли может быть использован в сколько-нибудь сложном решателе, поскольку неизменность большого количества фрагментов директив должна передоказываться в каждом состоянии предписания или директивы, что ничем по существу не отличается от обработки полных описаний предписаний и состояний предметов. Одним из возможных направлений развития этого метода является введение модальностей в логику первого и высших порядков, совокупно, хотя и приближенно, **описывающих причинные отношения между фрагментами директив.** Однако это направление находится на стадии постановки, и требуется детальное исследование его возможностей, прежде чем можно будет перейти к его практической реализации.

Близко по духу к методу коммуникационных границ и другое направление, основанное на анализе непротиворечивости, — *метод контрфактов,* или выявления нереальных ситуаций. Идея метода заключается в том, что после выполнения действия все фрагменты директив, которые были истинными в состоянии s_1, считаются истинными и в состоянии apply (f, s_1). **После этого множество фрагментов директив должно быть проверено на непротиворечивость.** Противоречивые фрагменты директив отбрасываются. Недостаток этого метода заключается в трудности определения, какие из фрагментов директив приводят к обнаруженному противоречию. Кроме того, с количественной точки зрения этот метод в чистом виде, по-видимому, не дает никакого выигрыша в сравнении с методом коммуникационных границ. **Метод коммуникационных границ является важной концептуальной основой для развития более близких к практическим целям методов.**

Рассмотрим особенности решения проблемы коммуникационных границы для предписания в исчислении предикатов первого порядка с использованием термов состояний (2.5.4). Трудность ее решения заключается в том, что если в состоянии s_0 нам был известен факт *F2:* At(*B*1, *B,* s_0) — ящик *B*1 находится в *B,* то неизвестно, где находится *B*1 в состоянии s. Этот факт должен устанавливаться в явной форме, т. е. введением дополнительной аксиомы (сравните неудачу директивы *A*1*):*

$$(\forall x)(\forall y)(\forall u)(\forall v)(\forall s)[\mathrm{At}(x, y, s) \wedge \wedge x \neq \mathrm{Robot} \rightarrow \mathrm{At}(x, y, f1(u, v, s))], \quad (2.19)$$

т. е. «положение объекта *x* останется неизменным после того, как отправитель перейдет из *u* в *v*».

Таким образом, нам нужны дополнительные аксиомы о том, что все факты (фрагменты директив), которые не изменяются в результате действия, действительно не изменяются.

Аксиома (2.19) работает и для факта типа *F*5. Однако если бы мы решили задачу *P*2, необходимо было бы передоказать истинность этого факта.

Таким образом, после выполнения каждой директивы необходимо передоказывать истинность всего множества неизменяемых фрагментов директив, что сводит практическую ценность этого метода к нулю для задач, содержащих большие множества фрагментов директив.

Наиболее подходящим в практическом отношении методом решения проблемы коммуникационных границ в декларативных предписаниях является *метод контекстов и контекстных графов* применительно к использованию исчисления предикатов для решения коммуникационных задач в пространстве состояний (п. 2.5.4).

Пусть на множестве состояний S определены предикаты P_i. Множество состояний $S_j \subseteq S$, для которых $P_i[S_j]=P_{ij}$ истинен, называется *контекстом, определяемым предикатом* P_{ij}. Например, факт *F*1 определяет контекст, удовлетворяющий предикату At (Отправитель, *A),* т. е. множество состояний, в которых отправитель находится в *A*. Далее, предикат At *(x, y)* определяет семейство контекстов, т. е. семейство множеств состояний, для которых объект *x* помещен в *y (x, y* — параметры). *Оператор* состоит из *наименования оператора, списка параметров* и двух специальных предикатов — *предиката предусловий K* и *предиката результатов R*. Так, оператор *f*1 из примера в начале параграфа будет представлен следующим образом:

$$\left.\begin{array}{l} f1\,(x,\ y), \\ K\colon \mathrm{At}\,(\text{Отправит.},\ x) \\ R\colon \mathrm{At}\,(\text{Отправит.},\ y) \end{array}\right\} \qquad (2.20)$$

где *f*1 — наименование оператора, *(x, y)* — список параметров,

At (Отправитель, *x)* — предикат предусловий, At (Отправитель, *y)* — предикат результатов.

Когда оператор применяется к контексту, т. е. в нашем примере к множеству состояний, удовлетворяющих At (Отправитель, *x),* он вычеркивает предусловия из списка фактов и добавляет результаты в список фактов, соответствующий состоянию. При этом изменяется контекст, т. е. производится переход в состояния, удовлетворяющие At (Отправитель, *y)*.

Факты, не удовлетворяющие At (Отправитель, *x),* т. е. контекст, определяемый ~At (Отправитель, *x),* не изменяются.

Таким образом, каждый фрагмент директивы, выраженный в форме логического выражения, хранится в системе директив однократно, однако необходимо сохранение истории преобразования контекстов, в которых он добавлялся или вычеркивался. Для этой цели служит *контекстный граф,* вершины которого соответствуют контекстам, а дуги — операторам, преобразующим один контекст в другой. Пусть *I* — начальный контекст. В результате применения последовательности операторов $f_1 \circ f_2 \circ \ldots \circ f_n(I)$ мы получаем последовательность контекстов $C_1, C_2, \ldots, C_n=G$ (*G* — целевой контекст). В общем случае графу поиска коммуникационного решения будет соответствовать контекстный граф. Отношение между контекстным графом и поисковым графом иллюстрируется рис. 2.11, где граф (рис. 2.11, а) отражает поиск пути из *A* в *E,* а граф (рис. 2.11, *б)* — соответствующий ему контекстный граф.

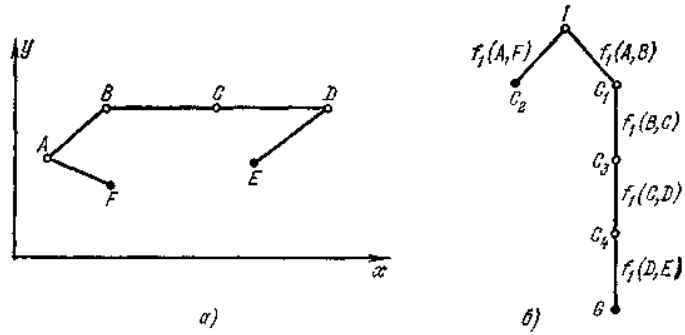


Рис. 2.11. Граф поиска пути из *A* в *E (а)* и соответствующий ему контекстный граф *(б).*

Основной недостаток метода заключается в том, что он не в состоянии прямым путем решить проблему коммуникационных границ для выведенных фактов типа *F*5. Поэтому в принципе все выведенные факты (фрагменты директив) (они определяются чисто синтаксически) должны передоказываться в каждом новом состоянии.

В заключение проконстатируем, что в настоящее время не предложено метода (а возможно, что его и нет), который решал бы в общем виде проблему коммуникационных границ в декларативных предписаниях. Заметим, что эта проблема сравнительно легко, во всяком случае в количественном аспекте, решается в процедуральных предписаниях (п. 2.8.7).

2.8. Процедуральные предписания

2.8.1. Общие характеристики ПОЯ.

Для формального описания коммуникационных систем, сетей, предписаний, директив и др. будем использовать языки ПОЯ. Основными характеристиками, отличающими ПОЯ от обычных языков программирования, являются:

1) Наличие выразительных средств и соответствующих механизмов для ассоциативного поиска и извлечения необходимой в данный момент информации из базы данных.

2) Вызов директив формирования коммуникационных предписаний указанием цели, которая должна быть достигнута, а не по имени.

3) Наличие механизма индикации успеха и неудачи в достижении цели формирования коммуникационных предписаний, позволяющего автоматически вернуться к точке ветвления процесса формирования коммуникационных предписаний, явившейся причиной неудачи, и автоматически исследовать другие альтернативы.

С точки зрения разработчика коммуникационных предписаний, работающего в процедуральном представлении, эти характеристики являются необходимыми для обеспечения целенаправленного механизма поиска решения задачи формирования коммуникационных предписаний.

2.8.2. База данных и механизмы сопоставления по образцу

Опишем общие принципы организации базы данных коммуникационных предписаний, директив и их фрагментов, безотносительные к какому-либо языку.

База данных состоит из предписаний (директив). Каждое предписание обычно содержит *синтаксическую компоненту* и *список свойств,* хранящий произвольные свойства, значениями которых могут быть предписания. Список свойств обычно содержит *семантическую* и *прагматическую информацию.*

Стандартные *семантические свойства предписаний* включают в себя значение предписания, множество равных ему предписаний, множество неравных ему предписаний, правила вычисления (формирования) и упрощения предписаний.

Прагматические свойства обычно выражают информацию, специфическую для данной задачи формирования предписания вообще или для текущего состояния процесса ее решения. Одним из способов задания прагматических свойств являются *рекомендации формирования предписаний*. Рекомендации указывают на то, какие альтернативы предписаний или их директив следует испытать (доказать) и в каком порядке, какие методы следует применить в попытке решить задачу, сохраняют историю попыток испытать те или иные способы действия и т. д.

Весьма важным является способ запоминания и извлечения предписаний из базы данных. Одним из наиболее распространенных способов организации базы данных является ее построение в виде *дискриминационной сети,* впервые предложенной Фейгенбаумом и развитой впоследствии в работах по GPS.

Дискриминационная коммуникационная сеть представляет граф, каждая вершина которого содержит функцию, извлекающую атомарную часть синтаксической компоненты, и является либо конечной вершиной, содержащей директиву, либо промежуточной, содержащей список дуг, исходящей из этой вершины (дуга является парой атом — дочерняя вершина).

Пусть, например, дискриминационная коммуникационная сеть хранит определения типа *M* (MTYPE), (ABC *XY*), (FCT *XY*), (STR *ZY*), (STR *YX*), где MTYPE, ABC, FCT, STR — произвольные атомарные синтаксические формы. Эта сеть приведена на рис. 2.12.

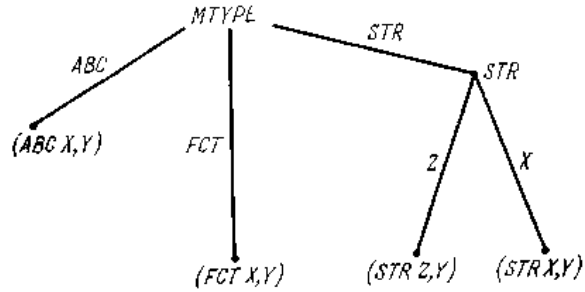


Рис.2.12. Пример дискриминационной коммуникационная сети.

Итак, при входе в сеть извлекается синтаксическая форма корневой вершины, выбирается соответствующая дуга, извлекается синтаксическая форма следующей вершины и т. д., пока либо не будет достигнута конечная вершина, либо не выяснится, что нет подходящей дуги. В последнем случае создается новая конечная вершина, т. е.

входная директива предписания заносится в сеть. При достижении конечной вершины входная директива предписания α сравнивается с синтаксической компонентой директивы предписания в конечной вершине β. Если α и β синтаксически идентичны, то сеть не изменяется. Если предписание α, сопоставилось по форме с предписанием β, то α заносится в список свойств предписания β (если его там не было). Если же предписание α не сопоставляется по форме с предписанием β, то список свойств β просматривается с целью найти там предписания, равные β и сопоставляющиеся по форме с α. В случае успеха α также заносится в список свойств β; при неудаче к α применяются правила упрощения из списка свойств β и проделываются указанные выше манипуляции. Наконец, если и это заканчивается неудачей, то первое синтаксическое различие между α и β запоминается как селектор признаков вновь созданной вершины, от которой проводятся две дуги: одна — к вершине, соответствующей предписанию β, и вторая — к новой вершине, созданной для синтаксической формы входной директивы предписания α. Одновременно создается список свойств этого предписания. Образованная *синтаксическая форма* входной директивы предписания называется *канонической*. Заметим, что здесь и далее мы употребляем понятие «форма» в классическом смысле.

Таким образом, каждое предписание хранится в дискриминационной коммуникационной сети только в одном экземпляре, так что свойства предписания автоматически связываются со всеми эквивалентными предписаниями.

Связанные переменные не могут использоваться как селектор признаков. Например, определения (LAMBDA *(x, y), (x+y)×(y+1))* и (LAMBDA (*u, v),* (l+*v*)×(*v*×*u*)) оцениваются как эквивалентные и преобразуются в одну и ту же каноническую форму (с одинаковым списком свойств).

Как указывалось в п. 2.3.2, процедуральное предписание довольно легко обходит трудности представления отношения равенства в исчислении предикатов первого порядка. Вместо аксиоматизации правил равенства вводятся разбиения предписаний на классы эквивалентных предписаний. На каждом шаге каждое предписания содержит множество логически равных ему на этом шаге предписаний. Эти множества для двух предписаний будут объединены, если будет доказано или высказано в виде утверждения их равенство. Каждое предписание содержит список всех не равных ему предписаний, причем вновь, как только формируется новое предписание о равенстве, эти множества для всех предписаний обновляются соответствующим образом. Следовательно, как только утверждение о равенстве

предписаний вызывает противоречие, это записывается непосредственно.

Рассмотрим вопрос о механизме, с помощью которого в базе данных производится запоминание и извлечение информации. Этим механизмом является *сопоставление по образцу*.

Сопоставление по образцу осуществляется в два шага. На первом шаге производится сопоставление входной директивы предписания α с предписаниями в базе данных β. В общем случае каждое из предписаний представляет собой *форму произвольной степени сложности*. Эта форму будем называть *образцом*. Поскольку предписание α является образцом большей степени общности, чем β, выбор предписания β может быть неоднозначным. На втором шаге по результатам сопоставления образцов производится связывание переменных, входящих в сопоставляемые предписания. В случае, если переменные связываются с некоторыми подопределениями, для присвоения переменным значений может потребоваться вычисление этих подопределенний. Механизм сопоставления является основой следующих операций над базой данных:

1. Композиция предписаний. Определяется процедура comp *(x, y),* создающая (формирующая) предписание, первый элемент (директива) которого есть значение предписания *формы x* и второй элемент — значение предписания *формы y*. Таким образом вычисление comp *(x, y)* для *x*=4, *y=(A (BC))* дает в результате *(A (A (BC)))*.

2. Декомпозиция предписаний. Определяется процедура decomp *(x, y),* вычисляющая значения подопределений исходного предписания *формы x* и *y* соответственно. В результате применения этой директивы к (4(*A (BC))) x* присваивается значение 4, а *y* — значение *(A (BC))*.

3. Извлечение. Директива извлечения находит в базе данных все элементы заданной в директиве формы и выдает в качестве результата произвольную директиву среди множества директив, сопоставляемых с этой формой. Например, применение процедуры Exists (существует)) (At *(x, y))* может извлечь из базы данных факты о местонахождении объектов *формы x* в месте *формы y*. Если ящик *B*1 находится в *A*, то результатом применения процедуры Exists будет At *(B*1, *A)*.

Формы *x* и *y* в режимах композиции и декомпозиции и форма At(*x, y)* в режиме извлечения являются *образцами*.

Следует отметить, что, так как выбор может быть неоднозначным, сопоставление по образцу вносит в процесс решения элемент *недетерминистичности,* так что в случае неудачи разработчику предписаний необходимо вернуться в исходную точку и сделать альтернативный выбор.

2.8.3. Стандартные операторы

Весьма затруднительно описать множество функций, выполняемых ПОЯ. Во-первых, многообразие их велико, во-вторых, особенности выполнения многих из них зависят от структуры конкретного языка, поэтому мы опишем лишь некоторые наиболее общие функции и операторы, вводя необходимые дополнения по мере рассмотрения примеров.

Общим для процедуральных предписаний механизмом работы с директивами является *вызов директив сопоставлением по образцу*. При этом вызов директивы осуществляется не по имени, а указанием предписания, являющегося целью ее применения, а сами директивы воспринимают в качестве аргументов предписания определенной структуры, сопоставляющейся с формой их описания. Например, если мы хотим поставить ящик $B2$ на ящик $B1$, то предписание-цель On($B2$, $B1$) может вызвать из базы данных директиву формы Put(x, y), которая предназначена для того, чтобы поставить предмет x на предмет y. Такой механизм вызова директив полностью совпадает с режимом извлечения общего механизма сопоставления по образцу, однако входное выражение-цель является образцом меньшей степени общности, чем образец в описании директивы. Здесь снова может быть извлечено множество альтернативных директив, что вносит в процесс решения элемент недетерминистичности.

В большинстве ПОЯ используются четыре типа операторов: *цели, директивы, стратегии* и *непосредственные преобразования*. Цели и директивы соответствуют предписаниям и аксиомам в предписании с помощью доказательства предписаний (директив). Непосредственное преобразование предписаний имеет вид правила. Оно воспринимает входную директиву, сопоставляет ее с образцом, который является левой частью правила, и преобразует ее в выходную директиву, подставляя полученные при сопоставлении значения переменных в правую часть правила. Непосредственное преобразование, таким образом, имеет вид $P \rightarrow Q$. Стратегия включает в себя управляющие механизмы, непосредственные преобразования и другие стратегии и управляет процессом решения задачи. Задача ставится как оператор-цель. Система пытается найти непосредственные преобразования и стратегии, помогающие достижению цели. Если стратегия, связанная с целью, приводит к неудаче, то создаются подцели первоначальной цели, т. е. решение идет аналогично процессу редукции. Однако в процедуральном предписании обычно имеется некоторая управляющая программа, анализирующая свойства предписаний и пытающаяся на их

основе и с помощью непосредственных преобразований определить необходимую стратегию.

Предположим, что в управляющую программу введено предписание. Производится два действия:

1. Предписание заносится в базу данных.
2. Предписание обрабатывается с помощью непосредственных преобразований, сопоставляющихся по образцу с предписанием. Результирующие предписаниия также заносятся в базу данных.

Оператор-цель вводится в управляющую программу вместе с рекомендацией. Управляющая программа пытается с помощью сопоставления по образцу и на основе семантической информации и рекомендаций найти полезные преобразования, из которых составляется стратегия, образующая подцели, обрабатываемые аналогичным образом.

Следует отметить, что новые правила преобразования и стратегии становятся достоянием управляющей программы, так что ее арсенал может непрерывно пополняться. Новые правила и стратегии могут вводиться либо пользователем системы, либо накапливаться в результате опыта решения задач.

Изложенная схема представляет собой весьма грубое приближение к действительно имеющей место организации процесса решения коммуникационных задач. Мы намеренно не упоминали существенно рекурсивную структуру всех конструкций процедуральных предписаний и сложные процессы локализации переменных в рекурсивно вложенных контекстах, желая дать максимальную упрощенную картину. Приведенный в 2.8.5 пример послужит дополнительной иллюстрацией процесса решения задачи в процедуральном предписании.

2.8.4. Механизм возврата к точке ветвления

Недетерминистичность выбора альтернатив в различных точках процесса решения коммуникационных задач (извлечение предписаний из базы данных, вызов директив сопоставлением по образцу, альтернативы) приводит к необходимости создания специального механизма, который в случае неудачно выбранной альтернативы возвращает процесс в точку ветвления, восстанавливая при этом все структуры данных и управляющие структуры. Необходимо снабжать коммуникационную систему такой семантической и прагматической информацией, чтобы она могла выбирать наилучшее решение первым. Однако рассчитывать на это было бы слишком оптимистичным, и указанный механизм, называемый *механизмом возврата к точке*

ветвления, является основным средством, позволяющим реализовать целенаправленность выбора стратегий формирования предписаний.

Будем различать *декларативный возврат* и *процедуральный возврат.*

Рассмотрим декларативный возврат. Сущность его заключается в следующем. Если в ходе решения встречается точка ветвления, то запоминается полное описание состояния процесса (т. е. текущие значения всех переменных). Если выбранная альтернатива неудачна, это описание состояния восстанавливается и решение направляется по другому альтернативному пути. Если все альтернативы в данной точке исчерпаны, то сигнал о неудаче распространяется к предшествующей в дереве целей точке ветвления с восстановлением полного описания состояния процесса, соответствующего этой точке.

Процедуральный возврат предложен в связи с разработкой теории недетерминистических алгоритмов. В процедуральном возврате показывается, что всем основным элементам вычислений (выполнениям коммуникационных процессов) — присвоениям, условным ветвлениям, обращениям к выходам из подпрограмм (директивы) и т. д.— можно сопоставить инверсии, т. е. операции, воспринимающие выход данной директивы предписаний как вход и выдающие в качестве результата вход директивы предписаний. Таким образом, в случае неудачного исхода альтернативы можно осуществить локальный пошаговый возврат, применяя инверсные операции, пока не будет достигнута точка ветвления.

Рассмотренные два метода возврата к точке ветвления и являются основой для осуществления механизма возврата в ПОЯ. Каждый метод имеет свои преимущества и недостатки.

Основным преимуществом декларативного возврата является его высокая эффективность: возврат в случае неудачи осуществляется с помощью одного шага. Преимущество процедурального возврата — в отсутствии необходимости запоминать большие объемы ненужной информации при прямом ходе выполнения директив. Один из основанных на процедуральном подходе методов реализации механизма возврата заключается в том, чтобы полностью автоматически запоминать и восстанавливать при возврате *управляющие структуры* и переложить запоминание и восстановление *структур данных* на пользователя, заставив его в явном виде указывать директивы, для которых следует запоминать и восстанавливать локализованные в них переменные.

С одной стороны, этот подход позволяет исключить сохранение ненужной информации и обеспечивает пользователю большую управляемость процессом решения коммуникационной задачи. Однако он требует от программиста точного знания информации, которую

следует запоминать в точке ветвления, т. е. фактически предсказания хода процесса в альтернативных случаях. Кроме того, вероятность ошибок как в сторону недооценки, так и переоценки количества запоминаемой информации резко возрастает. В первом случае это приведет к отказам программы, а во втором — снизит эффективность системы.

Основанные на декларативном подходе механизмы возврата, как правило, используют для запоминания информации специальные *стеки возврата* и *стеки состояний* или *контекстные механизмы*. Эти методы различаются лишь деталями реализации. Основная же идея в обоих случаях заключается в том, что каждой рассматриваемой альтернативе отводится свое поле памяти, или *контекст.* Прямой ход процесса, таким образом, осуществляется с отдельной базой данных, в которой локализованы необходимые для данной - альтернативы предписания. При возврате контекст просто уничтожается (если не принимать во внимание извлечение прагматической информации) (п. 2.8.6).

Следует отметить, что использование механизма возврата является критическим моментом с точки зрения оценки эффективности директивных решателей задач. Дело в том, что в своей первоначальной постановке возврат представляет из себя исчерпывающий поиск в глубину, т. е. в худшем случае полный перебор возможных альтернатив. Это обстоятельство порождает парадоксальную ситуацию: для совершенной стратегии механизм возврата не нужен, так как в каждой точке ветвления она точно знает, что делать. А для плохих стратегий, т. е. стратегий, не обладающих достаточной информацией для выбора альтернатив, механизм возврата становится слепым.

Другая, не менее важная проблема возникает в связи с оценкой действий разработчика предписаний в случае неудачного исследования альтернатив. Чистый механизм возврата не вырабатывает в общем случае информации, которая могла бы повлиять на дальнейший выбор, поскольку в большинстве рассмотренных механизмов все следствия, полученные в результате гипотезы о том, что данная альтернатива полезна, после отбрасывания альтернативы уничтожаются. Это равносильно утверждению о том, что в каждой точке ветвления все альтернативы независимы, т. е. признанию полного перебора.

Указанные принципиальные недостатки механизма возврата усугубляются тем, что в большинстве ПОЯ он используется не только для испытания альтернативных стратегий, но и при каждом вызове директивы или извлечения предписания сопоставлением по образцу. Это может сделать пространство поиска недопустимо большим.

Решение проблемы следует искать там же, где и решение всех основных проблем формирования предписаний: в наложении на процесс поиска эффективных эвристик. Но, как нам уже известно, эти эвристики могут быть получены либо в результате предварительного глобального исследования пространства поиска, либо обобщением опыта в процессе решения задач. В принципе, как указывалось в начале этого параграфа, процедуральное предписание дает богатые возможности для использования семантической и прагматической информации, которая могла бы управлять процессом поиска решения. Однако методы ее извлечения, особенно по результатам исследования неудачных альтернатив, а также, что еще важнее, ее обобщения изучены пока слабо.

2.8.5. Пример

Для лучшего понимания работы некоторых описанных выше механизмов рассмотрим следующий пример.
Докажем силлогизм:

«Тьюринг—человек.
Все люди ошибаются.

Следовательно, Тьюринг ошибается.»

В терминах ПОЯ PLANNER решение может быть получено вычислением выражения (GOAL (ошибается Тьюринг)), где GOAL — упомянутый выше оператор-цель, с помощью следующей процедуры:
<ASSERT (человек Тьюринг)> (посылка в базе данных)
<ASSERT <DEFINE THEOREM1
<CONSEQUENT *(Y)* (ошибается ?*Y*)
(GOAL (человек ?*Y*)>>>>,
где вызовы операторов заключены в скобки < >. ASSERT (утверждение) — оператор-утверждение. Этот оператор, после его применения, заносит свой аргумент в базу данных утверждений. С помощью функции DEFINE THEOREM (определим теорему) мы определяем теорему формы CONSEQUENT (следствие). Это означает, что доказать цель формы (ошибается ?*Y*) можно путем доказательства цели (человек ?*Y*), т. е. первая цель является следствием второй. ?*Y* означает идентификатор, которому может быть присвоено значение в результате сопоставления с образцом, в который ?*Y* входит. Все атомы, не снабженные префиксом ?, являются константами. Работа происходит следующим образом. Если бы нам надо было вычислить

<GOAL (человек Тьюринг)>, то требуемое утверждение было бы найдено непосредственно в базе данных (так как это посылка силлогизма). Однако утверждение (ошибается Тьюринг) в базе данных отсутствует, и мы должны его вывести. Испытываются все теоремы, следствия которых сопоставляются с целью, находится теорема THEOREM 1 и осуществляется переход к доказательству <GOAL (человек ?*Y*)>. В результате сопоставления <GOAL (человек ?*Y*)> и (человек Тьюринг) мы связываем переменную *Y* с константой «Тьюринг». Теорема устанавливает новую цель (человек Тьюринг), и поскольку она находится в базе данных, <GOAL(человек ?*Y*)> достигнута, т.е. доказана <GOAL (ошибается Тьюринг)>. Ниже приведены шаги вычисления:

1) <GOAL (ошибается Тьюринг)>.
2) Теорема 1 активируется.
3) *Y* присваивается значение Тьюринг.
4) <GOAL (человек Тьюринг)>.
5) Результат (ошибается Тьюринг).

Рассмотрим теперь вопрос «кто-нибудь ошибается?» или, в логической форме, EXISTS *X* (ошибается *X)*. В ПОЯ PLANNER это записывается в виде <THPROG (*X*) <GOAL (ошибается ?*X*)>>. В данном случае THPROG, являющийся аналогом функции PROG в языке LISP, действует как квантор существования. Попытка непосредственного вычисления цели приводит к неудаче, так как в базе данных нет утверждения формы (ошибается *?X)*. В поисках доказательства THPROG ищет теорему со следствием этой формы и находит выше определенную теорему. Идентификатор *Y* из теоремы связывается с идентификатором *X* цели. Однако *X* еще не имеет значения, и поэтому *Y* тоже не получает значения. Теорема устанавливает цель (человек ?*Y*). Соответствующий оператор GOAL ищет в базе данных утверждение, сопоставляющееся с этим образцом, и находит утверждение (человек Тьюринг). Поэтому *Y*, а следовательно, и *X* связываются с константой «Тьюринг», и доказательство завершается, выдавая результат (ошибается Тьюринг).

Пусть нам даны дополнительные утверждения <ASSERT (человек Сократ)>, <ASSERT (грек Сократ)>, так что в базе данных находятся три утверждения: (человек Тьюринг), (человек Сократ), (грек Сократ) и теорема

<CONSEQUENT (*Y*) (ошибается ?*Y*)
<GOAL (человек ?*Y*))>.

Мы задаем вопрос «есть ли ошибающийся грек?», представляемый выражением

<THPROG *(X)*

<GOAL (ошибается ?X)>
<GOAL (грек ?X)>>.

Первая цель удовлетворяется рассмотренным выше выводом. Если при поиске в базе данных (человек Тьюринг) встретится раньше, чем (человек Сократ), то цель (человек ?*Y*) теоремы будет достигнута, связывая *Y* и *X* с константой «Тьюринг».

Тогда THPROG устанавливает цель (грек Тьюринг), которая не может быть доказана, так как нет ни соответствующего утверждения в базе данных, ни применимых теорем.

При неудаче работает механизм возврата, который найдет как причину неудачи связывание *Y* с константой «Тьюринг», после чего извлечет (человек Сократ) и продолжит доказательство. Результатом теоремы будет значение (ошибается Сократ), переменные *Y* и *X* связываются с константой «Сократ», и THPROG устанавливает цель (грек Сократ), которая достигается, так как соответствующее утверждение находится в базе данных. В качестве результата THPROG выдает (грек Сократ).

Приведем шаги процесса:

1) Активация THPROG.
2) <GOAL (ошибается ?*X*)) приводит к вызову по сопоставлению образцу, так как в базе данных нет утверждения формы (ошибается ?X).
3) Активация теоремы 1.
4) Сопоставление (ошибается *?Y)* и (ошибается *?X), Y* связывается с *X*.
5) <GOAL (человек ?*Y*)> находит (человек Тьюринг) в базе данных.
6) *Y* принимает значение Тьюринг, следовательно, *X* принимает значение Тьюринг.
7) Результат (человек Тьюринг).
8) Результат теоремы (ошибается Тьюринг).
9) <GOAL (грек Тьюринг)> терпит неудачу, так как в базе данных нет такого утверждения и сопоставляющихся с целью теорем типа CONSEQUENT. Возврат к шагу 5).
6') <GOAL (человек ?*Y*)> находит (человек Сократ) в базе данных.
7') *Y* принимает значение Сократ, следовательно, *X* принимает значение Сократ.
8') Результат (человек Сократ).
9') Результат теоремы (ошибается Сократ).
10) <GOAL (грек Сократ)>.
11) Результат THPROG (грек Сократ).

2.8.6. Контекстный механизм

Как отмечалось в п. 2.8.4, при реализации механизма возврата к точке ветвления необходимо иметь средства запоминания информации при рассмотрении альтернатив. Соответствующий механизм мы назвали *контекстным.*

Контекстный механизм имеет более широкое применение, чем в механизме возврата. Он используется везде, где оказывается необходимым выделение из глобальной базы данных некоторых локальных областей, инициация работы с этими локальными областями без изменения глобальной базы данных и принятие решения по результатам работы о том, следует ли производить изменения в глобальной базе данных, и если следует, то какие.

Такого рода механизмы оказываются полезными для реализации гипотетических планов формирования предписаний, параллельных процессов, описания моделей внешнего мира активных источников действия и т. д.

Контекстный механизм реализуется с помощью разветвленного и ветвящегося в процессе работы стека, который можно представить в виде дерева. Вершины этого дерева соответствуют коммуникационному процессу или состоянию предписания таким образом, что изменение предписаний и их свойств вызывает изменения только в меняющем их процессе и последующих субпроцессах и состояниях (дочерних вершинах). Пример дерева контекстов приведен на рис. 2.13.

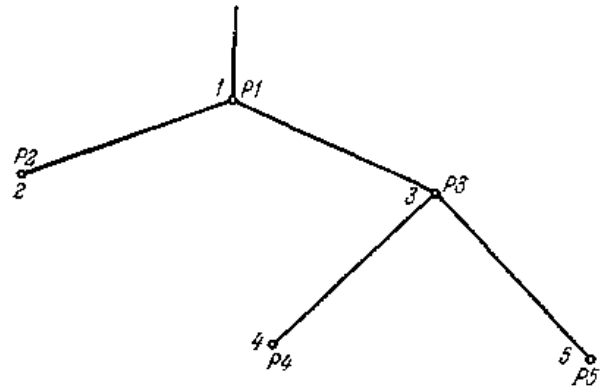


Рис. 2.13. Пример дерева контекстов.

Здесь P1, P2,.., P5 — коммуникационные процессы, а цифры, стоящие рядом с вершинами дерева, обозначают номера контекстов, создаваемых соответствующими коммуникационными процессами. Таким образом, контексты, к которым может иметь доступ коммуникационныый процесс *P4,* будут (4,3,1), *P3* — (3,1) и т. д. Сами контексты представляются в виде списка предписаний, отличающих этот контекст от контекста высшего порядка.

Стандартный набор директив для работы с контекстами включает в себя директивы создания, активации и уничтожения контекста, предписания работы с указанным контекстом или с данным набором образцов в отдельном контексте, а также внесения изменения в глобальную базу данных. Этот набор позволяет выделять локальные области предписаний, хранить историю тех или иных процессов в том или ином контексте, осуществлять гипотетическое планирование формирования предписаний и параллельное построение планов.

Рассмотрим возможную организацию гипотетического плана формирования предписаний. Необходимо создать локальный контекст, сделать требуемые гипотетические посылки и вывести соответствующие заключения. После завершения гипотетического планирования контекст должен быть уничтожен, поскольку истинность посылок в локальном контексте вовсе не предусматривает их истинность в глобальной базе данных. Нам требуется лишь результат типа «если бы посылки были верны, то было бы истинно следующее заключение-предписание». Описанный процесс имел бы следующий вид:

1) Доказать, что $X \rightarrow Y$, где X и Y — произвольные предписания относительно контекста C.
2) Создать новый контекст C'.
3) ASSERT X относительно контекста C'.
4) GOAL (Y) относительно контекста C'.
5) DELETE C' (здесь ««уничтожить контекст C'»).
6) ASSERT $X \rightarrow Y$ относительно контекста C'.

Еще раз отметим, что X истинно только в контексте C' (п. 3 нашего описания).

Параллельные коммукационные процессы также легко реализуются с помощью контекстного механизма. Предположим, что нам надо доказать предписание (построить план) вида $X \vee Y$. Тогда, создав отдельные контексты для X и Y, мы можем запустить параллельное выполнение процессов X и Y, завершив доказательство, когда будет завершен один из процессов X или Y. Естественно, в последовательном вычислительном средстве эти процессы в действительности не будут выполняться параллельно, так что для извлечения выгоды из фиктивной параллелизации доказательства необходимо организовать

взаимодействие процессов *X* и *Y*. Схема организации выглядит следующим образом:

1) Выбрать более легкий по некоторым критериям процесс и доказывать его.

2) Если процесс доказательства сходится быстро в некотором смысле, то продолжать, иначе сохранить его контекст и перейти к доказательству другого процесса, используя, однако, всю полезную информацию, полученную в процессе доказательства первого процесса.

3) Продолжать повторять п. 2 до тех пор, пока не будет найдено доказательство или не будет выработан сигнал о неудаче.

Другим примером параллельных процессов является задача вида $(\exists x)\{P(x) \wedge Q(x)\}$, где *P* и Q — произвольные предписания. Здесь план решения состоит в том, чтобы

1) Найти *x,* удовлетворяющее *P(x)*.

2) Проверить, удовлетворяет ли *x Q(x)*. Если да, то процесс заканчивается, иначе выбрать *x,* удовлетворяющий *Q(x),* используя информацию в контексте для *P(x),* и проверить, удовлетворяет ли *x P (x)* и т. д.

2.8.7. Проблема границ в процедуральных предписаниях

В простейших ситуациях разграничение изменяющихся фрагментов директив от неизменяющихся может быть произведено просто с помощью операторов присваивания или, в крайнем случае, с помощью блочной структуры языка. Эти механизмы охватывают все ситуации, которые разрешает, например, метод контекстов и контекстных графов. Более серьезные меры следует принимать для выведенных фрагментов директив, однако все они довольно легко реализуются для локально-недетерминистичных процедур.

Рассмотрим, например, механизм решения этой задачи в диалекте языка PLANNER — языке POPLER. В этом языке обеспечивается механизм активации определенных процедур при занесении информации в базу данных (ASSERT) и стирании информации в базе данных (ERASE *(*стирать*))*. Когда в базу данных добавляется информация, этот механизм ищет в соответствии с рекомендациями процедуры типа ASSERTING, образцы которых сопоставляются этой информации, и активирует такие процедуры. Аналогичным образом при стирании активируются процедуры типа ERASING.

Рассмотрим действие этого механизма на примере. Пусть база данных содержит три факта:
ASSERT <At HAND (рука) P1) —рука находится в *P1,*
ASSERT <At OBJ2 P1) —объект 2 находится в *P*1,
ASSERT (HOLDING (держит)) OBJ2>—держит объект 2,
утверждающие, что если рука держит объект, то рука и объект находятся в одном месте.
Пусть в базе данных содержатся две процедуры типа ASSERTING:
PROCEDURE ATHAND ASSERTING (At HAND *?X)*
PROCVARS *X Y;*
GOAL <HOLDING *?Y* >;
ASSERT <At *?Y* ?*X*>;
END PROC;
PROCEDURE ATANY (нечто находится где-то) ASSERTING (At*?X ?Y)*
PROCVARS *X Y Z;*
GOAL <At *?X* ?Z);
IF *?Y* =?Z THEN FAIL (сигнал неудачи, возврат к точке ветвления);
ERASE (At ?*X* ?Z);
END PROC;
здесь PROCVARS обозначает описание в процедуре следующих за ней переменных.
Введем в базу данных ASSERT (At HAND *P2)*. Тогда активируются обе процедуры, поскольку они типа ASSERTING, а их образцы сопоставляются с (At HAND *P*2). Предположим, что ATANY испытывается первой. При этом *X* связывается с HAND, а *Y* с *P2*. Оператор GOAL может связать Z либо с *P*1, либо с *P2*. Однако *P2* исключается условным оператором IF, так что стирается только старый факт (At HAND P1). Процедура ATHAND связывает *X* с *P2,* ее GOAL найдет в базе данных (HOLDING OBJ2), связывая *Y* с OBJ2, так что в базу данных попадет утверждение (At OBJ2 P2). Это утверждение вновь активирует процедуру ATANY, которая вновь сотрет старый факт (At OBJ2 PI).
Надо полагать, что наличие в ПОЯ контекстного механизма позволит решать проблему границ и в более сложных мирах.

2.9. Семантические коммуникационные сети

2.9.1. Определения семантических коммуникационных сетей

Развиваемые семантические коммуникационные предписания являются шагом на пути к построению коммуникационных систем и сетей — систем (сетей), базирующихся на знании. Семантические коммуникационные сети являются основой систем зрительного восприятия преметов коммуникаций, понимания естественного языка и непрерывной речи, т. е. систем, осуществляющих связь с внешним миром — одним из главных источников знания коммуникационных систем (другим является сама коммуникационная система). Именно характер развития и разноплановость использования семантических коммуникационных сетей послужили основной причиной разработки многочисленных вариаций предписаний, базирующихся на нескольких сравнительно общих идеях и методах реализации. Мы рассмотрим в настоящем параграфе ряд основных принципов построения и характеристик семантических коммуникационных сетей, ссылаясь там, где это необходимо, на соответствующие конкретные реализации.

Основой всех вариантов семантических коммуникационных сетей является **формализация структур семантического знания в виде направленного графа с помеченными вершинами и дугами, причем вершинам соответствуют некоторые объекты, а дугам — семантические отношения между этими объектами**. Метки, приписываемые вершинам, носят чисто ссылочный характер, представляя собой некоторые мнемонические имена, в частности, слова естественного языка. Заметим, что в этом частном случае слова дают ссылку на словарь, входам которого соответствуют некоторые смысловые эквиваленты, или лексические значения, причем это соответствие может быть неоднозначным в обе стороны. Метки, приписанные дугам, соответствуют элементам множества отношений, заданных на графе, причем этими элементами могут быть как семантические свойства, так и семантические выводы. Описанный выше граф мы будем называть *семантической коммуникационной сетью.*

На семантической коммуникационной сети могут быть определены некоторые подграфы определенной структуры, называемые *предписаниями (директивами)*. Каждый такой подграф представляет собой граф, корнем которого является *предикатная вершина*;

остальные вершины называются *концептуальными*. Следует подчеркнуть, что такое разделение вершин предписания (директивы) можно четко реализовать, лишь рассматривая каждое предписание в изоляции. В структуре семантической коммуникационной сети одна и та же вершина может быть предикатной относительно одного предписания и концептуальной относительно другого. Мы вводим ***понятие предписания (директивы)*** лишь с целью подчеркнуть, что оно является *минимальной единицей информации*, вводимой и хранящейся в семантической коммуникационной сети. На рис. 2.14 приведен пример изображения предписания «Виктор ввел предписание в сеть». Буквами *A, B, C* условно помечены семантические отношения между объектом в предикатной вершине и объектами в концептуальных вершинах.

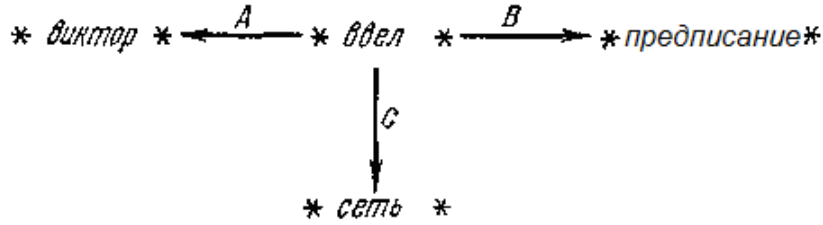


Рис. 2.14. Пример изображения предписаниия в семантической коммуникационной сети.

Заметим, что в семантической коммуникационной сети каждый объект представляется точно одной вершиной. Тем самым мы разрешаем, чтобы от этой вершины (в нее) исходило (входило) несколько дуг, связанных с несколькими различными предписаниями (директивами). На рис. 2.15 изображены абстрактная коммуникационная сеть (C, *R)*, где *C*— множество объектов, *R*— множество отношений (рис. 2.15, *а*), и соответствующая теоретико-графическому предписанию запись в виде списков свойств и троек вида $C_iR_kC_j$, $R_k \in R$, $C_i, C_j \in C$ (рис. 2.15,*б* и 2.15, *в* соответственно), отражающая связь семантических предписаний с процедуральным предписанием и исчислением предикатов соответственно.

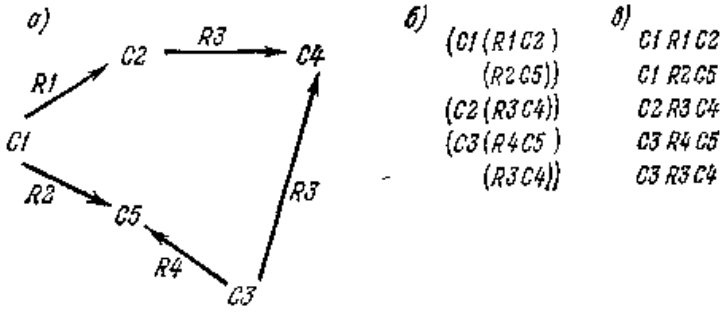


Рис. 2.15. Абстрактная коммуникационная семантическая сеть.

На этом наиболее общая часть определения коммуникационных сетей может быть завершена. Очевидно, что разнообразие конкретных представителей семантических предписаний определяется типами объектов и, в еще большей степени, типами отношений, определенными в каждом из предписаний.

2.9.2. Типы объектов

В семантических коммуникационных сетях используются три основных типа объектов: *понятия, события* и *свойства*.

Понятия являются константами или параметрами сущностей, описываемых коммуникационной семантической сетью, и обычно указывают предметы или абстракции.

События представляют собой действия, которые могут произойти в коммуникационных сетях. Если мы определим *ситуацию* как описание части коммуникационной сети в определенный момент времени, то можно сказать, что все, что изменяет данную коммуникационную сеть, является событием. Одним из методов представления событий является задание *глубинно-падежных семантических отношений*, которые указывают характеристики и действующих коммуникационных сетей данного события. Не вдаваясь в подробности грамматики глубинных падежей, отметим, что отношения *A, B* и С, приписанные дугам предписания (рис. 2.14), эквивалентны соответственно агентивному, объективному и локативному падежам Филмора. Другой метод описания событий состоит в указании изменений, которые событие производит, будучи применено к структуре коммуниеационной сети, отражающей данную ситуацию. Результатом события является также некоторая ситуация, которую мы можем

определить как образец в некотором предписании, описывающем последовательность действий, приводящем к этой ситуации.

Свойства используются для уточнения или модификации понятий, событий или других свойств. В случае понятий свойства могут быть особенностями, чертами или характеристиками, присущими или приписанными понятию. В случае событий свойства описывают некоторые общие, универсальные, постоянные характеристики, например, место, время, длительность и т. д.

Формально свойство является бинарным отношением, отображающим область своего определения, т.е. вершины, к которым свойство применяется, в область значений, т. е. значения, которое свойство может принимать. Рис. 2.16 иллюстрирует, как применение свойства «структура» к понятию «сеть» и свойства «время» к событию «ввел» со значениями «дискриминационная» и «вчера» соответственнодает высказывание «Виктор ввел вчера высказывание в дискриминационную сеть».

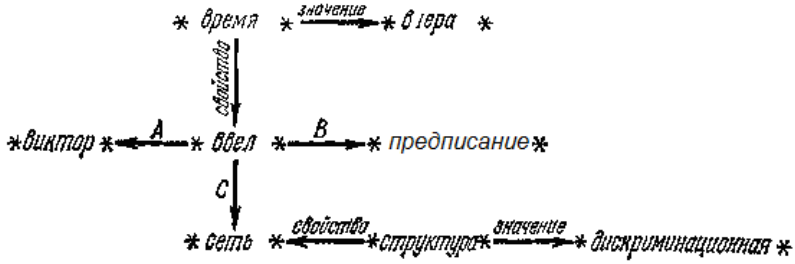


Рис. 2.16. Представление свойств в семантической сети определений.

Здесь дуга, помеченная «свойство», указывает на аргумент, а дуга, помеченная «значение»,— на значение свойства. В некоторых случаях возможно расширение понятия свойства от бинарного к тернарному и многоместным отношениям. При этом дополнительные характеристики свойства связываются с вершиной свойства дугами, помеченными «относительно». Рис. 2.17 показывает, как введение аргумента «относительно момента T_0» уточняет значение «вчера» свойства «время».

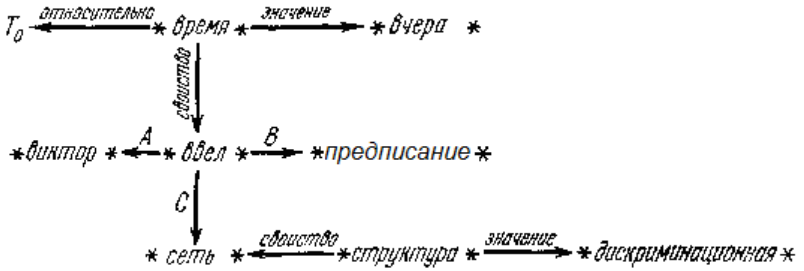


Рис. 2.17. Представление дополнительных характеристик свойств.

Вершины, входящие в коммуникационную семантическую сеть, независимо от их типа могут быть разделены на два класса. Один из них включает в себя понятия, события и свойства общего характера, описывающие в совокупности законы, действующие в мире, представленном семантической коммуникационной сетью. Другой класс — частные случаи общих объектов, описывающие конкретные проявления указанных выше законов, или просто некоторые факты. Вершины первого класса мы будем называть *общими,* вершины второго класса — *фактуальными*. Пример такого рода вершин приведен на рис. 2.18, причем на рис. 2.18, *а* утверждается что ПОЛ — свойство ЖИВОТНОГО и принимает возможное значение МУЖСКОЙ (общие вершины условно обозначены прописными буквами), а на рис. 2.18, *б* указывается, что Виктор — мужского пола (фактуальные вершины условно ограничиваются с двух сторон звездочками).

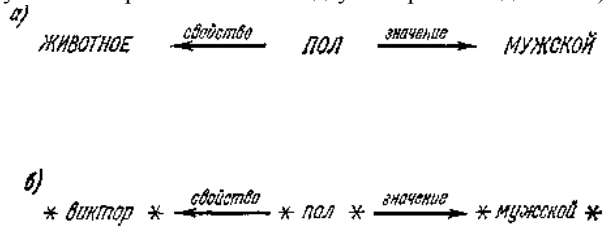


Рис. 2.18. Общие и фактуальные вершины.

Кроме рассмотренных выше типов вершин — понятий, событий, свойств, в целях повышения эффективности вывода в семантических коммуникационных сетях вводятся так называемые *процедуральные* вершины. Одним из примеров фундаментального характера такого рода вершин являются скелеты (п. 2.9.4). Существуют и другие процедуральные вершины более специализированного характера. Чаще

всего они располагаются в районе границы между общими и фактуальными вершинами, т. е. на периферии семантической коммуникационной сети.

2.9.3. Типы отношений

Все многообразие семантических отношений, используемых в семантических предписаниях, может быть условно разделено на четыре класса: *лингвистические, логические, теоретико-множественные* и *квантификационные*.

Лингвистические отношения включают в себя прежде всего глубинно-падежные отношения, уже упомянутые в п. 2.9.2 в связи с представлением событий. В коммуникационных системах этот тип отношений играет ключевую роль в общей организации семантической коммуникационной сети, определяя как взаимосвязи отдельных объектов, так и структуру вывода в коммуникационной сети.

Другими двумя типами лингвистических отношений являются *характеризации глаголов* и *атрибутивные отношения*. В нашем изложении они описываются как вершины-свойства, однако в ряде конкретных систем эти отношения выделены введением отдельных дуг для каждого из них. К числу глагольных характеризаций относятся время, наклонение, вид, род, число, залог используемого глагола. К числу атрибутивных отношений относятся модификация, цвет, размер, форма, отношение собственности («притяжательность») и т. д. На рис. 2.19, *а* приведено подробное представление глагола «ввел» с использованием глагольных характеризаций, а на рис. 2.19, *б* представление того же глагола с использованием вершин типа свойств.

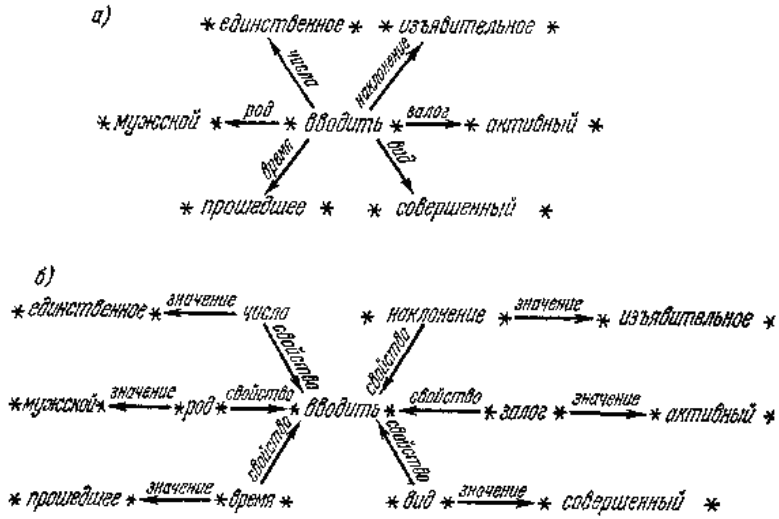


Рис. 2.19. Представление характеризаций глаголов.

На рис. 2.20 иллюстрируется использование атрибутивных отношений для представления словосочетания «маленькая древовидная дискриминационная сеть Виктора». Из приведенных примеров ясно, что глагольные характеризаций и атрибутивные отношения эквивалентны свойствам событий и понятий соответственно.

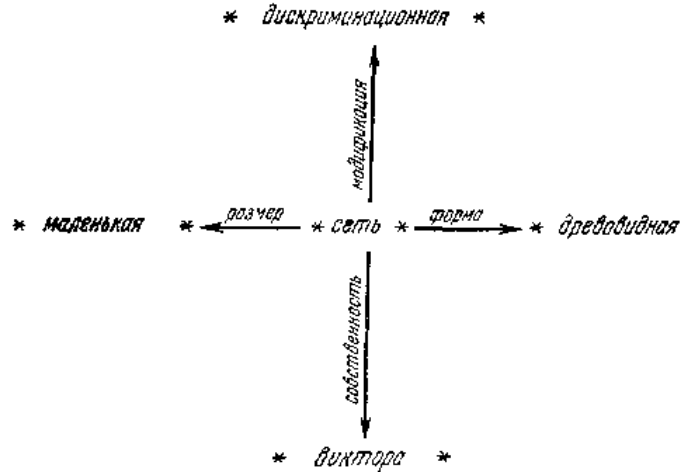


Рис. 2.20. Представление атрибутивных отношений.

К логическим отношениям относятся операции исчисления предписаний: дизъюнкция, конъюнкция, отрицание, импликация. Практически все структуры семантических коммуникационных сетей включают неявное представление конъюнкции, поскольку все занесенные в коммуникационную сеть предписания считаются истинными. Однако требование полноты логических отношений обуславливает необходимость добавления к конъюнкции как минимум отрицания. Отметим, что в большинстве семантических коммуникационных сетях определяется отрицание лишь элементарных предписаний - директив, введенных в п. 2.9.1. В этом случае отрицание не образует логически полную систему с конъюнкцией. Если допустить явное представление предписаний всех операций исчисления предписаний в виде дуг семантической сети, может быть нарушено соглашение об истинности всех предписаний в коммуникационной сети: в семантической коммуникационной сети истинными станут только те предписания, которые не являются конституентами составных предписаний. Возникает вопрос о том, как указать истинность конституенты в составном предписании. На самом деле этой проблемы нет, если конституенты в составном предписании связаны конъюнкцией, дизъюнкцией или импликацией, так как указание истинности одной из конституент сводит такое предписание к конъюнкции, т. е. к неявному указанию истинности любой конституенты. Например,

$$p \wedge (p \vee q \vee r) = p,$$

$$p \wedge (p \rightarrow q) = p \wedge q.$$

Однако иногда возникает необходимость в независимом от составного предписания указании истинности конституенты (например, при выражении в предписании причин, намерений, отношения к чему-либо и т. д.). Эти случаи охватываются применением конъюнкции с константой «истина» или «ложь». На рис. 2.21, *а, б,* соответственно изображено представление двух составных предписаний «Робот считает, что Виктор ввел высказывание в сеть, и он ввел его» и «Робот считает, что Виктор ввел высказывание в сеть, а он не ввел его».

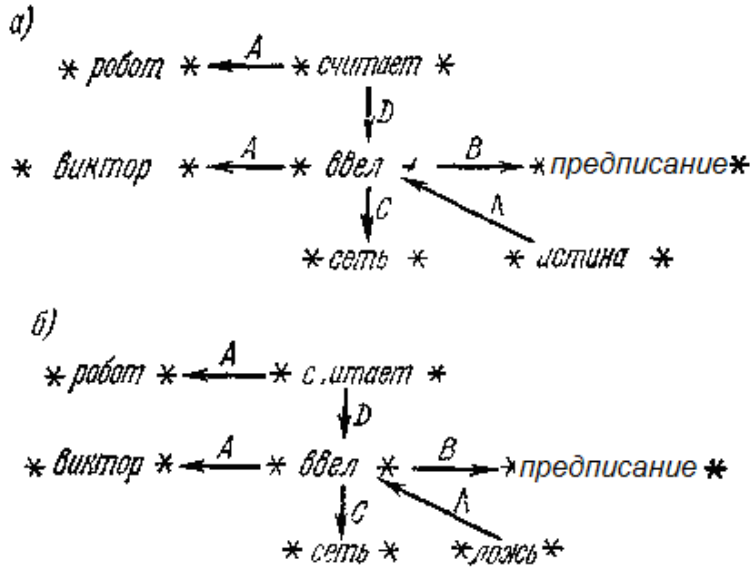


Рис. 2.21. Представление составных модальных предписаний.

Теоретико-множественные отношения включают в себя такие отношения, как *подмножество* (SUB), *супермножество* (SUP), отношения части и целого, *элемент множества,* или пример (E), и другие.

Большинство отношений этого класса является представителями класса *транзитивных отношений,* т. е. отношений, связывающих вложенные друг в друга понятия от более общих к более частным (SUB) или наоборот (SUP), причем все свойства SUP-понятия автоматически становятся свойствами SUB-понятия, а свойства SUB-понятия представляют собой ограничения, накладываемые на SUP-понятия с целью определения SUB-понятия. Отношение E является частным случаем SUB, связывающим общую вершину с фактуальной. Отношения части и целого, подчиненности, сходства, близости и т. д. основаны на отношениях SUP и SUB, однако могут иметь более сложную структуру предписания и (или) вывода. Так, например, два понятия являются сходными, если они имеют общее SUP-понятие, а большинство одноименных свойств этих понятий имеет одинаковые значения. Таким образом, сходство является *размытым аналогом* эквивалентности, которое в свою очередь также представляет собой теоретико-множественное отношение.

Отношения SUP и SUB могут быть определены и для событий. В этом случае структуры событий, образуемые этими отношениями, аналогичны множествам соответствующих событий и связанных с ними частных случаев событий. При этом свойства SUB-событий могут иметь в качестве значений наречия («быстро», «бесплатно») или другие модификаторы, которым сопоставляются числовые эквиваленты в понятиях теории лингвистических переменных.
Лингвистическая переменная — в теории нечётких множеств, переменная, которая может принимать значения фраз из естественного или искусственного языка. Например, лингвистическая переменная «скорость» может иметь значения «высокая», «средняя», «очень низкая» и т. д. Фразы, значение которых принимает переменная, в свою очередь являются именами нечетких переменных и описываются нечетким множеством.
Математическое определение.

Лингвистической переменной называется пятерка $\{x, T(x), X, G, M\}$, где x — имя переменной; $T(x)$ — некоторое множество значений лингвистической переменной x, каждое из которых является нечеткой переменной на множестве X; G есть синтаксическое правило для образования имен новых значений x; M есть семантическая процедура, позволяющая преобразовать новое имя, образованное процедурой G, в нечеткую переменную (задать вид функции принадлежности), ассоциирует имя с его значением, понятием. $T(x)$ также называют базовым терм-множеством, поскольку оно задает минимальное количество значений, на основании которых при помощи правил G и M можно сформировать остальные допустимые значения лингвистической переменной. Множество $T(x)$ и новые образованные при помощи G и M значения лингвистической переменной образуют расширенное терм-множество.

Пример: нечёткий возраст
Рассмотрим лингвистическую переменную, описывающую возраст человека, тогда:

- x: «возраст»;
- X: множество целых чисел из интервала [0, 120];
- $T(x)$: значения «молодой», «зрелый», «старый». множество $T(x)$ - множество нечетких переменных, для каждого значения: «молодой», «зрелый», «старый», необходимо задать функцию

принадлежности, которая задает информацию о том, людей какого возраста считать молодыми, зрелыми, старыми;

- *G*: «очень», «не очень». Такие добавки позволяют образовывать новые значения: «очень молодой», «не очень старый» и пр.
- *M*: математическое правило, определяющее вид функции принадлежности для каждого значения образованного при помощи правила *G*.)

Квантификационные отношения включают в себя *логические кванторы* (общности, существования), *нелогические кванторы* (много, несколько), а также просто *числовые характеристики* объектов. Приведем ряд причин, по которым необходимо ввести в семантическую коммуникационную сеть логические кванторы.

1. Многие предписания содержат кванторы общности или существования.
2. Логические кванторы требуются для декларативного предписания общих знаний (законов мира).
3. Определение сложных понятий и событий требует логической квантификации (например, «ходить — это значит в любой момент времени касаться хотя бы одной ногой земли»).
4. Кванторы требуются для определяющих описаний множеств.

Несмотря на важность введения логической квантификации в семантическую коммуникационную сеть, достижения в этой области ограничивались приписыванием одного или двух кванторов к предикатам.

Одним из способов введения логических кванторов является представление семантической коммуникационной сети в языке, подобном исчислению предикатов первого порядка (п.2.5). Рассмотрим вариант такого формализма предписания на примере высказывания «Каждая собака преследует некоего кота». Это высказывание может быть преобразовано из исходного предписания

$$(\forall x)\,(\text{собака}\ (x) \rightarrow (\exists y)\,(\text{кот}\ (y) \wedge \text{преследует}\ (x, y))) \qquad (2.21)$$

в вид, близкий к бесквантворной нормальной форме

$$\text{собака}\ (x) \rightarrow (\text{кот}\ (f(x)) \wedge \text{преследует}\ (x, f(x))). \qquad (2.22)$$

Соответствующее предписания в семантической коммуникационной сети приведено на рис. 2.22, *а*.

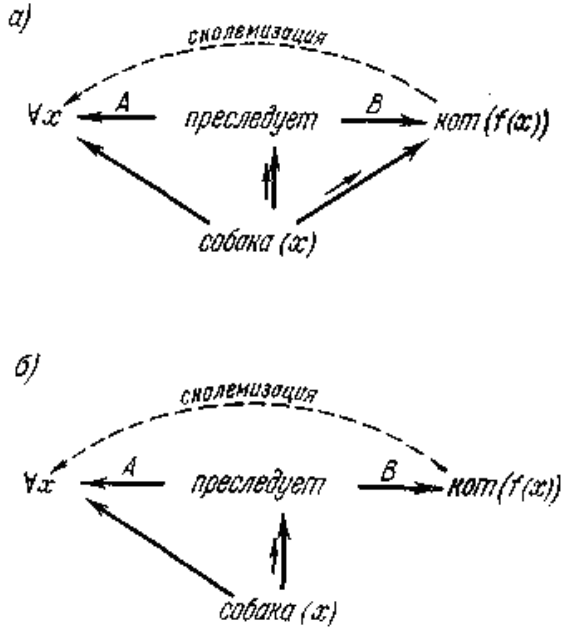


Рис. 2.22. Представление предписания «Каждая собака преследует некоего кота».

Таким образом, мы неявно указываем квантор существования, вводим новую вершину, соответствующую квантору общности, и проводим к ней две дуги, одна из которых указывает связывание переменной в предикате «собака» с введенным квантором, а вторая, соответствующая сколемизации, проводится от вершины, связанной неявно квантором существования, поскольку в (2.21) он стоит за квантором общности.В общем случае сколемизирующие дуги проводятся от всех вершин, связанных неявно кванторами существования, ко всем вершинам тех кванторов общности, которые стоят перед кванторами существования. Заметим, что если мы предположим, что *(∃y) (кот(y)),* то (2.22) приводится к виду

кот *(f(x))* ∧ (собака *(x)* → преследует *(x, f(x))),* (2.23)

и этому высказыванию соответствует фрагмент семантической сети (рис. 2.22, *б*).

Другим способом введения логических кванторов является *разбиение семантической коммуникационной сети на пространства*. Это разбиение осуществляется таким образом, что каждая вершина и дуга семантической коммуникационной сети принадлежит точно одному

пространству. Все вершины и дуги, лежащие в различных пространствах, различимы между собой, а дуги, связывающие различные пространства, принадлежат тому из них, в котором они начинаются. Разбиение семантической коммуникационной сети на пространства задает структуру связи этих пространств в виде направленного графа *(S, V)*, пример которого показан на рис. 2.23.

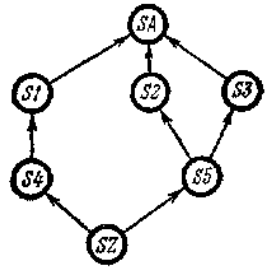


Рис. 2.23. Представление структуры связи пространств в виде графа

Вершины этого графа соответствуют пространствам семантической коммуникационной сети, а дуги задают отношение «видимости» на этом графе. Из любой вершины *Si* видны те и только те вершины, которые лежат на всех возможных путях, ведущих из *Si* в *SA*. Например, из *S*5 графа, (рис. 2.23) видны вершины *S2*, *S3* и *SA*. Очевидно, что *SA* видима из всех пространств сети, а из *SZ* видна вся семантическая коммуникационная сеть. Рассмотрим представление логической квантификации с помощью разбиений, используя следующую последовательность примеров:

а) Собака Шарик преследует кота Ваську.
б) Каждая собака преследует некоего кота.
в) Все собаки преследуют кота Ваську.
г) Все собаки преследуют всех котов.

Приведем формальную запись этих высказываний:

а) ($\exists$ Васька $\in$ КОТЫ) ($\exists$ Шарик $\in$ СОБАКИ)
(преследует (Шарик, Васька)).

б) ($\forall x$) (собака *(x)* $\rightarrow$ ($\exists y$) (кот *(y)* $\wedge$ преследует *(x, y)))*.

в) ($\forall x$) (собака (*x*) $\rightarrow$ ($\exists$ Васька $\in$ КОТЫ) (преследует *(x,* Васька))). (2.24)

г) ($\forall x$) *(* $\forall y$*)* (собака *(x)* $\rightarrow$
$\rightarrow$(кот(*y*) $\wedge$ преследует *(x, y)))*.

Соответствующие семантические предписания показаны на рис. 2.24, *а—г*.

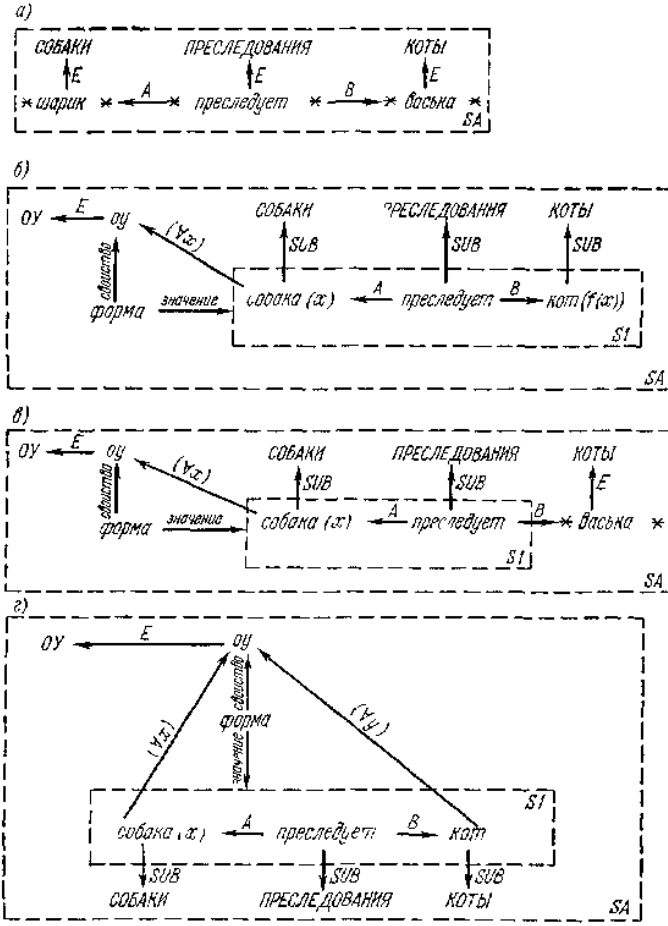


Рис. 2.24. К разбиению семантической коммуникационной сети на пространства.

Предписание (2.24, *а)* размещается в одном пространстве самого верхнего уровня *(SA),* поскольку в данном случае речь идет о конкретном событии, происшедшем с конкретными индивидуумами. Поскольку последние представлены фактуальными вершинами, то мы дали для иллюстрации их SUP-вершины с указанием отношения E.

Так как предписание (2.24, *б)* описывает некоторое множество событий и действующих лиц, то ему как *форме* соответствует некоторое *общее утверждение* оу ∈ ОУ, где ОУ — множество всех общих утверждений,

включающих кванторы общности. Мы представляем это введением формы как свойства оу, имеющего в данном случае в качестве значения выражение, лежащее в области действия квантора общности, т. е. выражения в пространстве *S*1. Таким образом, пространство *S*1 является областью действия квантора $\forall x$, а сколемизация переменных, связанных квантором существования, представляется помещением предикатов от этих переменных в пространство *S*1 (в нашем примере — кот *(f(x))*). Формализм представления квантора общности завершается связыванием оу дугой, помеченной $\forall x$, к квантифицируемым предикатам, причем эта дуга идет из *S*1 в *SA*, *S*1 *V SA* (читается «*SA* видимо из *S*1»). Заметим, что на рис. 2 24, *б* связь SUP-объектов с объектами в *S*1 осуществлена с помощью SUB (а не E), поскольку форма описывает некоторые подмножества событий и понятий, а не частный пример. Отметим, что рис. 2 24, *б* иллюстрирует тот факт, что константа, находящаяся под знаком $\exists$, выносится из пространства *S*1 (естественно, что она не подлежит сколемизации), а рис. 2.24, *г* показывает, что последовательность кванторов общности представляется в одном пространстве с соответствующим количеством дуг, исходящих из оу к квантифицируемым предикатам. На рис. 2.25 показан пример представления сложной последовательности логических кванторов в виде вложенных пространств так, что *S*4 *V* S1, *S*1 *V SA* (отношение видимости, конечно, транзитивно).

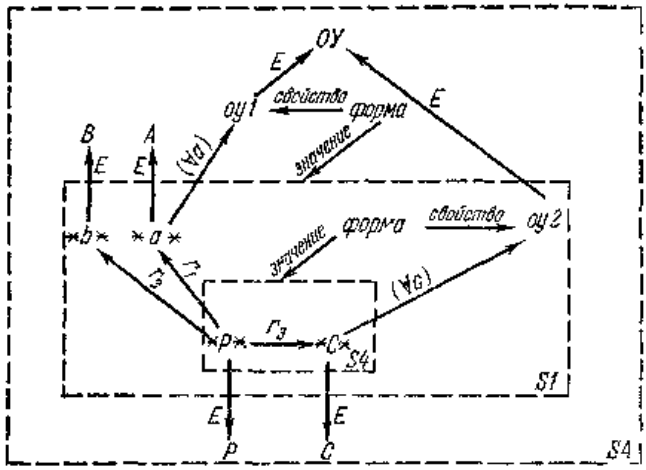


Рис. 2.25. Представление последовательности логических кванторов, $(\forall a \in A)(\exists b \in B)(\forall c \in C)(p(a,b,c))$.

Изложенные методы указания областей действия кванторов могут быть использованы для представления λ-выражений и модальностей. Мы не будем останавливаться на изучении этих возможностей, однако с целью иллюстрации приведем на рис. 2.26 предписание с помощью разбиений модального высказывания (рис. 2.21).

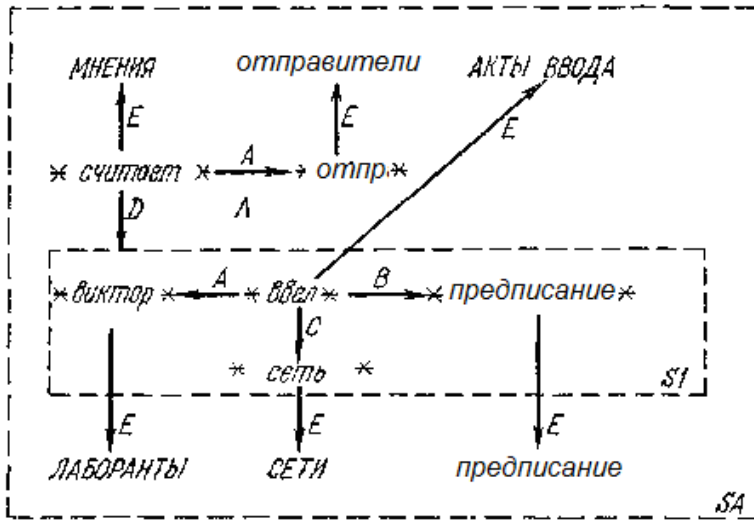


Рис. 2.26. Представление модальностей с помощью разбиений.

Представление *нелогических кванторов* остается проблемой в семантических сетях так же, как и в других формализмах. Аппарат для работы с нелогическими кванторами, как и с другими размытыми понятиями, разрабатывается в рамках теории лингвистических переменных, однако этот подход еще не связан с семантическими коммуникационными предписаниями.

Наконец, *числовые характеристики* объектов представляют собой просто количество объектов данного типа и могут быть выражены, например, с помощью вершин свойств.

2.9.4. Скелеты и сценарии

В п. 2.9.2 мы упомянули два способа описания событий — с помощью глубинно-падежных отношений и с помощью директив. Эти способы отражают соответственно семантическое и процедуральное описание весьма общего класса структур данных для представления хорошо

известных, стереотипизированных ситуаций, которые Минский называет «скелетами» (frame), а Шенк — «сценариями». По существу ***скелеты (сценарии)*** **— это совокупность условий применения (предпосылок), моделей действия и выводов для достижения определенной цели, описывающей стереотипизированную ситуацию**. Сценарий содержит целый ряд предпосылок, которые, как предполагается, достаточно хорошо описывают стандартный набор условий его активизации. Так, например, если мы рассматриваем сценарий «пойти на день рождения», то такие *условия,* как «одеть костюм», «купить подарок», будут считаться *стандартными,* в то время как «одеть каску» (во избежание обвала потолка) будут считаться *исключительными*. Чем меньше предпосылок будет иметь сценарий, тем более общим он является, и наоборот. Число действительных характеристик ситуации, в которой активируется сценарий, может не соответствовать числу его предпосылок. В случае недостатка предпосылок оказывается необходимым введение в систему специализированных директив, соответствующих избыточным (и поэтому исключительным) характеристикам ситуации. Однако в этом случае сценарий может быть использован для указания того, какого рода специализированные директивы необходимы. Заметим, что в сценарии создаются ветвления, соответствующие исключительным для него условиям.

В случае избытка предпосылок сценарий не может функционировать целиком, однако при этом могут образовываться неполные выводы или (и) указания того, каких характеристик ситуации не хватает для полного использования сценария.

В процессе выполнения сценария могут возникнуть новые ситуации, требующие вызова других сценариев, так что могут **образовываться последовательные, вложенные, рекурсивные или параллельные *композиции сценариев***. В рамках настоящего параграфа мы ограничимся кратким описанием **семантического предписания сценариев** (именуемого впредь ***сценарием)*** и **процедурального предписания сценариев** (именуемого впредь ***скелетом)*** применительно к их использованию в семантических сетях формирования предписаний.

Назовем *сценарием* совокупность событий и свойств, связанных отношениями с помощью соответствующих дуг и понятий, заполняющих глубинные падежи, или с помощью отношений времени и (или) причины-следствия. С операциональной точки зрения сценарий следует рассматривать как некоторый *сложный образец предписания*, который при сопоставлении его с некоторой структурой позволяет системе делать определенные *выводы* и *предсказания.* В качестве

простейшего сценария можно было бы привести высказывание на рис. 2.14 (при условии замены фактуальных вершин на общие), которое могло бы сопоставляться со структурой, где Виктор вводил бы вполне определенное высказывание во вполне определенную сеть. Этот сценарий, однако, тривиален, поскольку в нем не содержится возможности делать какие-либо выводы или предписания.

Между сценариями могут быть установлены SUP- и *SUB-отношения.* Этого можно достигнуть, строя SUP- или SUB-объекты объектов, входящих в определение сценария. Сценарии могут быть организованы в структуры, связанные *определяющим отношением* (отношением DEF). Это отношение задает организацию сценария как композицию других, более простых сценариев, причем, как указывалось ранее, эта композиция может быть последовательной, вложенной, рекурсивной или параллельной. В частности, может быть реализована структура сценариев, аналогичная пропозициональному графу. Сценарии задаются совокупностью вершин, отражающей структуру событий той или иной степени сложности и общности, а также действующих лиц и характеристик событий, связанных с событиями глубинно-падежными отношениями.

В отличие от сценариев, *скелеты* представляются в семантической сети специальными вершинами. Прежде всего рассмотрим отличия вершин скелетов от обычных вершин семантической сети.

1. Прежде чем переходить от вершины скелета к связанным с нею вершинами, следует осуществить проверку применимости скелета. Если он неприменим, то необходимо обратиться к списку альтернативных скелетов, описывающих сходный класс ситуаций. В случае применимости скелета он может определить, по каким дугам идти дальше и в каком порядке.
2. При достижении некоторого скелета он может взять на себя управление и уже не возвращать его вызвавшему скелету.
3. С вершиной скелета могут быть связаны дугами некоторые *субскелетные вершины.* Здесь под *субскелетными вершинами* мы подразумеваем вершины, которые не входят в семантическую сеть, могут быть выбраны только при активации соответствующих скелетов и которым соотносятся некоторые процедуры или функции. Последние могут проверять условия и выполнять действия, используя для этого другие субскелетные вершины данного скелета или скелета, который вызвал данный скелет. В частности, через указанные процедуры (функции) могут быть активированы другие скелеты. Схема скелета приведена на рис. 2.27.

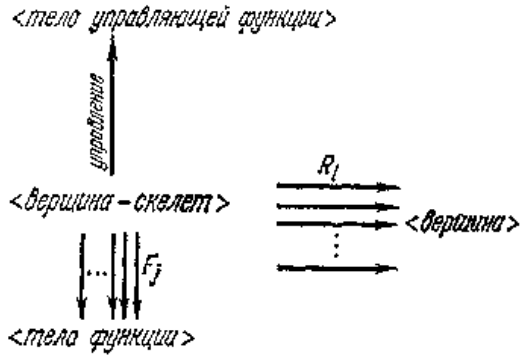


Рис. 2.27. Схема скелета.

Здесь R_i — обычные отношения, заданные в семантической сети, «управление» — дуга, ведущая к вычислению тела управляющей функции (проверка применимости скелета, указание последовательности перехода к субскелетным вершинам, возврат управления вызывающему скелету), F_j — метки дуг, направленных к субскелетным вершинам. Если дуга F_j выбрана управляющей функцией, то вычисляется тело процедуры или функции F_j. Заметим, что в случае невыполнения некоторых из условий F_j может сигнализировать о недостатке информации о характеристиках ситуаций.

2.9.5. Процессы понимания и вывода в семантических предписаниях

Ранее мы описали различные элементы представления семантического знания в семантических коммуникационных сетях. Сейчас рассмотрим ряд вопросов, связанных с обработкой информации в коммуникационной сети. К числу этих вопросов относятся:

— как извлекать необходимую информацию?

— как вводить новую информацию в коммуникационную сеть?

— каким образом выделять в сложных и разветвленных коммуникационных сетях участки, имеющие отношение к интересующей систему области?

— как осуществлять процессы вывода ответов на вопросы, заданные семантической сети?

Оказывается, что ответы на эти вопросы тесно связаны между собой и основаны на общей трактовке процессов понимания. **Мы определяем *понимание* как интерпретацию новых фактов относительно**

текущего контекста. Более формально, пусть $C(T_1, T_2, ..., T_i)$ — ситуация или *контекст,* установленный в результате понимания последовательности предписаний $T_1, T_2, ..., T_i$. Пусть далее $I(T_{i+1}, C(T_1, T_2, ..., T_j))$ — *интерпретация входного предписания* T_{i+1} в контексте, установленном $T_1, T_2, ..., T_j$. Тогда эффективный алгоритм вычисления $I(T, C)$ называется *пониманием.* С точки зрения нашего предписания все поставленные выше вопросы обретают общую концептуальную основу. Нам необходимо применить общие методы эффективного поиска участка семантической коммуникационной сети, имеющего отношение к рассматриваемой области (или к заданным вопросам, или вновь вводимому предписанию), и уже только потом осуществить некоторые специализированные манипуляции в зависимости от конкретного характера задачи.

Становление семантических предписаний началось с настоящей работы. В них в качестве объектов определяются понятия и свойства, а основным видом отношений являются транзитивные теоретико-множественные и логические отношения. Классические коммуникационные сети носят, во-первых, статический и декларативный характер и, во-вторых, обладают значительной однородностью. Эта однородность позволяет описать процессы выделения контекста в таких коммуникационной сетях с помощью одного базового метода *поиска пересечений,* выделяющего участок коммуникационной сети, связанный с входными понятиями (директивами) $X_1, X_2, ... X_n$. Суть метода состоит в том, чтобы, начиная от вершин $X_1, X_2, . . ., X_n$ и двигаясь по дугам, помеченным транзитивными отношениями, искать такие понятия (возможно, ближайшие в некотором смысле), которые находятся на пересечении построенных путей. Тогда множество пройденных вершин и соединяющих их дуг образуют *контекст,* связывающий исходные понятия (директивы) $X_1, X_2, . . ., X_n$. Транзитивность отношений позволяет строить достаточно длинные пути. Проиллюстрируем сказанное для случая двух исходных понятий X, Y и транзитивного отношения R на примере ряда вопросов и ответов относительно связи X и Y (в качестве R берется отношение SUB). Для того чтобы определить, находится ли X в отношении R с Y, ищутся все возможные пересечения X и Y. При этом может возникнуть ряд вариантов.

1. Некоторый путь связывает X с Y, т. е. XRY
2. Некоторый путь связывает Y с X, т. е. $XR^{-1}Y$, R^{-1} — обратное и потому также транзитивное отношение.
3. Пути, начинающиеся в X и Y, не пересекаются.
4. Существует пересечение путей, начинающихся в X и Y, т. е.

$$(\exists\ w)(XRw \wedge YRw). \qquad (2.25)$$

В случае 1 мы даем утвердительный ответ на вопрос, является ли *X* SUB-объектом для *Y*.

Пример. Является ли Шарик собакой?

Да.

В случае 2 утвердительный ответ можно дать лишь в некоторых частных случаях.

Пример. Является ли собака Шариком?

Возможно, что да.

В случае 3 ответ является отрицательным, так как *X* и *Y* не имеют общих SUP-объектов.

Пример. Является ли трамвай желанием?

Нет.

Рассмотрим подробнее случай наличия пересечения (или пересечений) входных объектов. Здесь следует выделить три случая:

а) *X* и *Y* взаимно исключают друг друга.

Ответ отрицательный.

Пример. Является ли собака кошкой? (пересечение—млекопитающее).

Нет.

б) *X* и *Y* содержат идентичные свойства с различными, по крайней мере для одного свойства, значениями.

Ответ отрицательный.

Пример. Является ли индеец лордом?

Нет.

в) Свойства *X* и *Y*, а также значения этих свойств не различаются. Здесь возможны два подслучая. Если *X* и *Y* обладают в сети исчерпывающим списком свойств, то следует вывод об идентичности *X* и *Y*, т. е. ответ утвердительный. Если же список свойств *X* и *Y* не является исчерпывающим, то точный ответ неизвестен и следует применять приблизительные методы, в частности, *функциональный, негативный* и *индуктивный*. Мы не будем останавливаться на этих методах, отсылая читателя к соответствующим работам [Коллинз, Квиллиан, 1972; Карбонелл, Коллинз, 1973]. Отметим лишь, что *индуктивный вывод* в семантических коммуникационных сетях играет важную роль при вложении новой информации в сеть. Действительно, если в сеть «часто» (в некотором смысле) вводятся понятия с одинаковым значением некоторого свойства, то есть смысл приписать это свойство общему SUP-понятию вводимых понятий. Например, если воробьи, грачи, ласточки и т. д. летают, то это свойство следует приписать понятию «птица». Здесь реализуется *режим обобщения индуктивного вывода*. Поскольку отклонения от этого свойства редки (например, пингвин не летает), то такие отклонения следует

приписывать самим вводимым понятиям. Это — *режим дискриминации индуктивного вывода.* Итак, в процессе ввода информации в сеть общие свойства обобщаются, а частные — дискриминируются.

Рассмотрим теперь, как выделяется контекст в семантических сетях с событиями и сценариями. Этот процесс осуществляется путем построения *неполных выводов, ожиданий* и *предсказаний* [Майлопулос и др., 1975]. Ввод некоторого объекта в контекст представляет собой *ожидание* системой того, что этот объект вскоре понадобится. Это ожидание создает некоторые неполные пути вывода, причем «дырки» в этих путях образуют предписание необходимой дополнительной информации. По мере ее поступления пути вывода наполняются недостающими деталями. Заметим, что таким выводам должны быть присвоены эвристические оценки с целью ограничения глубины их распространения и, следовательно, излишнего расширения контекста.

Рассмотрим три примера выделения контекста при работе со сценариями:

1. *A вырабатывает B,* где *A* —событие или сценарий, *B* — понятие (формально это могло бы быть представлено как *A* result *B,* result — глубинный падеж). Этот пример соответствует окончанию работы сценария *A* и вычислению частного случая (подстановки) *B*. Последний добавляется к контексту. Если оказывается, что *B* используется другим событием или сценарием *C* в качестве входа, то возникает предписание активации *C*, причем указанный вывод объясняет, почему *B* — предусловие *C*.
2. *A вызывает B,* т.е. *B* — следствие *A* (формальная запись *A* effect *B,* effect —-отношение причины-следствия). После вычисления *A* следует активировать *B,* найти его соответствующий частный случай и поместить в контекст. Это ожидание позволяет системе предписать большую часть сценария на основе его небольших фрагментов директив. Точно так же, как только вычислен *B,* предположение о том, что существует *A,* которое вызвало *B,* является весьма правдоподобным и сильным.
3. *B — предусловие A* (формальная запись — *A* prereq *B,* prereq — отношение причины-следствия). Здесь *B* может быть характеристикой или событием. Если *A* введено в контекст, то мы можем уверенно ввести и *B* в контекст, поскольку *A* не может быть вычислено, пока *B* не будет удовлетворено. С другой стороны, если *B* уже находится в контексте, это серьезное основание для предписания того, что *A* должно быть введено в контекст.

Таким образом, образование контекста является фактически процессом создания *виртуальной базы данных.*

Естественно, что контекст оказывает влияние на ввод новой информации в сеть. Рассматривая этот процесс, следует подчеркнуть, что наличие сценариев обуславливает определенную стратегию введения: мы заинтересованы в том, чтобы сопоставить вводимую информацию с *наиболее конкретным сценарием* из всех возможных, причем возможно, что в случае неоднозначности либо потребуется дополнительная входная информация, либо информация будет помещена на тот уровень, где еще имеется однозначность, и будет «спущена» SUB-сценариям по мере поступления уточняющей информации.

Сущность алгоритма ввода новой информации состоит в поиске всех кандидатов-сценариев, сопоставляющихся вводимой структуре путем последовательного анализа всех событий и характеристик. В случае нахождения некоторого сценария S производится сопоставление входной структуры всем SUB-сценариям S. Таким образом, создается список наиболее конкретных SUB-сценариев, сопоставляющихся входной структуре. После этого поиск идет ниже (в смысле SUB) каждого из этих SUB-сценариев в попытке найти частное сопоставление с входной структурой. В результате или часть структуры идентифицируется с уже существующими в сети вершинами, или новые вершины создаются и помещаются ниже всех сценариев в списке.

Взаимодействие описанного процесса с установленным к этому моменту контекстом заключается в том, что

— поиск производится, начиная с контекста, и лишь при неудаче распространяется вверх по сети (в смысле отношения SUP);

— в случае, если сопоставление происходит вне контекста, эта часть семантической коммуникационной сети с соответствующими путями вывода присоединяется к контексту.

Таким образом, и в сложных декларативно-процедуральных семантических предписаниях процессы выделения контекста и вложения новой информации тесно связаны.

3. Отображения логики высказываний и логики предикатов в теории коммуникаций

3.1. Основы логики высказываний

Пусть x_1 и x_2 — некоторые *высказывания,* которые могут быть истинными (1) или ложными (0), например: «Я пойду в театр» *(x_1)* и «Я встречу друга» *(x_2)*. Дизъюнкцией $x_1 \vee x_2$ является сложное высказывание «Я пойду в театр *или* встречу друга», а конъюнкцией $x_1 \wedge x_2$ — высказывание «Я пойду в театр *и* встречу друга».

Ясно, что если высказывание истинно, то его отрицание ложно. Сложное высказывание, образованное дизъюнкцией двух высказываний, истинно при условии, что истинно хотя бы одно из них. Сложное высказывание, образованное конъюнкцией двух истинных высказываний истинно, если истинны оба эти высказывания одновременно.

Итак, высказывания можно рассматривать как двоичные переменные, а связки «не», «или», «и», с помощью которых образуются сложные высказывания,— как операции над этими переменными. В алгебре высказываний используются еще две операции: *импликация* $x_1 \rightarrow x_2$, соответствующая связке «если, то» и *эквиваленция* $x_1 \sim x_2$, соответствующая связке «если и только если». Они задаются следующими таблицами:

$x_1 \rightarrow x_2$

x_1	x_2 = 0	x_2 = 1
0	1	1
1	0	1

$x_1 \sim x_2$

x_1	x_2 = 0	x_2 = 1
0	1	0
1	0	1

В нашем примере импликацией будет высказывание: *«Если'* я пойду в театр, *то* встречу друга», а эквиваленцией— «Я пойду в театр, *если и только если* встречу друга». Как видно из таблиц, импликация высказываний ложна только в случае, когда первое из простых

высказываний истинно, а второе ложно. Эквиваленция является истинным высказыванием, если оба простые высказывания истинны или ложны одновременно.

Обозначив буквами простые высказывания, можно представить сложное высказывание формулой с помощью соответствующих связок. Например, высказыванию *«Если* давление масла на шарик клапана больше усилия его пружины (x_1), *то* масло открывает клапан (x_2) *и* частично перетекает из нагнетательной полости во впускную *(x_3)»* соответствует формула $x_1 \rightarrow x_2 x_3$.

3.1.1. Закон исключения третьего

Рассматривая высказывания как двоичные переменные, обычно считают, что они удовлетворяют *закону исключения третьего:* каждое высказывание может быть истинным или ложным (третьего не дано). При этом высказывание не может быть одновременно и истинным и ложным *(закон противоречия)*. Значения «истина» и «ложь», соответствующие 1 и 0 в двузначной логике, в логике высказываний обозначаются через «И» и «Л».

Истинность данного высказывания в повседневной жизни устанавливается на основе анализа его смысла. Например, высказывание «Киев — столица Украины» — истинно, а «100 < 10» — ложно. Однако даже в таких категоричных случаях их истинность относительна. Первое предложение перестает быть истинным, если речь идет о периоде, когда столицей УССР был Харьков. Второе предложение становится истинным, если считать, что число 100 записано в двоичной системе счисления, а 10 — в десятичной («4 < 10»).

Таким образом, высказывание может быть либо истинным, либо ложным в зависимости от обстоятельств, которыми руководствуются при его истолковании. Обычно эти обстоятельства не фигурируют явно в простом высказывании. Например, истинность таких высказываний, как «Хорошая погода», «Сегодня — 16 января», «Результат измерений диаметра цилиндра равен 52 мм» зависит соответственно от вкусов или критерия оценки погоды, сегодняшней даты, требуемой точности измерения. Логика высказываний отвлекается от конкретного смысла предложений, и ответственность за их истолкование возлагает на лиц, компетентных в соответствующей области. **Она дает лишь общие методы анализа сложных высказываний и принципы логических рассуждений и доказательств.**

Принятие закона исключения третьего позволяет полностью использовать в логике высказываний аппарат двузначной логики.

Дальнейшее развитие логики высказываний основано на допущении нескольких значений истинности (например, кроме значений «истина» и «ложь» допускается третье значение — «неопределенность»). В подобных случаях используется аппарат многозначной логики. Если истинность предложений определяется с некоторой вероятностью, то логика высказываний превращается в вероятностную логику. Мы рассмотрим только двузначную логику высказываний, причем для обозначения значения «истина» будем применять 1, а значения «ложь» — 0.

3.1.2. Сентенциональные связки

Сентенциональными связками называют слова «не», «и», «или», «если..., то» и «если и только если», с помощью которых в обычном языке из простых предложений образуются сложные предложения. Как указывалось выше каждой из этих связок соответствует своя логическая операция: отрицание, крнъюнкция, дизъюнкция, импликация и эквиваленция. Обычно высказывания обозначают прописными буквами, а для операций используются те же символы, что и в алгебре логики. Таблицы соответствия в логике высказываний называют *истинностными таблицами*. Для указанных пяти связок они имеют вид:

P	0	1
$\bar{P}$	1	0

P	0	0	1	1
Q	0	1	0	1
PQ	0	0	0	1
$P \vee Q$	0	1	1	1
$P \rightarrow Q$	1	1	0	1
$P \sim Q$	1	0	0	1

Сентенциональные связки в разговорном языке допускают различные варианты. Поэтому при записи сложного предложения в виде формулы алгебры логики важно выяснить характер логической связи между предложениями, не вдаваясь в смысл самих предложений.

Истолкование *отрицания* $\bar{P}$, *конъюнкции* PQ и *дизъюнкции* $P \vee Q$ обычно не вызывает трудностей.

Импликации $P \rightarrow Q$ в обычной речи соответствует условное предложение «если P, то Q», причем P называется *посылкой (антецедентом)*, а Q — *следствием (консеквентом)*. Могут встретиться и другие выражения, имеющие тот же тип логической связи, например: «P влечет Q», «P только тогда, когда Q», «P есть достаточное условие для Q», «Q при условии, что P», «Q есть необходимое условие для P» и т. п.

Эквиваленция $P \sim Q$ определяет логическую связь в так называемых *биусловных предложениях* типа «*P*, если и только если *Q*» или в других грамматических формах: «*P* тогда и только тогда, когда *Q*», «если *P*, то *Q* и обратно, если *Q*, то *P*», «*Q* есть необходимое и достаточное условие для *P*».

3.1.3. Формулы и подстановки

Всякое сложное предложение, которое состоит из простых предложений, связанных сентенциональными связками, можно представить в символической форме. В результате получаем *высказывательную формулу*. На каждом наборе значений истинности букв (переменных) формула принимает некоторое значение. Следовательно, всякую формулу логики высказываний можно рассматривать как *истинностную функцию*.

Рассмотрим, например, сложное высказывание: «*Если* применить стальные конструкции (*P*), *то* масса снижается (*Q*) *и* стоимость увеличивается (*R*). Стальные конструкции *не* применяются ($\bar{P}$), *а* масса снижается (*Q*)». Соответствующая формула $(P \to QR)\ \bar{P}\ Q$ представляется следующей таблицей истинности:

P	0	0	0	0	1	1	1	1
Q	0	0	1	1	0	0	1	1
R	0	1	0	1	0	1	0	1
QR	0	0	0	1	0	0	0	1
$P \to QR$	1	1	1	1	0	0	0	1
$\bar{P}Q$	0	0	1	1	0	0	0	0
$(P \to QR)\ \bar{P}Q$	0	0	1	1	0	0	0	0

Отсюда видно, что сложное предложение истинно на двух наборах значений аргументов *P*, *Q*,*R*, а именно: (0, 1,0) и (0, 1. l), а на остальных наборах оно ложно.

В логике высказываний дается следующее определение формулы: 1) переменные высказывания суть формулы; 2) если *A* и *B* — формулы, то (AB), $(A \vee B)$, $(A \to B)$, $(A \sim B)$ и $\bar{A}$ также формулы. Это определение имеет *рекурсивный характер* в том смысле, что первая его часть определяет элементарные формулы, а вторая позволяет из любых формул образовать новые формулы. При записи формул используются обычные упрощения, указанные ранее. Пусть, например, требуется получить формулу $(A \to \overline{AB}) \to ((C \vee D) \to AB)$. Выбираем необходимое множество элементарных формул *A, B, C, D*. Затем последовательно получаем формулы

$$\overline{AB},\ A \to \overline{AB},\ (C \vee D) \to AB,\ (A \to \overline{AB})((C \vee D) \to AB).$$

Как видно, процесс образования формулы происходит путем расширения их множества до тех пор, пока это множество не будет содержать требуемую формулу. Все формулы, построенные в указанном процессе, называются *частями результирующей формулы*.

Если имеется некоторая высказывательная формула, то можно построить соответствующее сложное предложение, заменяя буквы простыми предложениями (одинаковые вхождения букв замещаются одним и тем же предложением). Полученное таким путем предложение называется *подстановкой в данную формулу*. Так, полагая P — «идет снег», Q — «2 × 2 = 4» и R — «слоны зеленые», по формуле $P \rightarrow QR$ получаем подстановку: «Если идет снег, то 2 × 2 = 4 и слоны зеленые». **Истинность этого высказывания определяется только приведенной выше таблицей и никоим образом не связана с конкретным содержанием как простых предложений, так и полученного в результате их объединения сложного предложения.** Как видно из таблицы, истинностная функция истинна на всех наборах значений аргументов, кроме наборов (1, 0, 0), (1, 0, 1) и (1, 1, 0). Например, при P = 0, Q= 0 и R = 1, получим истинное высказывание: «Если не идет снег, то 2 × 2 = 4 и слоны зеленые».

3.1.4. Сложные высказывания и «здравый смысл»

При первом знакомстве с логикой высказываний трудно без чувства юмора принять подобные предложения. Наш опыт подсказывает, что подвергать сомнению истину «2×2 = 4» так же нелепо, как и утверждать, что «слоны зеленые». Кроме того, между посылкой «идет снег» и ее следствием нет причинной связи. Поэтому с точки зрения «здравого смысла» такие высказывания кажутся несуразными и возможность их появления в логике высказываний следовало бы исключить.

Однако необходимо преодолеть психологический барьер и понять, что ограничения, основанные на «здравом смысле» и причинной связи в логике высказываний не только невозможны, но и нежелательны. В п. 3.1.1 уже указывалось на относительность истинности или ложности того или иного высказывания. Если бы множество допустимых высказываний было подвергнуто испытанию «здравым смыслом», то возникли бы непреодолимые трудности **из-за отсутствия строгого определения, что следует под этим понимать**. Человеку, который никогда не видел снега и не слышал о нем, фраза «идет снег» покажется бессмысленной, а высказывание «слоны зеленые» может иметь вполне определенный смысл, если речь идет, например, о выборе цвета для игрушечных слонов. Аналогичные соображения можно привести и в пользу допущения логической связи между

любыми предложениями без учета причинной зависимости между ними.

Поэтому **логика высказываний, отвлекаясь от конкретного содержания высказываний, по существу занимается лишь анализом и синтезом высказывательных формул и изучением отношений между высказываниями.** Что же касается **«здравого смысла», то он должен проявляться при использовании законов логики высказываний в ее конкретных приложениях.**

3.1.5. Тавтологии

Тождественно истинная формула, т. е. такая формула, которая принимает значения 1 при любых значениях ее компонентов, называется *тавтологией*. Тождественно ложная формула на всех наборах ее компонентов принимает значение 0 и называется *противоречием*. Если в технических приложениях логические функции, выражаемые тавтологиями или противоречиями, практически не представляют интереса, то в логике высказываний они играют первостепенную роль.

Примером тавтологии может служить высказывание: «Если внедрить новую технологию *(P),* то качество продукции улучшится *(Q)*. При улучшении качества продукции (*Q*), ее сбыт увеличивается *(R)*. Новая технология внедрена *(P)*. Следовательно, сбыт продукции увеличился *(R)»*. Оно выражается формулой $(P\to Q)(Q\to R)P\to R$.

Чтобы выяснить, является ли данная формула тавтологией, можно составить для нее истинной ную таблицу. Так, для приведенной выше формулы имеем:

P	0	0	0	0	1	1	1	1
Q	0	0	1	1	0	0	1	1
R	0	1	0	1	0	1	0	1
$P\to Q$	1	1	1	1	0	0	1	1
$Q\to R$	1	1	0	1	1	1	0	1
$(P\to Q)(Q\to R)P$	0	0	0	0	0	0	0	1
$(P\to Q)(Q\to R)P\to R$	1	1	1	1	1	1	1	1

Можно также воспользоваться зависимостями

$$x_1\to x_2=\tilde{x}_1\vee x_2,\ x_1\sim x_2=x_1x_2\vee\tilde{x}_1\tilde{x}_2=(x_1\vee\tilde{x}_2)(\tilde{x}_1\vee x_2)$$

и преобразовать высказывательную формулу к нормальной форме. Если хотя бы один член дизъюнктивной нормальной формы окажется равным 1, то соответствующая ей формула является тавтологией. Если хотя бы один член конъюнктивной нормальной формы окажется

равным 0, то соответствующая ей формула является противоречием. Так, для нашего примера имеем:

$$(P \to Q)(Q \to R)P \to R = (\bar{P} \vee Q)(\bar{Q} \vee R)P \to R =$$
$$= (\bar{P}Q \vee \bar{P}R \vee QR)P \to R = PQR \to R = \overline{PQR} \vee R = \bar{P} \vee \bar{Q} \vee \bar{R} \vee R =$$
$$= \bar{P} \vee \bar{Q} \vee (\bar{R} \vee R) = \bar{P} \vee \bar{Q} \vee 1 = 1.$$

Очевидно, формула не является тавтологией, если она принимает значение 0 хотя бы на одном наборе значений переменных. Этим обстоятельством можно воспользоваться для распознавания тавтологий сокращенным методом «обратного рассуждения», заключающемся в поиске таких переменных, при которых формула оказываекя ложной. Так, приведенная выше формула можем принять значение 0, если и только если R ложно, а $(P \to Q)(Q \to R)P$ истинно. При этом должны бьпь истинны $P \to Q$, $Q \to R$ и P. При истинном P формула $P \to Q$ истинна только при истинном Q. В свою очередь, при истинном Q формула $Q \to R$ истинна только при истинном R. Таким образом, анализируемая формула может быть ложной, если и только если R одновременно и истинно и ложно, что невозможно в силу закона противоречия. Следовательно, она является тавтологией.

Для указания на то, что данная формула является тавтологией, используется знак $\models$, который помещается перед формулой, например:

$$\models (P \to Q)(Q \to R)P \to R.$$

3.1.6. Законы логики высказываний

Различные подстановки в тавтологию, независимо от их конкретного содержания, всегда являются истинными предложениями в силу одной только своей логической структуры. Иначе говоря, тавтологии можно рассматривать как некоторые *логически истинные схемы* рассуждений или утверждений. Поэтому они играют роль *законов (теорем) логики высказываний,* претендующих на установление методов построения правильных умозаключений.

Существует бесконечное множество тавтологий, а значит, и законов логики высказываний. Наиболее часто используемые из них следующие: $P \to P$ *(закон тождества),* $P \vee \bar{P}$ *(закон исключения третьего),* $\overline{P\bar{P}}$ *(закон противоречия),* $\bar{\bar{P}} \sim P$ *(закон двойного отрицания),* $P \to (Q \to P)$ *(добавление антецедента* или *verum ex quodlibet* — истина из чего угодно), $\bar{P} \to (P \to Q)$ *(ex falso quodlibet* — из ло-

жного что угодно), $(P \to Q)P \to Q$ *(закон отделения* или *modus ponens),* $(P \to Q)\,\overline{Q} \to \overline{P}$ *(modus tollens),* $(P \to Q)(Q \to R) \to (P \to R)$ *(закон силлогизма),* $(P \to Q) \to (\overline{Q} \to \overline{P})$ *(закон контрапозиции).*

Каждый из законов логики высказываний отображает в символической форме некоторую схему доказательства. Например, в соответствии с законом отделения, если истинно, что некоторое высказывание *P* имплицирует высказывание *Q* и, кроме того, *P* истинно, то истинно и *Q.* Modus tollens применяется при доказательстве от противного: желая доказать утверждение *P,* предполагается, что *P* ложно, и показывается, что *P* имплицирует некоторое высказывание *Q,* о котором известно, что оно ложно ($\overline{Q}$ истинно). Отсюда заключается, что *P* истинно.

3.1.7. Равносильность

Две формулы называются *равносильными,* если на всех наборах значений входящих в них переменных эти формулы принимают одинаковые значения. Для обозначения этого отношения часто употребляют символ $\Leftrightarrow$, так что равносильность формул *A* и *B* символически записывается как $A \Leftrightarrow B$. Легко видеть, что равносильность — это отношение эквивалентности: оно рефлексивно $(A \Leftrightarrow A)$, симметрично (если $A \Leftrightarrow B$, то $B \Leftrightarrow A$) и транзитивно (из $A \Leftrightarrow B$ и $B \Leftrightarrow C$ следует, что $A \Leftrightarrow C$). Поэтому равносильность называют также *логической эквивалентностью.*

Равносильность формул логики высказываний вытекает из тождественности соответствующих формул алгебр и логики. Так, в соответствии с тождественными преобразованиями получаем следующие равносильности:

$\overline{\overline{A}} \leftrightarrow A;\ A \vee A \leftrightarrow A;\ AA \leftrightarrow A;\ A \vee B \leftrightarrow B \vee A;\ AB \leftrightarrow BA;\ A \vee (B \vee C) \leftrightarrow (A \vee B) \vee C;\ A(BC) \leftrightarrow (AB)C;\ A(B \vee C) \leftrightarrow AB \vee AC;\ A \vee BC \leftrightarrow (A \vee B)(A \vee C);\ \overline{A \vee B} \leftrightarrow \overline{A}\,\overline{B};\ \overline{AB} \leftrightarrow \overline{A} \vee \overline{B};\ A \vee AB \leftrightarrow A;\ A(A \vee B) \leftrightarrow A;\ A \vee \overline{A}B \leftrightarrow A \vee B$ и т. д. Кроме того, с помощью отношения равносильности выражаются различные связки между формулами:

$A \to B \leftrightarrow \overline{A} \vee B;\ A \sim B \leftrightarrow AB \vee \overline{A}\,\overline{B} \leftrightarrow (A \vee \overline{B})(\overline{A} \vee B);\ A \vee B \leftrightarrow \overline{A} \to B;$
$AB \leftrightarrow \overline{A \to \overline{B}};\ A \sim B \leftrightarrow (A \to B)(B \to A).$

Эти и подобные им равносильные соотношения можно использовать для преобразования и упрощения структуры сложного высказывания. Так, для примера из (п.3.1.3) имеем:

$(P \to QR)\ \bar{P}\,Q \Leftrightarrow (\bar{P} \vee QR)\ \bar{P}\,Q \Leftrightarrow \bar{P}\,Q \vee \bar{P}\,QR \Leftrightarrow \bar{P}\,Q.$

Между отношением равносильности и эквиваленцией формул существует следующая связь: если A и B — равносильны, то $A \sim B$ — тавтология, и обратно, если $A \sim B$ — тавтология, то A и B— равносильны. Это сокращенно записывается так: $\models A \sim B$, если и только если $A \Leftrightarrow B$. Справедливость этого утверждения следует непосредственно из определения равносильности и таблицы истинности для эквиваленции. Действительно, если $A \Leftrightarrow B$, то A может принимать только то значение, что и B и, следовательно, их эквиваленция $A \sim B$ всегда истинна и является тавтологией. Если $A \sim B$ — тавтология, то A и B могут иметь только одинаковые значения (0 или 1) и, следовательно, $A \Leftrightarrow B$.

Из изложенного ясно, что тавтологии можно получить из равносильности заменой знака $\Leftrightarrow$ на ~. Так, из равносильности $A \vee AB \Leftrightarrow A$ получаем тавтологию $\models (A \vee AB) \sim A$. Доказательств тавтологии, например $\models (A \to B)\ (A \to C) \sim (A \to BC)$ можно выполнить с помощью преобразований:

$$(A \to B)(A \to C) \Leftrightarrow (\bar{A} \vee B)(\bar{A} \vee C) \Leftrightarrow \bar{A} \vee \bar{A}B \vee \bar{A}C \vee BC \Leftrightarrow$$
$$\Leftrightarrow \bar{A} \vee BC \Leftrightarrow A \to BC.$$

3.1.8. Логическое следствие

Говорят, что формула B являсяя *логическим следствием* формулы A и пишут $A \Rightarrow B$, если B истинно на всех наборах значений переменных, для которых A истинно. Легко убедиться, что $A \Rightarrow B$, если и только если $\models A \to B$. Действительно, в соответствии с определением импликации $A \to B$ ложно только при истинном A и ложном B и, следовательно, если $A \to B$ — тавтология, то из истинности A всегда следует истинность B, т. е. $A \Rightarrow B$. Обратно, если $A \Rightarrow B$, то исключается случай, когда A истинно и B ложно, а значит $A \to B$ истинно на всех наборах значений переменных, т. е. $\models A \to B$.

Логическое следствие $A \Rightarrow B$ означаем что из истинности A следует истинность B, но если A ложно, то относительно B ничего утверждать нельзя. Это отношение обобщается на совокупность высказываний: B есть логическое следствие высказываний $A_1, A_2, ..., A_m$, если из истинности всех A_i $(i = 1, 2, ..., m)$ следует истинность B. Из определения конъюнкции можно заключить, что это сводится к соотношению $A_1\ A_2\ ...\ A_m \Rightarrow B$, необходимым и достаточным условием которого является тавтология $\models A_1 \times A_2 ... A_m \to B$.

Пусть, например, даны высказывания $(A \to B)(C \to D)$, $BD \to E$, $\overline{E}$ и необходимо установить, является ли высказывание $\overline{A} \vee \overline{C}$ логическим следствием. Это сводится к доказательству тавтологии $\models((A \to B)\ (C \to D))\ (BD \to E)\overline{E} \to (\overline{A} \vee \overline{C})$. Воспользовавшись методом «обратного рассуждения», положим, что следствие $\overline{A} \vee \overline{C}$ ложно (A и C истинны) при истинном значении всех посылок. Тогда, как следует из первой посылки, B и D должны быть истинны, а из истинности BD и второй посылки следует истинность E. Но это противоречит третьей посылке $\overline{E}$, что и доказывает данную тавтологию.

Между логическим следствием и логической эквивалентностью имеется связь, которая вытекает из соотношения $A \sim B \leftrightarrow (A \to B) \wedge (B \to A)$, приведенного в (3.1.7). Оно означает: $A \sim B$, если и только если $A \to B$ и $B \to A$. Пусть $A \sim B$ — тавтология, тогда $A \to B$ и $B \to A$ — также тавтологии, т. е. $\models A \sim B$, если и только если $\models A \to B$ и $\models B \to A$. А это равносильно утверждению: $A \Leftrightarrow B$, если и только если $A \Rightarrow B$ и $B \Rightarrow A$.

Логическое следствие есть отношение порядка; так, оно рефлексивно $(A \Rightarrow A)$, транзитивно (если $A \Rightarrow B$ и $B \Rightarrow C$, то $A \Rightarrow C$) и антисимметрично (из $A \Rightarrow B$ и $B \Rightarrow A$ следует $A \Leftrightarrow B$).

3.1.9. Правила вывода

Формальная теория вывода ставит своей главной задачей образование из некоторой совокупности исходных тавтологий новых формул, которые также являются тавтологиями. Эта задача решается с помощью *правил вывода:*

1) если A — тавтология, то, заменяя в ней букву X всюду, где она входит, произвольной формулой B, получаем также тавтологию *(правило подстановки);*

2) если A и $A \to B$ суть тавтологии, то B — также тавтология *(правило заключения)*.

Первое из этих правил почти очевидно, а второе непосредственно следует из закона *modus ponens* (6).

Формула называется *выводимой в исчислении высказываний,* если она может быть получена из конечной совокупности исходных формул путем конечного числа шагов применения правил вывода. Вообще говоря, не все тождественно истинные формулы могут быть выведены из произвольного множества тавтологий. В то же время строго

доказано, что можно выбрать такую конечную совокупность исходных тавтологий *(аксиом исчисления высказываний)*, из которой выводимы все тождественно истинные формулы. Это важное положение решает проблему *полноты исчисления высказываний.*

Предложено много различных систем аксиом исчисления высказываний. Одна из них включает следующие тавтологии:

1) $A \to (B \to A)$;

2) $((A \to B) \to A) \to A$; 3) $(A \to B) \to ((B \to C) \to (A \to C))$; 4) $AB \to A$;

5) $AB \to B$; 6) $(A \to B) \to ((A \to C) \to (A \to BC))$; 7) $A \to (A \vee B)$;

8) $B \to (A \vee B)$; 9) $(A \to C) \to ((B \to C) \to ((A \vee B) \to C)))$; 10) $(A \sim B) \to$

$\to (A \to B)$; 11) $(A \sim B) \to (B \to A)$; 12) $(A \to B) \to ((B \to A) \to (A \sim B))$;

13) $(A \to B) \to (\bar{B} \to \bar{A})$; 14) $A \to \bar{\bar{A}}$; 15) $\bar{\bar{A}} \to A$.

Выведем, например, тавтологию $AB \to BA$. Подстановка в аксиому (6) вместо A формулы AB даст $\models (AB \to B) \to ((AB \to C) \to (AB \to BC))$, что после подстановки A вместо C приводится к

$\models (AB \to B) \to ((AB \to A) \to (AB \to BA))$. Посылка в этой формуле есть аксиома (5), поэтому на основе правила заключения

$\models (AB \to A) \to (AB \to BA)$. Так как посылка в полученной тавтологии является аксиомой (4), то, применяя еще раз правило заключения, получаем $\models AB \to BA$, что и требовалось доказать.

Формализация процесса вывода имеет большое теоретическое значение и позволяет построить схему доказательства, которая может быть реализована на вычислительных машинах. Однако сложность аксиоматического подхода к выводу тавтологий заставляет искать и применять специальные правила, которые сокращают многократное применение основных правил вывода.

3.1.10. Дедуктивный метод

Более краткий и простой способ вывода основан па *теореме дедукции:* если формула B является логическим следствием формул $A_1, A_2, \ldots, A_m$, т. е. $A_1, A_2, \ldots, A_m \Rightarrow B$, то $\models A_1 \to (A_2 \to (\ldots \to (A_m \to B) \ldots))$. При этом говорят, что формула B *выводима из формул* $A_1, A_2, \ldots, A_m$.

Дадим алгебраическое доказательство теоремы дедукции, рассматривая в соответствии с (8) логическое следствие $A_1, A_2, \ldots, A_m \Rightarrow B$ как $A_1 A_2 \ldots A_m \Rightarrow B$. Преобразуем по формулам из (7) тавтологию

$$A_1 A_2 \ldots A_m \to B \leftrightarrow \overline{A_1 A_2 \ldots A_m} \vee B \leftrightarrow \bar{A}_1 \vee \bar{A}_2 \vee \cdots \vee \bar{A}_m \vee B \leftrightarrow \bar{A}_1 \vee$$

$$\vee \bar{A}_2 \vee \ldots (A_m \to B) \Leftrightarrow (A_1 \to (A_2 \to (\ldots \to (A_m + B) \ldots))).$$

Так как исходная формула —тавтология, то полученная логически эквивалентная ей формула также является тавтологией, что и требовалось доказать.

Значение теоремы дедукции состоит в том, что логическое следствие B из совокупности посылок $A_1, A_2, \ldots, A_m$ представимо в виде тавтологий типа $\models A_1A_2 \ldots A_p \rightarrow (A_{p+1} \rightarrow \ldots (A_m+B)\ldots)$. Справедливо и обратное утверждение: если имеется тавтология, содержащая цепочку импликаций типа $(A_1 \rightarrow (A_2 \rightarrow \ldots (A_p, \rightarrow (A_{p+1} \rightarrow \ldots (A_m \rightarrow B) \ldots))))$, то она может быть представлена эквивалентной формулой $\models A_1A_2 \ldots A_p \rightarrow (A_{p+1} \rightarrow \ldots (A_m \rightarrow B) \ldots)$, которой соответствует соотношение $A_1A_2 \ldots A_p \Rightarrow (A_{p+1} \rightarrow \ldots (A_m \rightarrow B) \ldots)$. Из теоремы дедукции и определения логического следствия вытекают следующие положения:

1) $A_1, A_2, \ldots, A_m \Rightarrow A_i$ $(i = 1, 2, \ldots m)$, т. е. любая из совокупности посылок является логическим следствием этой совокупности:

2) если $A_1A_2 \ldots A_m \Rightarrow B_j$ $(j = 1, 2, \ldots n)$ и $B_1, B_2, \ldots, B_n \Rightarrow B$, то $A_1, A_2, \ldots, A_m \Rightarrow B$.

С помощью этих правил можно представить доказательство того, что формула B есть логическое следствие формул $A_1, A_2, \ldots, A_m$ в виде *цепочки формул,* последней из которых является B. Промежуточные формулы $B_1, B_2, \ldots, B_n$ получаются на основании известных логических законов, аксиом и эквивалентностей. На основе теоремы дедукции используемые тавтологии и результирующее соотношение преобразуются к требуемой форме.

В качестве примера докажем, что $(A \vee B) \rightarrow C$, $C \rightarrow (D \vee E)$, $E \rightarrow F$, $\bar{D}\bar{F} \Rightarrow \bar{A}$. Из первой пары посылок на основе закона силлогизма получаем $(A \vee B) \rightarrow (D \vee E)$. Из последней посылки следует $\bar{D}$ и $\bar{F}$. Из посылки $E \rightarrow F$ и $\bar{F}$ выводим (modus tollens) $\bar{E}$. Из $\bar{D}$ и $\bar{E}$ получаем $\bar{D}\,\bar{E} \Leftrightarrow \overline{D \vee E}$, что совместно с $(A \vee B) \rightarrow (D \vee E)$ в соответствии с modus tollens дает $\overline{A \vee B} \Leftrightarrow \bar{A}\,B$, откуда выводим $\bar{A}$. Наглядно этот процесс вывода изображается диаграммой, показанной на рис. 1.

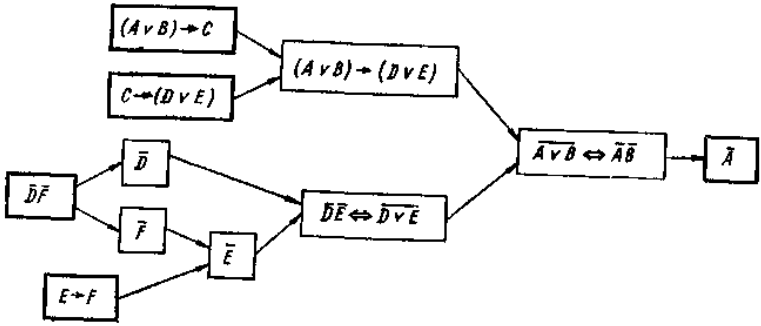


Рис. 1. Диаграмма вывода $\overline{A}$ из посылок $(A \vee B) \to C$, $C \to (D \vee E)$, $E \to F$, $\overline{D}\overline{F}$.

Если в качестве логического следствия выводится конъюнкция некоторого высказывания и его отрицания $A \wedge \overline{A}$, то это свидетельствует о *противоречивости* посылок (из нее выводится произвольное высказывание, как истинное, так и ложное).

3.2. ЛОГИКА ПРЕДИКАТОВ

Обычно высказывания выражают свойства одного или нескольких объектов. Содержательная часть высказывания играет роль определяющего свойства совокупности объектов, для которых это высказывание истинно, и называется *предикатом.* Например, высказывание «Иванов — отличник» истинно или ложно в зависимости от оценок, которые имеет данный студент. В то же время предикат *«x* — отличник» определяет подмножество отличников на некотором множестве студентов (группа, курс, факультет). Подставив вместо *x* фамилии студентов, получим множество высказываний. Совокупность истинных высказываний и будет соответствовать подмножеству отличников.

Предикат представляет собой логическую функцию *P(x),* принимающую, как и булевы функции, значение 0 или 1, но в отличие от них, значения аргумента *x* выбираются из некоторого множества *M* объектов $(x \in M)$. В общем случае такая функция может зависеть от многих аргументов $x_1, x_2, ..., x_n$, принимающих значения из одного и того же или различных множеств. Ее записывают $P(x_1, x_2, ..., x_n)$ и называют *n-местным предикатом.* Например: «*x*— четное число»,

«*x* — компонента коммуникационной цепи» —одноместные предикаты *P(x); «x* брат *y», «x* меньше *y»* — двуместные предикаты *P(x, y);* «*x* и *y* — родители z», «*x* — сумма *y* и z» — трехместные предикаты *P(x, y,* z) и т. д. Если аргументы $x_1, x_2, ..., x_n$ замещены конкретными значениями (объектами), то предикат переходит в высказывание, которое рассматривают как 0-*местный предикат.*

Так как предикаты способны принимать только значения 0 и 1, то их, как и булевы переменные, можно связывать логическими операциями. В результате получаем формулы, определяющие более сложные предикаты. Так, если *P(x)* означает *«x* — инженер», а *Q(x)* — «*x* — сотрудник нашего отдела», то $P(x) \wedge Q(x) = R(x)$ есть одноместный предикат *«x* — инженер и сотрудник нашего отдела» или проще *«x* — инженер нашего отдела». Очевидно, если *P* — множество инженеров, а *Q* — множество сотрудников данного отдела, то этот предикат соответствует пересечению $P \bigcap Q$. Таким образом, имеет место тесная связь между логикой предикатов и операциями над множествами.

3.2.1. Высказывания и предикаты

В то время как логика высказываний проявляет интерес только к логической связи между предложениями, логика предикатов проникает и в структуру самих предложений в смысле связи того, о ком или о чем идет речь *(субъект)* с тем, что говорится о данном предмете *(предикат)*. Поэтому язык логики предикатов лучше приспособлен для выражения логических связей между различными понятиями, утверждениями, коммуникационными предписаниями и директивами.
Как указывалось выше, *n-местный предикат* $P(x_1, x_2, ..., x_n)$ является неоднородной двузначной логической функцией. Аргументы $x_1, x_2, ..., x_n$ представляют собой объекты из множеств их определения $X_1, X_2, ..., X_n$, т. е. $x_1 \in X_2, x_2 \in X_2, ..., x_n \in X_n$ и называются *предметными переменными*. Конкретные значения аргументов называют *предметными постоянными*. Предметные переменные и предметные постоянные образуют класс логических понятий, называемых *термами*.
При замещении аргумента x_k (предметной переменной) некоторым его значением *a* (предметной постоянной) *n*-местный предикат $P(x_1, x_2, ..., x_n)$ превращается в *(n* — 1)-местный предикат $P(x_1 .., x_{k-1}, a, x_{k+1}, ... , x_n)$ и от переменной x_k он уже не зависит. Приписав значения всем переменным $x_1, x_2, ..., x_n$ из соответствующих областей

определения, мы получим высказывание, которое можно рассматривать как 0-*местный предикат.*

Например, трехместный предикат $P(x_1, x_2, x_3)$ = «x_1 есть сумма x_2 и x_3» при подстановке $x_1 = 5$ переходит в двуместный предикат $P(5, x_2, x_3)$ = «5 есть сумма x_2 и x_3», а при дальнейшей подстановке $x_2 = 2$ — в одноместный предикат $P(5, 2, x_3)$ = «5 есть сумма 2 и x_3». Очевидно, при $x_3 = 3$ он становится истинным высказыванием, а при всех $x_3 \neq 3$ ложным высказыванием.

3.2.2. Кванторы

В логике предикатов большое значение имеют две операции, называемые *кванторами,* с помощью которых выражают отношения общности и существования. Пусть $P(x)$ — предикат, определенный на множестве M. Утверждение, что все $x \in M$ обладают свойством $P(x)$, записывают с помощью *квантора общности* $\forall x$ в виде $\forall x P(x)$, что читается «для всех x, P от x». Утверждение, что существует хотя бы один объект x из M, обладающий свойством $P(x)$, записывают с помощью *квантора существования* $\exists x$ в виде $\exists x P(x)$, что читается «существует такое x, что P от x».

Хотя в выражениях $\forall x P(x)$ и $\exists x P(x)$ и встречается буква x, но они не зависят от значений этой переменной. Кванторы $\forall x$ и $\exists x$ *связывают переменную x,* превращая одноместный предикат в высказывание. Очевидно, $\forall x P(x)$ истинно только при условии, что $P(x)$ тождественно истинный предикат, а во всех остальных случаях это высказывание ложно. Высказывание $\exists x P(x)$ всегда истинно, кроме единственного случая, когда $P(x)$ — тождественно ложный предикат.

Рассмотрим, например, предикат $P(x)$=«x — простое число», определенный на множестве натуральных чисел. Подставляя вместо x числа натурального ряда, получаем счетное множеаво высказываний. Некоторые из них, например $P(1)$, $P(2)$, $P(3)$, $P(5)$ и т. д., являются истинными. Высказывание $\forall x P(x)$ — «все натуральные числа простые» — ложно, а $\exists x P(x)$ — «некоторые из натуральных чисел — простые» — истинно.

Между кванторами $\forall x$ и $\exists x$ имеют место соотношения, обобщающие законы де Моргана: $\forall x \overline{P(x)} = \exists x P(x)$; $\overline{\exists x} P(x) = \forall x \overline{P(x)}$.

3.2.3. Связанные и свободные переменные

Применение квантора к n-местному предикату превращает его в $(n$—1)-местный предикат. Кванторы можно также применять к нескольким различным переменным (по одному квантору какого-либо типа к каждой переменной). Если к n-местному предикату применяется k кванторов, ю он превращается в $(n - k)$-местный предикат, а при $n=k$ — в высказывание. Переменные, к которым применяются кванторы, называются *связанными,* а остальные переменные — *свободными*. Например, из двухместного предиката $P(x, y)$ с помощью кванторов получаем одноместные предикаты $\forall xP(x, y)$; $\exists xP(x, y)$; $\forall P(x, y)$ и $\exists yP(x, y)$, а также высказывания $\forall x \forall yP(x, y)$; $\forall x \exists yP(x, y)$; $\exists x \exists yP(x, y)$ и т. п.

Порядок следования одноименных кванторов не имеет значения, но разноименные кванторы переставлять нельзя. Так, $\forall x \forall yP(x, y)$ эквивалентно $\forall y \forall xP(x, y)$;, но высказывания $\forall x \exists yP(x, y)$; и $\exists y \forall xP(x, y)$, вообще говоря, различны. В этом можно убедиться на примере предиката $P(x, y)$ = «x делит y», который в первом случае превращается в высказывание «для всякого x существует такое y, что x делит y» (истинно), а во втором — «существует такое y, что любое x делит y» (ложно).

Квантор связывает переменную в области своего действия. Эта область обычно заключается в скобки, если она содержит не один предикат, а совокупность предикатов, связанных символами логических операций. Выражения, которые можно образовать применением к предикатам сентенциональных связок и кванторов, представляют собой *формулы логики предикатов*. Переменная *свободна в формуле,* если хотя бы на одно ее вхождение не распространяется действие квантора. Переменная *связана в формуле,* если она связана по меньшей мере одним квантором. Например, в формуле $\exists y \forall xP(x, y) \rightarrow \forall zQ(z))$ вхождение каждой из переменных связано, а в формуле $\forall x(P(x, y) \vee \exists y\, Q(y)) \vee R(x)$ переменная x одновременно и свободная и связанная.

3.2.4. Категорические высказывания

Перевод предложений с русского или какого-либо другого языка на символический язык логики предикатов вызывает определенные трудности из-за отсутствия механических правил. Он основан не столько на форме обычных предложений, сколько на выявлении их смысловой связи.

В традиционной логике большое внимание уделяется четырем типам *категорических высказываний,* которые обычно обозначаются заглавными латинскими буквами *A, E, I, O:*

A — общеутвердительное высказывание «Всякое *S* суть *P»:*

$\forall\ x(S(x) \to P(x))$, что означает: «Для всех *x,* если *x* обладает свойством S, то *x* обладает и свойством *P»;*

E — общеотрицательное высказывание «Никакое *S* не есть *P»:*

$\forall\ x(S(x) \to P(x))$, что означает «Для всех *x,* если *x* обладает свойством *S,* то он не обладает свойством *P»;*

I — частноутвердительное высказывание «Некоторые S суть *P»:*

$\exists x(S(x) \wedge P(x))$, что означает «Существует такой объект *x,* обладающий свойством *S,* который также обладает свойством *P»;*

O — частноотрицательное высказывание «Некоторые S не суть *P»:*

$\exists x\ (S(x)\ \wedge\ \overline{P(x)})$, что означает «Существует такой объект *x,* который обладает свойством *S* и не обладает свойством *P*».

Пусть, например, *S(x)* = «*x* — селедка» (свойство «быть селедкой») и *P(x)* = «*x* — рыба» (свойство «быть рыбой»). Тогда четырем типам категорических высказываний соответствуют следующие утверждения: *A* = «Всякая селедка — рыба»; *E* = «Никакая селедка не является рыбой»; *I* = «Некоторые селедки — рыбы»; *O* = «Некоторые селедки не являются рыбами».

На основе правил преобразования высказываний (п.3.1.7) и зависимостей между кванторами (2) можно записать: $\forall\ x(S(x) \to P(x)) \Leftrightarrow$ $\leftrightarrow \overline{\exists} x\ \overline{(S(x) \vee P(x))} \leftrightarrow \overline{\exists} x\ (S(x) \wedge \overline{P(x)})$. Аналогично преобразуются и другие типы высказываний, в результате чего получаем зависимости:

$$\forall x\ (S(x) \to P(x)) \leftrightarrow \overline{\exists} x\ (S(x) \wedge \overline{P(x)});$$
$$\forall x\ (S(x) \to \overline{P(x)}) \leftrightarrow \overline{\exists} x\ (S(x) \wedge P(x));$$
$$\overline{\forall} x\ (S(x) \to \overline{P(x)}) \leftrightarrow \exists x\ (S(x)\ \wedge\ P(x));$$
$$\overline{\forall} x\ (S(x) \to P(x)) \leftrightarrow \exists x\ (S(x) \wedge\ \overline{P(x)}).$$

Как видно из приведенных равносильностей, высказывания *A* и *O,* а также *E* и *I* являются отрицаниями друг от друга (если одно из них истинно, то другое ложно и обратно) и называются *противоположными.* Из коммутативности операции конъюнкции следует, что суждения *E* и *I* допускают перестановку предикатов *S(x)* и *P(x),* т. е.

$$\overline{\exists} x\ (S(x) \wedge P(x)) \leftrightarrow \overline{\exists} x\ (P(x) \wedge S(x));$$
$$\exists x\ (S(x) \wedge P(x)) \leftrightarrow \exists x\ (P(x) \wedge S(x)).$$

3.2.5. Непосредственные заключения

Приняв одно из категорических высказываний в качестве посылки, а другое — в качестве следствия, можно построить так называемые *непосредственные заключения.* Истинность или ложность заключения зависит только от его формы или, как говорят, от его *модуса*.

Обычно категорические высказывания сокращенно обозначают совокупностью трех букв *SaP, SeP, SiP, SoP,* где *a, c, i, o* указывают на тип высказывания *(A, E, I, O); S* и *P — понятия,* означающие свойства (в таком порядке, в каком они входят в высказывание). Например, непосредственное заключение уж $\forall\ x(S(x) \rightarrow P(x)) \rightarrow \exists x\ (P(x) \wedge S(x))$ в принятых обозначениях запишется как $SaP \rightarrow SiP$.

Простой анализ показывает, что *SiP* является логическим следствием *SaP,* а *SoP* — логическим следствием *SeP.* Высказывания *SaP* и *SeP* могут одновременно быть ложными, но не истинными и поэтому называются *противоречивыми.* Высказывания *SiP* и *SoP* могут быть одновременно истинными, но не ложными и поэтому называются *антипротиворечивыми.*

Традиционная схема отношений между категорическими высказываниями, называемая *логическим квадратом,* показана на рис. 255.

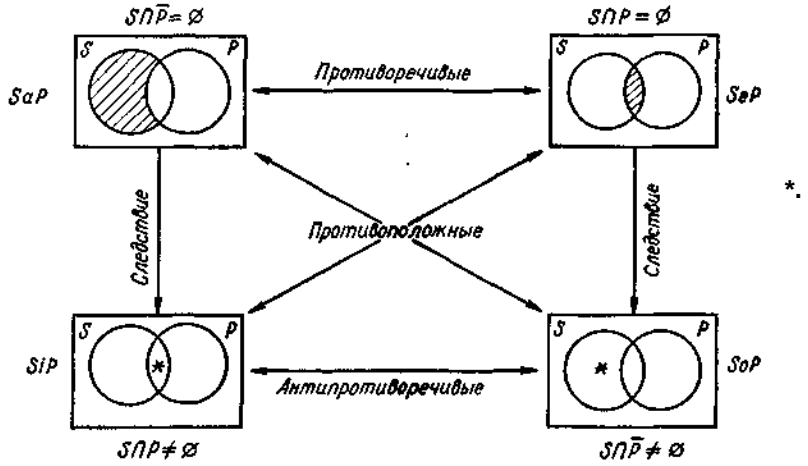


Рис. 2. Логический квадрат.

Там же приведены диаграммы Венна для каждого из четырех типов высказываний. Они непосредственно вытекают из правых частей выражений в (4) и теоретико-множественной интерпретации

логических операций над предикатами, причем заштрихованные области соответствуют пустым множествам, а отмеченные звездочкой (*) — непустым множествам. Так как $S \cap \overline{P} = \varnothing$, если и только если $S \subset P$, то высказывание *SaP* соответствует отношению включения множеств $S \subset P$. В случае высказывания *SeP* множества S и *P* являются непересекающимися, а в случае высказывания *SiP* множества S и *P* должны иметь непустую общую часть. Наконец, высказывание *SoP* в силу тождества $S \cap \overline{P} = S \backslash P$ соответствует дополнению S до *P*.

Поскольку каждый из четырех типов высказываний может быть как посылкой, так и следствием, то всего можно образовать 4 • 4 = 16 модусов непосредственных заключений с одинаковым положением понятий *S* и *P* в посылках и следствиях. Изменив порядок следования понятий в следствиях *(SP* на *PS),* получим еще 16 модусов. Итак, имеется всего 32 существенно различных модусов непосредственных заключений. Анализ (например, с помощью диаграмм Венна) показывает, что только 10 из них являются тавтологиями, т. е. *правильными модусами.* Кроме четырех модусов, в которых посылки и следствия являются одинаковыми высказываниями, и двух модусов, допускающих перестановку понятий *(SeP* $\Rightarrow$ *PeS, SiP* $\Rightarrow$ *PiS),* правильными являются модусы: *SaP* $\Rightarrow$ *SiP; SeP* $\Rightarrow$ *SoP; SaP* $\Rightarrow$ *PiS; SeP* $\Rightarrow$ *PoS.*

К таким выводам приходим, если, следуя традиционной формальной логике, считать, что понятия *S* и *P* всегда соответствуют непустым множествам, т. е. предикаты *S(x)* и *P(x)* не могут быть тождественно ложными. Если же, например, *S(x)*—тождественно ложно *(S* = $\varnothing$), то высказывания *SaP* и *SeP* всегда истинны, а *SiP* и *SoP* — ложные (это хорошо видно из рис. 2). Тем самым нарушается правильность ряда модусов традиционной логики.

Пусть, например, *S(x)* = «*x* — летающие черепахи», а *P(x)* означает «жить в зоопарке». Тогда категорические высказывания четырех типов суть следующие:

SaP = «Все летающие черепахи живут в зоопарке»,

SeP = «Никакие летающие черепахи не живут в зоопарке»,

SiP = «Некоторые летающие черепахи живут в зоопарке»,

SoP = «Некоторые летающие черепахи не живут в зоопарке».

Первые два высказывания истинны, что ясно из их эквивалентного представления: *SaP* = «Не существует такого объекта *x,* который был бы летающей черепахой и не жил в зоопарке» и *SeP* = «Не существует такого объекта *x,* который был бы летающей черепахой и жил в зоопарке». Истинность этих высказываний следует уже из того, что действительно «не существует такого объекта, который был бы

летающей черепахой», т. е. в силу тождественной ложности предиката $S(x)$. По этой же причине ложными являются два других высказывания SiP и SoP.

Ясно, что при тождественно ложном $S(x)$ высказывание I не является следствием A и высказывание O не является следствием E, т. е. модусы $SaP \Rightarrow SiP$; $SeP \Rightarrow SoP$; $SaP \Rightarrow PiS$; $SeP \Rightarrow PoS$ перестают быть правильными. Теряют смысл и некоторые отношения между высказываниями, изображенные на логическом квадрате. Традиционная логика находит выход из этого положения, не допуская тождественно ложных предикатов, а значит, и пустых множеств. Но современная логика предикатов не может встать на такую точку зрения, которая сильно сузила бы область ее применения. О целесообразности допущения пустых множеств уже говорилось ранее. Рассматривая пустое множество как подмножество любого множества, мы не нарушаем теоретико-множественных соотношений для различных типов категорических высказываний (рис. 2).

3.2.6. Категорические силлогизмы

Так называтсуждения типа $XY \rightarrow Z$, где X, Y и Z — категорические высказывания. Из истинности конъюнкции XY (она истинна только при истинных X и Y) на основании modus ponens можно выводить истинность высказывания Z. Если $\models XY \rightarrow Z$, то $XY \Rightarrow Z$ — *правильный силлогизм.*

Во всяком силлогизме X — *большая посылка,* содержащая понятия M и P; Y — *малая посылка,* содержащая понятия M и S, и Z — *заключение,* в котором S играет роль *подлежащего* и P — *сказуемого*. Таким образом, в силлогизме участвуют три понятия, называемые: S — *малый термин, M — средний термин* и P — *большой термин,* причем некоторое суждение от S и P выводится из двух высказываний — посылок, в которых участвует средний термин M, отсутствующий в заключении. Например, $MaP \bullet SaM \rightarrow SaP$ означает: «Если все M суть P и все S суть M, то все S суть P», что принято записывать в виде:

Все M суть P
Все S суть M
Все S суть P

В зависимости от порядка следования понятий в посылках совокупность силлогизмов распадается на четыре группы, называемые *фигурами* силлогизмов:

$$\frac{MP}{SM};\quad \frac{PM}{SM};\quad \frac{MP}{MS};\quad \frac{PM}{MS}.$$
$$\overline{SP}\qquad \overline{SP}\qquad \overline{SP}\qquad \overline{SP}$$

В данной фигуре каждое из высказываний может относиться к одному из четырех типов *A, E, I, O,* поэтому из нее можно образовать $4^3 = 64$ модуса, а общее количество модусов для всех четырех фигур равно $64 \cdot 4 = 256$. Основная задача теории силлогизмов состоит в выделении множества правильных модусов, т. е. таких, которые при любых конкретных понятиях позволяют из истинных посылок делать истинные заключения. Можно доказать, что из 256 модусов правильными являются только 15. Для наименования правильных модусов применяются слова, содержащие три из четырех букв *a, e, i, o,* которые указывают последовательно на типы высказываний посылок и заключения. Они выглядят (по фигурам) следующим образом:

	Barbara	Celarent	Darii	Ferio
1)	*MaP* *SaM* *SaP*	*MeP* *SaM* *SeP*	*MaP* *SiM* *SiP*	*MeP* *SiM* *SoP*
	Cezare	Camestres	Festino	Baroco
2)	*PeM* *SaM* *SeP*	*PaM* *SeM* *SeP*	*PeM* *SiM* *SoP*	*PaM* *SoM* *SoP*
	Datisi	Feriso	Disamis	Bocardo
3)	*MaP* *MiS* *SiP*	*MeP* *MiS* *SoP*	*MiP* *MaS* *SiP*	*MoP* *MaS* *SoP*
	Calemes	Fresison	Dimatis	
4)	*PaM* *MeS* *SeP*	*PeM* *MiS* *SoP*	*PiM* *MaS* *SiP*	

Традиционная логика признавала правильными еще девять модусов, которые имеют место при условии, что понятиям соответствуют непустые множества объектов. Правильность модусов доказывается на основе законов логики высказываний. Так, для модуса Celarent имеем: $\forall x\,(M(x) \to \overline{P(x)})) \;\forall x\,(S(x) \to M(x)) \Leftrightarrow \forall x((S(x) \to M(x)) \wedge (M(x) \to \to \overline{P(x)}))$. Согласно закону силлогизма $((A \to B)\,(B \to C)) \Rightarrow (A \to C)$, если для всякого x выражение $(S(x) \to M(x))\,(M(x) \to \overline{P(x)})$ истинно,

то истинно и выражение S *(x)* → $\overline{P(x)}$. Таким образом, в сокращенной записи имеем *MeP* • *SaM* ⇒ *SeP,* что и представляет собой силлогизм Celarent. Аналогично доказываются и другие правильные силлогизмы. Придавая понятиям *S, M, P* конкретное содержание, из истинных посылок всегда будем получать истинные заключения. Например, в соответствии с модусом Festino имеем:

Никакие черепахи не летают
Некоторые животные летают
Некоторые животные — не черепахи

В то время как правильность модуса требует строгого доказательства, для установления неправильности какого-либо модуса достаточно привести опровергающий его контрпример. Так, модус
MaP • *MiS* → *SaP* опровергается ложным суждением:

Всякое четное число делится на 2
Некоторые четные числа — простые
Всякое простое число делится на 2

Правильный вывод из приведенных посылок «Некоторые простые числа делятся на 2» (множество таких простых чисел содержит единственный элемент 2) следует в соответствии с модусом Datisi.

3.2.7. Символизация языка

Логика предикатов располагает более общими и универсальными методами обоснования правильных выводов, чем традиционная формальная логика. Первым этапом построения какого-либо доказательства или теории является символизация исходных положений, подвергающихся логическому анализу или принимаемых в качестве аксиом данной теории. Этот процесс обычно сводится к переводу некоторых высказываний на символический язык логики предикатов. Приведем некоторые примеры.

Рассмотрим сложное высказывание, выраженное на обычном языке: «Некоторые студенты выполнили все задания. Ни один студент не выполнял графиков. Следовательно, ни одно задание не являлось графиком». В первом предложении участвуют одноместные предикаты — свойства *P(x)* = *«x* — студент», *Q(y)* = *«y* — задание» и двуместный предикат *R(x, y)* = *«x* — выполнил *y»*. Так как в нем говорится о «некоторых студентах», то соответствующая форма будет
$\exists x(P(x) \wedge A(x))$, где *A(x)* — сложное высказывание, характеризующее предикат *P(x),* а именно: «выполнили все задания». Поскольку речь

идет о «всех заданиях», то переменная y связывается квантором общности и высказывание $A(x)$ представляется формулой $\forall\, y(Q(y) \rightarrow R(x, y))$, которая дословно переводится «для всякого y, если y — задание, то x выполнили y», смысл которого соответствует фразе «выполнили все задания». Итак, символическая запись первого предложения имеет вид: $\exists x(P(x) \wedge \forall y(Q(y) \rightarrow R(x, y)))$. Аналогично записывается и второе предложение $\forall x(P(x) \rightarrow \forall y\{S(y) \rightarrow \overline{R(x, y)})\}$, где $S(y)$ = «y — график». Заключение «Ни одно задание не являлось графиком» представляет собой категорическое высказывание типа QeS. Таким образом, получаем окончательно:

$$\exists x(P(x) \wedge \forall y(Q(y) \rightarrow R(x, y))) \wedge \forall x(P(x) \rightarrow \forall y(S(y) \rightarrow R(x, y))) \rightarrow$$
$$\rightarrow \forall x\,(Q(x) \rightarrow \overline{S(x)}).$$

Рассмотрим примеры символической записи свойств и определений. Пусть $P(x, y)$ — бинарное отношение, определенное на некотором множестве M. Рассматривая его как двуместный предикат, записываем основные свойства отношений: $\forall x P(x,x)$ — рефлексивность, $\forall x \forall y(P(x,y) \rightarrow (P(y, x))$ — симметричность, $\forall x \forall y \forall z\,(P(x,y) \wedge \wedge P(y, z) \rightarrow P(x, z))$ — транзитивность, $\forall x \forall y(P(x,y) \wedge P(y,x) \rightarrow (x = y))$ — антисимметричность и т. д. С помощью этих и подобных им выражений определяются любые типы бинарных отношений, обладающих некоторой совокупностью свойств. Так, отношение эквивалентности определяется как двуместный предикат, удовлетворяющий формуле: $\forall x\, P(x, x) \wedge \forall x \forall y\,(P(x, y) \rightarrow \rightarrow P(y, x)) \wedge \forall x \forall y \forall z(P(x, y) \wedge P(y, z) \rightarrow P(x, z))$.

Язык логики предикатов широко используется в теории коммуникаций. Поэтому необходимо научиться уверенно расшифровывать формулы, записанные на этом языке. Пусть, например, $\forall x(P(x) \rightarrow \exists y(Q(y) \wedge R(x, y)))$, где $P(x)$ = «x — простое число», $Q(x)$ = «x — четное число», $R(x, y)$ — «y делится на x». Это общеутвердительное высказывание, в котором $P(x)$ играет роль подлежащего, а $\exists y\,(Q(y) \wedge R(x, y))$ — сказуемого. В свою очередь, сказуемое является частноутвердительным высказыванием относительно переменной y (x — свободная переменная) и означает: «Существует такое четное число y, которое делится на x». Тогда исходная формула расшифровывается следующим образом: «Для всех x, если x — простое число, существует такое четное число y, которое делится на x» или проще: «Для всякого простого числа можно подыскать такое четное число, которое делится на это простое число».

3.2.8. Оценочная процедура

Истинное значение формулы в логике предикатов можно установить с помощью *оценочной процедуры.* Она сводится к определению значений входящих в данную формулу предикатов при замещении свободных переменных элементами из множества их определения. При этом последовательно используются общие свойства сентенциональных связок и кванторов. Исходными данными являются неоднородные функции, представляющие предикаты, и конкретные значения высказываний и свободных переменных, для которых требуется найти значение формулы.

Проиллюстрируем рассматриваемую процедуру на примере формулы $\forall\, x(P(x, y, z) \rightarrow \exists y Q(x, y)) \vee Q(x, y)S$, где предикаты заданы на двухэлементном множестве $\{a, b\}$ таблицами соответствия:

$P(x, y, z)$

	y	a	a	b	b
	z	a	b	a	b
x	a	0	1	1	0
	b	0	1	0	1

$Q(x, y)$

	y	a	b
x	a	0	0
	b	0	1

Пусть S = 0; $x = b$; $y = a$; $z = a$. Подставляя эти значения в формулу, получаем $\forall\, x(P(x, a, a) \rightarrow \exists\, yQ(x, y)) \vee Q(b, a) \bullet 1$. Так как $Q(b, a) = 0$, то формула упрощается к виду $\forall\, x(P(x, a, a) \rightarrow \exists y(Q(x, y))$. Это выражение представляет собой высказывание, для установления значения которого необходимо выяснить, является ли одноместный предикат в скобках истинным для всех значений x. Соответствующая таблица имеет вид:

x	$P(x, a, a) \rightarrow \exists y Q(x, y)$		
a	0	1	0
b	0	1	1

Здесь значения $P(x, a, a)$ взяты из первого столбца таблицы для $P(x, y, z)$. Значения $\exists\, yQ(x, y)$ получены на основе таблицы для $Q(x, y)$. Так как первая ее строка содержит только нули, то $\exists\, yQ(x, y)$ при $x = a$ получает значение 0. Во второй строке имеется единица, откуда заключаем, что $\exists\, yQ(x, y)$ при $x = b$ имеет значение 1. Истинностные значения выражения $P(x, a, a) \rightarrow \exists\, yQ(x, y)$ помещены в таблице под знаком импликации (так часто поступают для сокращения места).

Как видим, это выражение тождественно истинно относительно переменной x, следовательно, $\forall\, x(P(x, a, a) \rightarrow \exists\, yQ(x, y))$ также истинно, т. е. исследуемая формула имеет значение 1. Аналогично можно определить истинностные значения формулы и при других

значениях переменных *x, y, z* и высказывания *S*. Рассмотренная процедура трудоемка даже для сравнительно простых формул, особенно, если требуется найти истинностные значения на всевозможных наборах (при этом необходимо выполнить эту процедуру для всех функций *P(x, y, z)* и *Q(x, y))*.

3.2.9. Общезначимость

Особый интерес представляют общезначимые формулы, которые истинны (принимают значения 1) при каждом приписывании значений входящих в них свободных переменных и предикатов. Если *A* — общезначимая формула, то она, как и тавтологии, обозначается $\models A$.

Для доказательства общезначимости формул используется аппарат логики высказываний, дополненный теоремами для выражений, содержащих кванторы. Приведем некоторые из них.

1) Пусть *Q(x)* — формула, свободная для y; тогда:

а) $\models \forall x Q(x) \to Q(y)$; *б)* $\models Q(y) \to \exists x Q(x)$;

2) Пусть *R* — формула, не содержащая свободных вхождений переменной *x,* и *Q(x)* — какая-либо формула; тогда: а) если $\models R \to Q(x)$, то $\models R \to \forall x Q(x)$; б) если $\models Q(x) \to R$, то $\exists x Q(x) \to R$.

3) $\models Q(x)$, если и только если $\models \forall x Q(x)$ (следствие из теорем 1 и 2).

На основе этих теорем строятся правила вывода, которые, наряду с правилами исчисления высказываний (правила подстановки и заключения, теорема дедукции и др.), используются для доказательства логических следований.

Правило универсальной конкретизации (УК): из $\forall x Q(x)$, которая свободна для *y,* выводится *Q(y)* подстановкой в *Q(x)* вместо *x* переменной *y* (теорема 1 а).

Правило универсального обобщения (УО): если *Q(x)*— следствие посылок, ни одна из которых не имеет свободных вхождений *x,* то из нее выводится $\forall x Q(x)$ (теорема 2 а).

Кроме того, можно использовать еще два правила, представляющие собой аналоги приведенных выше правил для квантора существования.

Правило экзистенциальной конкретизации (ЭК) позволяет перейти от $\exists x P(x)$ к $P(\alpha)$, где α — неизвестный, но вполне определенный элемент такой, что, если $\exists x P(x)$ истинно, то $P(\alpha)$ также истинно.

Правило экзистенциального обобщения (ЭО) позволяет перейти от $P(\alpha)$ к $\exists x P(x)$, т. е., если существует такое α, что $P(\alpha)$ истинно, то истинно и $\exists x P(x)$.

В логику предикатов полностью переносятся все тавтологии, в частности соотношения: а) $\models A \sim B$, если и только если $A \Leftrightarrow B$;

б) $\models A \to B$, если и только если $A \Rightarrow B$.

3.2.10. Доказательство логического следствия

Исходя из понятия общезначимости, можно дать следующее определение *логического следствия в логике предикатов:* формула B есть логическое следствие формул $A_1, A_2, ..., A_m$, т. е. $A_1, A_2, ..., A_m \Rightarrow B$, если для каждого множества определения и для каждого приписывания формулам A_i $(i = 1, 2, ..., m)$ в этом множестве формула B истинна при условии, что все A_i истинны. При этом для всех свободных вхождений некоторой переменной x в какие-нибудь A_i выбирается одно и то же значение x из множества определения, т. е. такое x по существу рассматривают как постоянную.

Следуя общей схеме рассуждений, изложенной в (3.1.10), а также дополнительным правилам вывода (9), рассмотрим пример из (7), где $\exists x(P(x) \wedge \overline{R(x, y)))} y(Q(y) \rightarrow R(x, y)))$ и $\forall x (P(x) \rightarrow \forall y(S(y) \rightarrow \overline{R(x, y)))}$ — посылки и $\forall x(Q(x) \rightarrow \overline{S(x))}$ — заключение. Процесс доказательства представляется диаграммой, показанной на рис. 3.

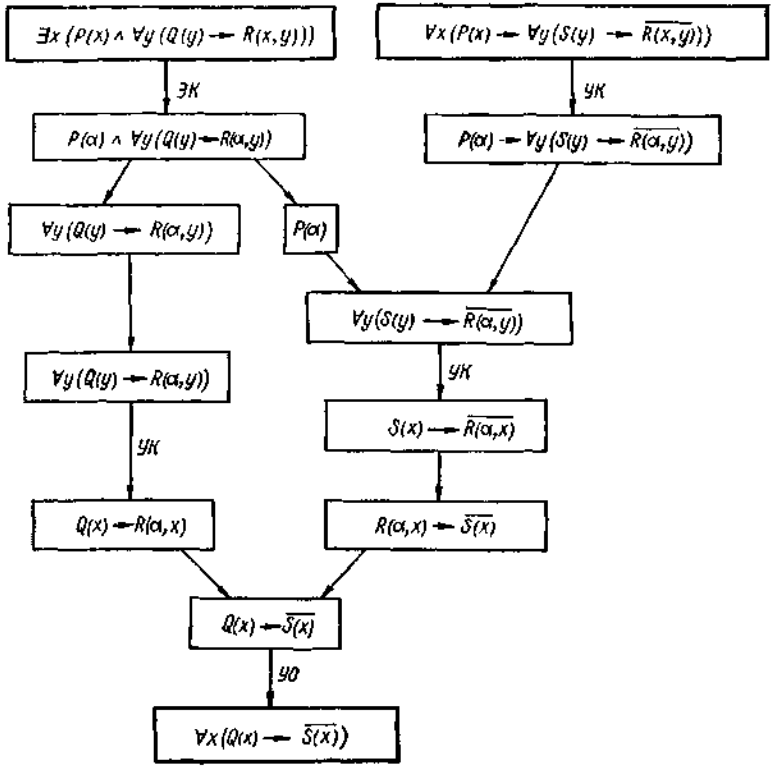


Рис.3. Диаграмма вывода $\forall x(Q(x) \rightarrow \overline{S(x)})$ из посылок

$\exists x(P(x) \wedge \overline{R(x,y)}))\, y(Q(y) \rightarrow R(x,y)))$ и $\forall x(P(x) \rightarrow \forall y(S(y) \rightarrow \overline{R(x,y)}))$.

Применение правил вывода, специфических для логики предикатов, указано здесь их сокращенными обозначениями. Остальные правила заимствованы из логики высказываний.

4. Формализация коммуникационных систем и сетей

Формальные коммуникационные системы — это системы операций над коммуникационными предметами (объектами и/или процессами), понимаемыми как последовательности символов (т. е. как слова в фиксированных алфавитах); сами операции также являются операциями над символами. Понятие «формальный» подчеркивает, что объекты (процессы) и операции над ними рассматриваются чисто формально, без каких бы то ни было содержательных интерпретаций символов. Предполагается, что между символами не существует никаких связей и отношений, кроме тех, которые явно описаны средствами самой формальной коммуникационной системы.

Если предложить читателю упорядочить объекты 53, 109, 3, то, скорее всего, он без всяких дополнительных вопросов расположит их в порядке 3, 53, 109. Иначе говоря, этой задаче будет дана обычная арифметическая интерпретация: последовательности цифр рассматриваются как изображения чисел в десятичной системе, упорядочение этих последовательностей есть расположение изображаемых ими чисел по возрастанию, а правила сравнения таких изображений чисел известны настолько хорошо, что обычно о них никто и не задумывается. В действительности же такое истолкование задачи, вообще говоря, не вытекает из текста «упорядочить объекты 53, 109, 3»; его можно понимать как задачу лексикографического упорядочения (что приведет к результату 109, 3, 53), как задачу распределения бегунов с номерами 53, 109, 3 по дорожкам (решение которой зависит от процедуры распределения и заведомо не связано с числовой интерпретацией объектов) и т. д. Возможность неоднозначного извлечения задач из указанного текста означает, что этот текст не содержит формального определения задачи. Для такого определения необходимо четко описать класс объектов, для которых задача решается, и явно ввести для них понятие упорядочения, описав его как систему локальных операций над символами, из которых эти объекты состоят.

По существу, при таком понимании «формальное описание» задачи означает ее точное, явное описание — все, что существенно для решения задачи, выписано явно. Поэтому уточнение задачи будем называть ее формализацией.

В определенном смысле проблему точного описания некоторого множества можно рассматривать как проблему построения алгоритма, перечисляющего или порождающего это множество.

Индуктивное определение формулы — простой пример описания перечислимого множества, в котором использованы все существенные составные части понятия «алгоритм», кроме одного — детерминированности. Отбрасывая несущественный здесь порядок перечисления элементов множества, мы выигрываем в компактности описания, которое при этом не становится менее точным. Такое описание, не являясь алгоритмом, представляет собой формальную систему, однозначно описывающую множество формул.

Исторически понятие формальной системы, так же как и теория алгоритмов, возникло в рамках оснований математики при исследовании строения аксиоматических теорий и методов доказательства в таких теориях. С их изучения и начнется знакомство с формальными коммуникационными системами.

При разработке и изучении коммуникационных систем возникает ряд вопросов, относящихся к конкретной коммуникационной системе, но между которыми имеется глубокая внутренняя общность. В результате оказывается возможным систематизировать их в общем плане и дать общие методы их описания и разработки, в значительной степени не зависящие от особенностей конкретной системы.

Все последующее изложение посвящено именно этим общим методам.

Ниже описаны три важнейшие проблемы, связанные с разработкой коммуникационных системы.

1. Проблема полноты. Она состоит в том, чтобы выяснить, может ли любая коммуникационная система из заданного класса (например, коммуникационная система, реализующая процессы непрерывного транспортирования определенного вида энергии) быть создана из наперед заданного набора предписаний или набора отдельных директив.

При отсутствии такого набора говорить о формализованных методах синтеза системы трудно или даже невозможно. В этом случае резко увеличивается время разработки всего комплекса предписаний и директив, так как их разработка является довольно трудоемким процессом. Более того, отсутствие такого набора предписаний чрезвычайно затрудняет определение параметров коммуникационной системы (времени ее работы, точности выполнения директив и др.).

2. Проблема выбора цели и критериев качества системы и подсистем. Установление критерия для всякой коммуникационной системы, в том числе и системы предписаний (директив), является сложным процессом и часто не поддается строгой формализации. Критерий может быть скалярным или векторным. Так, время реакции системы или среднее число запросов, обрабатываемых системой в единицу времени, являются примерами скалярных критериев.

Приведенные затраты на аппаратуру и потери от задержек при выполнении алгоритма управления — критерий векторный, так как приходится минимизировать две функции: функцию, определяющую приведенные затраты на аппаратуру, и функцию, определяющую приведенные потери, вызванные задержкой в выполнении алгоритма. Более того, этот критерий противоречив в том смысле, что чем более дешевая аппаратура (как правило, с меньшим быстродействием), тем медленнее будет выполняться алгоритм и тем выше будут потери от задержки при выполнении алгоритма управления. В таких случаях желательно выделять область компромиссов. Эта область обладает тем свойством, что все принадлежащие ей подсистемы не могут быть улучшены одновременно по всем локальным критериям — компонентам вектора качества.

В некоторых случаях это может привести к тому, что различные подсистемы коммуникационной системы могут иметь разные критерии. Более того, могут создаваться подсистемы, выполняющие одни и те же функции, но оптимизированные по разным критериям.

3. Проблема синтеза системы предписаний (директив). При решении задачи полноты и определении критерия оптимальности естественно возникает задача синтеза системы с заданным функционированием. При этом предполагается, что система будет создаваться из заданного набора подсистем и элементарных подсистем по заданному критерию.

К сожалению, в настоящее время не существует формализованных методов синтеза систем предписаний (директив). Пока можно говорить только о формализации отдельных этапов разработки системы. Такими этапами могут быть: выбор набора предписаний (директив) из заданного множества, распределение предписаний (директив) по уровням системы, компоновка предписаний (директив), и т. д.

4.1. Формальная теория коммуникаций и исчисление предписаний и/или директив

Принципы построения формальной теории коммуникаций. Теория коммуникаций определяется, во-первых, языком, т. е. некоторым множеством предписаний (директив), имеющих смысл с точки зрения этой теории, и, во-вторых, совокупностью теорем — подмножеством языка, состоящим из предписаний (директив), истинных в теории коммуникаций.

Каким образом теория коммуникаций получает свои теоремы?

Кононюк А.Е. Теория коммуникаций

В математике с античных времен существовал образец систематического построения теории — геометрия Евклида, в которой все исходные предпосылки сформулированы явно, в виде аксиом, а теоремы выводятся из этих аксиом с помощью цепочек логических рассуждений, называемых доказательствами. Однако до середины XIX в. математические теории, как правило, не считали нужным явно выделять действительно все исходные принципы; критерии же строгости доказательств и очевидности утверждений в математике в разные времена были различны и также явно не формулировались. Время от времени это приводило к необходимости пересмотра основ той или иной теории. Известно, напримео, что основания дифференциального и интегрального исчислений, разработанных в XVIII в. Ньютоном и Лейбницем, в XIX в. подверглись серьезному пересмотру; математический анализ в его современном виде опирается на работы Коши, Больцано, Вейерштрасса по теории пределов. В конце XIX в. такой пересмотр затронул общие принципы организации математических теорий. Это привело к созданию новой отрасли математики — **оснований математики, предметом которой стало как раз строение математических утверждений и теорий и которая поставила своей целью ответить на вопросы типа: «как должна быть построена теория, чтобы в ней не возникало противоречий?», «какими свойствами должны обладать методы доказательств, чтобы считаться достаточно строгими?» и т. д.** Одной из фундаментальных идей, на которые опираются исследования по основаниям математики, является идея **формализации теорий, т. е. последовательного проведения аксиоматического метода построения теорий**. При этом не допускается пользоваться какими-либо предположениями об объектах теории, кроме тех, которые выражены явно в виде аксиом; **аксиомы рассматриваются как формальные последовательности символов (выражения), а методы доказательств — как методы получения одних выражений из других с помощью операций над символами.** Такой подход гарантирует четкость исходных утверждений и однозначность выводов, однако может создаться впечатление, что осмысленность и истинность в формализованной теории не играют никакой роли. Внешне это так; однако в действительности и аксиомы, и правила вывода стремятся выбирать таким образом, чтобы построенной с их помощью формальной теории можно было придать содержательный смысл.

Более конкретно, ***формальная теория коммуникаций*** (или *исчисление)* строится следующим образом.

1. Определяется множество *формул директив*, или правильно построенных предписаний, образующее язык теории коммуникаций. Это множество задается конструктивными средствами (как правило, индуктивным определением) и, следовательно, перечислимо. Обычно оно и разрешимо.
2. Выделяется подмножество формул директив, называемых *аксиомами* теории коммуникаций. Множество может быть и бесконечным; во всяком случае оно должно быть разрешимо.
3. Задаются *правила вывода* теории коммуникаций. Правило вывода $R\ (F_1, ..., F_n, G)$ — это вычислимое отношение на множестве формул. Если формулы F_1 .... F_n, G находятся в отношении R, то формула G называется *непосредственно выводимой* из $F_1, ..., F_n$ по правилу R. Можно правило $R\ (F_1, ..., F_n, G)$ записывать в виде $\frac{F_1,...,F_n}{G}$.

Формулы $F_1, ..., F_n$ называются *посылками (отправлениями)* правила R, а G — его следствием или *заключением (получателем, адресатом)*. Примеры аксиом и правил вывода будут приведены несколько позднее.

Выводом формулы директивы B из формул директив $A_1, ..., A_n$ называется последовательность формул директив $F_1, ..., F_m$, такая, что $F_m = B$, а любая F_i (i=1, ..., m) есть либо аксиома, либо одна из исходных формул директив $A_1, ..., A_n$, либо непосредственно выводима из формул директив $F_1, ..., F_m$ (или какого-то их подмножества) по одному из правил вывода директив. Если существует вывод директив B из $A_1, ..., A_n$, то говорят, что B *выводима* из $A_1, ..., A_n$. Этот факт обозначается так: $A_1, ..., A_n \vdash B$. Формулы директив $A_1, ..., A_n$ называются гипотезами или посылками вывода директив. Переход в выводе от F_{i-1} к F_i называется i-м *шагом вывода директивы*.

Доказательством формулы директивы B в теории коммуникаций T называется вывод B из пустого множества формул, т. е. вывод, в котором в качестве исходных формул используются только аксиомы. Формула B, для которой существует доказательство, называется формулой, *доказуемой* в теории коммуникаций T, или *теоремой* теории коммуникаций T; факт доказуемости B обозначается $\vdash B$.

Присоединение формул директив к гипотезам не нарушает выводимости. Поэтому, если $\vdash B$, то $A \vdash B$ и если $A_1, ..., A_n \vdash B$, то $A_1, ..., A_n, A_{n+1} \vdash B$ для любых A и A_{n+1}. Порядок гипотез в списке несуществен.

При построении формальной теории коммуникаций мы будем иметь дело с двумя типами предписаний и/или директив: во-первых, с предписаниями и/или директивами самой теории (теоремами), которые рассматриваются как чисто формальные объекты, определенные ранее,

а во-вторых, с предписаниями и/или директивами о теории коммуникаций (о свойствах ее теорем, доказательств и т. д.), которые формулируются на языке, внешнем по отношению к теории коммуникаций, — метаязыке и называются *мета-теоремами*. Различие между теоремами и метатеоремами не всегда будет проводиться явно, но его обязательно надо иметь в виду.

Например, если удалось построить вывод B из A_1, ..., A_n, то предписание «A_1; ..., $A_n \vdash B$» является мета-теоремой; ее можно рассматривать как дополнительное («произвольное») правило вывода, которое можно присоединить к исходным и использовать в дальнейшем.

Общезначимые (тождественно-истинные) предписания типа $A \vee \bar{A}$ или $\forall x P(x) \rightarrow P(y)$, имеющче силу общих логических законов, должны содержаться в теории коммуникаций, как теории, претендующей на логический смысл.

Поэтому построение формальной теории коммуникаций начнем с исчислений, порождающих все общезначимые формулы.

Исчисление предписаний и/или директив. Аксиомы и правила вывода предписаний и/или директив.

В исчислении предписаний и/или директив мы встречаемся с объектами, с которыми уже имели дело, — с формулами алгебры логики. Однако здесь формулы рассматриваются не как способ представления функций, а как составные предписания, образованные из элементарных предписаний - директив (переменных) с помощью логических операций, или, как говорят в логике, связок $\vee$, &, $\neg$, $\rightarrow$. При этом особое внимание уделяется тождественно-истинным предписаниям, поскольку, как уже отмечалось, они должны входить в теорию коммуникаций в качестве общелогических законов. Их порождение и является основной задачей исчисления предписаний. Исчисление предписаний определяется следующим образом.

1. Алфавит исчисления предписаний состоит из переменных предписаний (пропозициональных букв): *A, B, C* ..., знаков логических связок $\vee$, &, $\neg$, $\rightarrow$ и скобок (,).

2. Формулы:

а) переменное исчисление (директива) есть формула;

б) если U и B — формулы, то $(U \vee B)$, $(U \& B)$, $(U \rightarrow B)$ и $\neg U$ — формулы;

в) других формул нет.

Внешние скобки у формул договоримся опускать: например, вместо

(U $\vee$ B) будем писать U $\vee$ B. Вместо синтаксически более удобного знака ¬ часто употребляется черта над формулой (она использовалась нами ранее).

3. Аксиомы. Приведем здесь две системы аксиом. Первая из них непосредственно использует все логические связки:

Система аксиом I

I 1. $A \to (B \to A)$;
1 2. $(A \to B) \to ((A \to (B \to C)) \to (A \to C))$;
I 3. $(A \,\&\, B) \to A$;
I 4. $(A \,\&\, B) \to B$;
I 5. $A \to (B \to (A \,\&\, B))$;
I 6. $A \to (A \vee B)$;
I 7. $B \to (A \vee 5)$;
I 8. $(A \to C) \to ((B \to C) \to ((A \vee B) \to$C);
I 9. $(A \to B) \to ((A \to \neg B) \to \neg A)$;
I 10. $\neg \neg A \to A$.

Другая система использует только две связки ¬ и →; при этом сокращается алфавит исчисления (выбрасываются знаки $\vee$, &) и соответственно определение формулы. Операции $\vee$, & рассматриваются не как связки исчисления исчисления предписаний, а как сокращения (употреблять которые удобно, но не обязательно) для некоторых его формул: $A \vee B$ заменяет $\neg A \to B$, $A \,\&\, B$ заменяет $\neg (A \to \neg B)$. В результате система аксиом становится намного компактнее.

Система аксиом II

II 1. $A \to (B \to A)$;
II 2. $(A \to (B \to C)) \to ((A \to$ B$) \to (A \to$ C$))$;
II 3. $\neg A \to \neg B) \to ((\neg A \to$ B$) \to A)$.

Приведенные системы аксиомы равносильны в том смысле, что порождают одно и то же множество формул. Такое утверждение нуждается в доказательстве, которое заключается в том, что показывается выводимость всех аксиом системы II из аксиом системы I и, наоборот, системы I из системы II (с учетом замечаний относительно $\vee$ и &). Доказательство этих выводимостей предоставляется читателю после того, как он познакомится с примерами вывода в исчислении предписаний (см. также пример 4.2, п. 1).

Возможны и другие системы аксиом, равносильные первым двум системам.

Какая из систем лучше? Это зависит от точки зрения. Система II компактнее и обозримее; соответственно более компактны и

доказательства различных ее свойств. С другой стороны, в более богатой системе I короче выводы различных формул.

4. Правила вывода.

1) Правило подстановки. Если U — выводимая формула, содержащая букву *A* (обозначим этот факт U(*A*)), то выводима формула U (B), получающаяся из U заменой *всех* вхождении *A* на произвольную формулу B:

$$\frac{A(A)}{A(V)}.$$

2) Правило заключения (Modus Ponens). Если U и U → B — выводимые формулы, то B выводима:

$$\frac{A, A \to V}{V}.$$

В этом описании исчисления предписаний аксиомы являются формулами исчисления (соответствующими определению формулы); формулы же, использованные в правилах вывода (U, U → B и т. д.), это «метаформулы», или *схемы формул*. Схема формул, например U → B, обозначает множество всех тех формул исчисления, которые получаются, если ее метапеременные заменить формулами исчисления: скажем, если U заменить на *A*, а B — на *A* & *B*, то из схемы формул U → B получим формулу *A* →*A* & *B*.

Использование схем формул можно распространить и на аксиомы. Например, если в системе II переменные (пропозициональные буквы) *A, B, C* заменить метапеременными U, B, S , то получаются три схемы аксиом, задающие три бесконечных множества аксиом. В результате возникает другой способ построения исчисления предписаний: с бесконечным множеством аксиом (задаваемым конечным числом схем аксиом), но *без правила подстановки,* поскольку оно неявно содержится в истолковании схем аксиом. Первый способ более последовательно конструктивен: все его средства явно зафиксированы и конечны; при вводе исчисления в ЭВМ (например, при автоматизации доказательства предписаний) он выглядит более естественным. С другой стороны, второй способ больше соответствует математической традиции истолкования формул: например, алгебраическое тождество $(a + b)^2 = a^2 + 2ab + b^2$ или тождества булевой алгебры истолковываются именно как схемы тождеств, а не конкретные тождества, верные лишь для конкретных букв. Правило подстановки при этом подразумевается. Переход от одного способа построения исчислений к другому не представляет труда.

Рассмотрим теперь примеры вывода в исчислении предписаний.

Пример 4.1. а. Покажем, что формула $A \to A$ выводима из системы аксиом II:

$$|\!\!- A \to A.$$

1. $(A \to ((A \to A) \to A)) \to ((A \to (A \to A)) \to (A \to A))$ (подстановка в аксиому II 2 $A \to A$ вместо B и A вместо C).
2. $A \to ((A \to A) \to A)$ (подстановка в II 1 $A \to A$ вместо B).
3. $(A \to (A \to A)) \to (A \to A)$ (из шагов 2, 1 по правилу заключения).
4. $A \to (A \to A)$ (подстановка в II 1 A вместо B).
5. $A \to A$ (из шагов 4, 3 по правилу заключения).
6. $A \vdash B \to A$.

Пусть A выводима. Тогда из A и аксиомы II 1 по правилу заключения получаем $\dfrac{A, A \to (B \to A)}{B \to A}$, что и доказывает искомую выводимость.

Как уже отмечалось ранее, всякую доказанную в исчислении выводимость вида Г |— U, где Г — список формул, U — формула, можно рассматривать как правило вывода $\dfrac{\Gamma}{A}$, которое можно присоединить к уже имеющимся. Полученную нами выводимость $A \vdash B \to A$ вместе с правилом подстановки можно рассматривать как правило $\dfrac{A}{V \to A}$ «если формула U выводима, то выводима и формула В →U, где В — любая формула». Воспользуемся этим правилом в следующем примере.

в. $A \to B,\ B \to C \vdash A \to C$.

1. $B \to C \vdash A \to (B \to C)$ (по новому правилу $\dfrac{A}{V \to A}$).
2. Из $A \to (B \to C)$ и аксиомы II 2 по правилу заключения следует $(A \to B) \to (A \to C)$; следовательно,

$$B \to C \vdash (A \to Я)\text{^}(A \to C).$$

3. Из $(A \to B)$ и $(A \to B) \to (A \to C)$ по правилу заключения следует $A \to C$; учитывая 2, получаем искомую выводимость.

При переходе от 1 к 2 неявно использовалось следующее свойство выводимости: если Г |— U (Г — список формул), а U |— В, то Г |— В. Это свойство *(транзитивность отношения выводимости)* непосредственно следует из определения выводимости.

Основные метатеоремы исчисления предписаний. Для получения выводов в исчислении предписаний оказывается полезной следующая метатеорема.

Теорема 4.1 (теорема дедукции).
Если Г, U |— B, то Г |— U → B (Г — множество формул, U, B — формулы).

Будем исходить из системы аксиом II и рассматривать их как схемы аксиом (т. е. не пользоваться правилом подстановки).

Пусть Г, U |— B. Тогда существует вывод B_1, ..., B_n из Г, U, такой, что B_n = B. Докажем по индукции, что для любого $k \le n$ Г |— U → B_k.

Рассмотрим сначала B_1. B_1, как первая формула вывода, должна либо быть аксиомой, либо содержаться в Г, либо совпадать с U. Из схемы аксиом II 1 следует, что $B_1 \to (U \to B_1)$ является аксиомой. Если B_1 — аксиома или содержится в Г, то по правилу заключения $U \to B_1$ выводима из Г. Если же B_1 = U, то из примера 4.1а имеем U → U, т. е. U → B_1. В любом случае получаем Г |— U →B_1.

Предположим теперь, что Г |— U →B_i для любого $i < k$, и рассмотрим B_k. Возможны четыре случая: а) B_k — аксиома; б) $B_k \in$ Г; в) B_k = U; г) B_k выводимо из некоторых предшествующих формул B_j, B_l по правилу заключения; но тогда B_l должно иметь вид $B_j \to B_k$. В случаях «а»—«в» доказательство точно такое же, как и для B_1 (случаи «а», «б» — с помощью аксиомы II 1; случай «в» — с помощью примера 4.1а). В случае «г» по индуктивному предположению имеем:

$$\Gamma \vdash U \to B_j \qquad (4.1)$$

и Г |— U →B_l, т. е.

$$\Gamma \vdash U \to (B_l \to B_k). \qquad (4.2)$$

Подставим в схему аксиом II 2 B_j вместо B и B_k вместо S. Получим:

$$(U \to (B_j \to B_k)) \to ((U \to B_j) \to (U \to B_k)). \qquad (4.3)$$

Применив правило заключения к (4.2) и (4.3), получим:

$$\Gamma \vdash (U \to B_j) \to (U \to B_k), \qquad (4.4)$$

а применив то же правило к (4.1) и (4.4), имеем: Г |— U → B_k. Остается положить $k = n$.

Отметим, что при построении выводов использовались только аксиомы II 1 и II 2, которые содержатся и в системе аксиом I. Поэтому приведенное доказательство теоремы дедукции справедливо и для исчисления высказываний, основанного на системе I.

Пример 4.2. а. В качестве первого применения теоремы дедукции покажем, что аксиома II 3 выводима из системы аксиом I.

1. Подставим в I 9 $\neg A$ вместо A. Получим:

   $(\neg A \to B) \to ((\neg A \to \neg B) \to \neg\neg A)$.

2. Двойное применение правила заключения к шагу 1 дает:

   $\neg A \to B, \neg A \to \neg B \vdash \neg\neg A$.

3. Так как из аксиомы I 10 следует по правилу заключения, что

$\neg\neg A \vdash A$, то по транзитивности выводимости (см. замечание к примеру 4.1 в) получим $\neg A \to B; \neg A \to \neg B \vdash A$.

4. Переставим гипотезы в полученной выводимости (их порядок неважен, как видно из определения выводимости):

$$\neg A \to \neg B; \neg A \to B \vdash A.$$

5.Применив два раза к шагу 4 теорему дедукции, получим аксиому II 3: $(\neg A \to \neg B) \to ((\neg A \to B) \to A)$.

При доказательстве выводимости системы II из системы I были использованы аксиомы системы I, не содержащие дизъюнкции и конъюнкции.

б. Важным методом математических доказательств является метод доказательства от противного: предполагаем, что A верно, и показываем, что, во-первых, из A выводится B, а во-вторых, что из A выводится $\neg B$, что невозможно, и, следовательно A неверно, т. е. верно $\neg A$. Этот метод формулируется как правило: «если $\Gamma, A \vdash B$ и $\Gamma, A \vdash \neg B$, то $\Gamma \vdash \neg A$». Докажем, что оно в исчислении предписаний выполняется. Действительно, по теореме дедукции, если $\Gamma, A \vdash B$ и $\Gamma, A \vdash \neg B$, то $\Gamma \vdash A \to B$ и $\Gamma \vdash A \to \neg B$. Из этих импликаций и аксиомы I 9 двойным применением правила заключения получаем $\Gamma \vdash \neg A$. Доказанное правило называется также правилом *введения отрицания.*

в. Докажем теперь закон исключенного третьего: $\vdash A \vee \neg A$.

1. $\neg (A \vee \neg A), A \vdash A \vee \neg A$ (аксиома I 6 при $B = \neg A$ и правило заключения).

2. $\neg (A \vee \neg A), A \vdash \neg (A \vee \neg A)$ (очевидно).

3. Применяя к шагам 1 и 2 только что доказанное правило введения отрицания, получаем:

$$\neg (A \vee \neg A) \vdash \neg A.$$

4. Аналогично доказывается $\neg (A \vee \neg A) \vdash \neg\neg A$.

5. Применяя к шагам 3 и 4 введение отрицания, получаем: $\vdash \neg\neg (A \vee \neg A)$.

6. С помощью аксиомы I 10 и правила заключения снимаем двойное отрицание в шаге 5 и получаем $\vdash A \vee \neg A$.

Исчисление предписаний и алгебра логики. Формула F исчисления предписания содержательно интерпретируется как составное предписание, истинность которого зависит от истинности входящих в него элементарных предписаний (директив). Эта зависимость в

точности соответствует зависимости значения логической функции, представляемой формулой F, от значений переменных этой функции. Иначе говоря, если задана формула $F(A_1, ..., A_n)$ и распределение истинностей входящих в нее элементарных предписаний $A_1, ..., A_n$, то для выяснения истинности ее нужно вычислить как логическую функцию на наборе $(\sigma_1, ..., \sigma_n)$, где $\sigma_i = 1$, если A_i истинно, и $\sigma_i = 0$, если A_i ложно. (В этом смысле можно считать, что набор $(\sigma_1, ..., \sigma_n)$ задает распределение истинностей.) Если $F(\sigma_1, ..., \sigma_n) = 1$, то предписание F истинно при данном распределении истинностей $A_1, ..., A_n$, если $F(\sigma_1, ..., \sigma_n) = 0$, то F ложно.

Возникает вопрос, как связано такое содержательное, «истинностное» истолкование формул с их выводимостью в исчислении предписаний? Для описания этой связи введем обозначения вида x^σ. Пусть для элементарных предписаний $A_1, ..., A_n$ задано распределение истинностей $(\sigma_1, ..., \sigma_n)$.

Обозначим

$$A_i^{\sigma_i} = \begin{cases} A_i, \text{ если } \sigma_i = 1, \text{ т. е. если } A_i \text{ истинно;} \\ \neg A_i, \text{ если } \sigma_i = 0, \text{ т. е. если } A_i \text{ ложно.} \end{cases}$$

Теорема 4.2. Пусть формула $U(A_1, ..., A_n)$ определяет логическую функцию f от n переменных. Тогда, если $f(\sigma_1, ... \sigma_n) = \sigma$, то в исчислении предписаний $A_1^{\sigma_1}, ... A_n^{\sigma_n} \vdash U^\sigma$.

Например, формула $U(A_1, A_2, A_3) = (A_1 \to A_2) \to A_3$ на наборе (0, 1, 0) равна нулю. Тогда теорема утверждает, что

$$\neg A_1, \neg A_2, \neg A_3 \vdash ((A_1 \to A_2) \to A_3).$$

Доказательство теоремы проводится индукцией по построению формулы (см. определение формулы).

Если U — буква A_i, то утверждение теоремы сводится к $A_i \vdash A_i$ и $\neg A_i \vdash \neg A_i$, что верно.

Пусть теорема верна для некоторых формул B и S. Тогда нужно доказать, что она верна для формул U, имеющих вид $\neg B$, $B \to S$, $B \,\&\, S$ и $B \vee S$. Перебор всех случаев опускаем; приведем лишь некоторые.

Пусть $U = \neg B$ и при заданном $(\sigma_1, ... \sigma_n)$ B ложно. По индуктивному предположению $A^{\sigma_1}, ... A^{\sigma_n} \vdash \neg B$ и, следовательно, $A^{\sigma_1}, ... A^{\sigma_n} \vdash \neg U$, что и требовалось.

Пусть $U = B \to S$ и S истинно. Тогда (в силу свойств импликации) U тоже истинно и $U^\sigma = U$. По индуктивному предположению

A^{σ_1}, ... A^{σ_n} ⊢¬ S . Но тогда по производному правилу из примера 4.1б A^{σ_1}, ... A^{σ_n} ⊢¬ B→ S ,т. е. A^{σ_1}, ... A^{σ_n} ⊢¬ U , что и требовалось.

Теоремы исчисления предписаний в понятиях истинности характеризуются довольно просто.

Теорема 4.3. Всякая теорема исчисления предписаний является тождественно-истинным предписанием.

Тождественная истинность аксиом проверяется либо прямым вычислением на всех наборах, либо приведением их к константе 1 путем тождественных преобразований булевой алгебры. Очевидно, что любая подстановка в тождественно-истинную формулу также даст тождественно-истинную формулу.

Остается показать, что правило заключения сохраняет тождественную истинность. Пусть формулы U и U → B тождественно истинны, т. е. U ≡ 1, U → B ≡ 1.

Так как $A \to B \equiv \neg A \vee B$, то 0 ∨ B ≡ 1 и B ≡ 1, т. е. формула B, выводимая из U и U → B по правилу заключения, также тождественно-истинна.

Итак, аксиомы тождественно-истинны, а правила вывода сохраняют тождественную истинность; поэтому любая доказуемая формула тождественно-истинна.

Справедлива и обратная теорема.

Теорема 4.4. Всякая тождественно-истинная формула является теоремой исчисления предписаний.

Пусть U(A_1, ..., A_n) — тождественно-истинная формула. Тогда по теореме 4.2 $A_1^{\sigma_1}$, ... $A_n^{\sigma_n}$ ⊢U для всех 2^n наборов (σ_1, ..., σ_n). Применив n раз теорему дедукции, получим 2^n выводимостей

$\vdash A_1^{\sigma_1} \to (A_2^{\sigma_2} \to (\ldots \to (A_n^{\sigma_n} \to U))\ \ldots)$.

Подставим в аксиому I 8 $\neg A$ вместо B. Получим

$$(A \to C) \to ((\neg A \to C) \to ((A \vee \neg A) \to C).$$

Из этой формулы и правила заключения (учитывая, что $A \vee \neg A$ — теорема: см. пример 4.2в) получаем производное правило: если ⊢$A \to$ C и ⊢ ¬$A \to$ C, то ⊢ C. Для любого набора (σ_2.....σ_n) имеем при $\sigma_1 = 1$

⊢ $A_1 \to (A_2^{\sigma_2} \to (\ldots \to (A_n^{\sigma_n} \to U))\ldots)$, а при σ_1=0

⊢ ¬$A_1 \to (A_2^{\sigma_2} \to (\ldots \to (A_n^{\sigma_n} \to U))\ldots)$. Применяя к этим формулам только что доказанное правило, получаем

$\vdash(A_2^{\sigma_2} \to ... \to (A_n^{\sigma_n} \to U)...)$, т.е. первая посылка «длинной импликации» удалена. Применяя это удаление еще (n— 1) раз, получаем $\vdash$ U.

Таким образом, исчисление предписаний действительно выполняет задачу порождения общелогических законов — тождественно-истинных предписаний.

В заключение отметим следующее. Эквивалентные соотношения булевой алгебры соединяют знаком = формулы, которые одновременно равны нулю или единице. В логике такая равнозначность выражается функцией ~; если $f_1 = f_2$— эквивалентное соотношение, то формула $f_1 \sim f_2$ является тождественно-истинным предписанием. В исчислении предписаний формула $f_1 \sim f_2$ является сокращением формулы $(f_1 \to f_2)$ & $(f_2 \to f_1)$. В силу теоремы 4.4 все такие эквивалентности, например $\overline{A \vee B} \sim \overline{A} \,\&\, \overline{B}$, являются теоремами исчисления предписаний.

Некоторые общие свойства исчисления предписаний будут рассмотрены в р. 4.3.

4.2. Исчисление понятийных предикатов

Аксиомы и правила вывода. 1. Алфавит исчисления понятийных предикатов состоит из *предметных переменных* $x_1, x_2, ...$, *предметных констант* $a_1, a_2, ...$, *предикатных букв (понятий)* $P_1^1, P^1_2, ..., P^j_k ...$ и *функциональных букв* $f_1^1, f^1_2, ..., f^j_k ...$, а также знаков логических связок $\vee, \&, \neg, \to$, *кванторов* $\forall, \exists$ и скобок (,).

Верхние индексы предикатных и функциональных букв указывают число аргументов, их нижние индексы служат для обычной нумерации букв. Переменные предписания в исчисление понятийных предикатов вводятся либо непосредственно как пропозициональные буквы $A_1, A_2 ...$, либо как 0-местные понятийные предикаты) $P_1^1, P^1_2, ...$, т. е. как понятийные предикаты без предметных переменных.

2. Понятийные формулы. Понятие формулы определяется в два этапа.

1) Понятийные термы:

а) предметные переменные и константы являются понятийными термами;

б) если f^n — функциональная буква, а $t_1, ..., t_n$ — понятийные термы, то $f^n(t_1, ..., t_n)$ — понятийный терм.

2) Понятийные формулы:

а) если P^n— понятийная предикатная буква (понятие), а $t_1, ..., t_n$ — понятийные термы, то $P^n(t_1, ..., t_n)$ — понятийная формула; все

вхождения предметных переменных в формулу вида $P^n(t_1, ..., t_n)$ называются свободными;

б) если F_1, F_2 — формулы, то формулами являются $\neg F_1$, $(F_1 \rightarrow F_2)$, $(F_1 \vee F_2)$, $(F_1 \& F_2)$; все вхождения переменных, свободные в F_1, F_2, являются свободными и в указанных четырех видах формул;

в) если $F(x)$ — формула, содержащая свободные вхождения переменной x, то $\forall x F(x)$ и $\exists x F(x)$ — формулы; в этих формулах все вхождения переменной x называются связанными; вхождения остальных переменных в F остаются свободными.

Функциональные буквы и термы введены «впрок», для целей различных прикладных исчислений понятийных предикатов. Чистое исчисление предикатов строится для произвольной предметной области; структура этой области и связи между ее элементами не имеют значения, поэтому в нем функциональные буквы и термы не обязательны (В частности, приводимые далее аксиомы Р1 и Р2 не учитывают наличия термов в формулах. Для исчислений с термами и функциональными буквами эти аксиомы имеют вид Р1' и Р2'.).

В прикладных исчислениях (например, в формальных коммуникационных системах и сетях) структура предметной области оказывается существенной, поэтому в исчислении необходимо иметь средства для описания связей между элементами, т. е. функций и отношений, определенных на области. Отношениям соответствуют понятийные предикатные буквы (понятия), функциям — функциональные буквы. Термы — это имена элементов предметной области, построенные с помощью функций. Они могут быть постоянными (если они построены из предметных констант) и переменными. Формулы — это предписания о понятийныз термах Например, 4+5•3 — постоянный терм любого исчисления, содержащего функциональные буквы + и •, а x + 7 — переменный терм этого же исчисления. Выражение 4+ 5•3= x+ 7 — это переменное предписание, полученное подстановкой двух термов в двухместный понятийный предикат равенства; его истинность зависит от значения переменной x.

3. Аксиомы исчисления понятийных предикатов делятся на две группы:

1) аксиомы исчисления предписаний (можно взять любую из систем I или II) и 2) две следующие понятийные предикатные аксиомы:

Р1. $\forall x F(x) \rightarrow F(y)$;

Р2. $F(y) \rightarrow \exists x F(x)$.

В этих аксиомах $F(x)$ — любая формула, содержащая свободные вхождения x, причем ни одно из них не находится в области действия

квантора по *y;* формула *F (y)* получена из *F (x)* заменой всех *свободных* вхождений *x* на *y*.

Чтобы пояснить существенность требования к вхождениям *x* в *F,* рассмотрим в качестве *F (x)* формулу $\exists y P(y, x)$, где это требование нарушено: свободное вхождение *x* находится в области действия $\exists y$. Подстановка этой формулы в аксиому P1 дает формулу $\forall x \exists y P(y, x) \rightarrow \exists y P(y, y)$. Если ее проинтерпретировать на множестве *N* натуральных чисел с предикатом *P* «быть больше», то получим высказывание: «если для всякого *x* найдется *y* больший, чем *x,* то найдется *y,* больший самого себя». Посылка этой импликации истинна на *N,* а ее заключение ложно, и, следовательно, само высказывание ложно.

4. П р а в и л а вывода:

1) правило заключения (Modus Ponens) — то же, что и в исчислении предписаний;

2) правило обобщения ($\forall$-введения):

$$\frac{F \rightarrow G(x)}{F \rightarrow \forall x G(x)},$$

где *G (x)* содержит свободные вхождения *x,* а *F* их не содержит;

3) правило $\exists$-введения:

$$\frac{G(x) \rightarrow F}{\exists x G(x) \rightarrow F}$$

при тех же требованиях к *F* и G, что и в предыдущем правиле.

Нарушения этих требований могут привести к ложным выводам из истинных предписаний. Пусть, например, *P (x)* — понятийный предикат «*x* делится на 6», *Q* (*x*) — понятийный предикат «*x* делится на 3». Предписание $P(x) \rightarrow Q(x)$, очевидно, истинно (выполнимо) для любого *x,* однако применение к нему правила обобщения дает предписание $P(x) \rightarrow \forall x Q(x)$, не являющееся всегда истинным (выполнимым). Если же к $P(x) \rightarrow Q(x)$ применить правило $\exists$-введения, то получим $\exists x P(x) \rightarrow$ Q (x), из которого путем (уже корректного!) применения правила обобщения получим предписание $\exists x P(x) \rightarrow \forall x Q(x)$, ложное (невыполнимое) на множестве натуральных чисел.

В теории коммуникаций возможны и другие системы аксиом и правил. В частности, существует метод последовательного проведения принципа минимизации числа логических операторов, который в исчислении предписаний оставляет лишь связки $\neg$ и $\rightarrow$ (что отражено в системе аксиом II). В исчислении понятийных предикатов он

выражается в том, что понятийный квантор $\exists x$ не считается самостоятельным символом, а рассматривается как сокращение предписания $\neg \forall x \neg$: например, предписание $\exists x P(x)$ эквивалентно предписанию $\neg \forall x \neg P(x)$.

Как видно из вышеизложенного, правило подстановки исчезло из изложения; тем самым из двух возможных истолкований системы аксиом (о чем шла речь в исчислении предписаний) выбрано второе, при котором правило подстановки отсутствует, а вместо аксиом рассматриваются схемы аксиом. Фактически этот выбор произошел уже тогда, когда аксиомы Р1 и Р2 были сопровождены словесным описанием ограничений на вхождения переменных. Тем самым аксиомы перестали быть предписаниями исчисления, а вместе с этим словесным текстом прекратились в метаописания множества формул, являющихся аксиомами, т. е. в схемы аксиом. Построение исчисления понятийных предикатов с правилом подстановки существенно более громоздко из-за необходимости различать свободные и связанные вхождения предметных переменных. Поэтому в большинстве случаев в логике используется подход со схемами аксиом. Предполагая, что после знакомства с исчислением предписаний разница между переменными и метапеременными уже усвоена, не будем больше употреблять готические буквы; в качестве метапеременных, обозначающих формулы, в этом разделе будут использоваться буквы *F* и G.

Приведем теперь примеры вывода в исчислении понятийных предикатов.

Пример 4.3. а. Покажем, что в исчислении понятийных предикатов из выводимости формулы $F(x)$, содержащей свободные вхождения x, ни одно из которых не находится в области действия квантора по y, следует выводимость $F(y)$. Это утверждение представляет собой *правило переименования свободных переменных.*

1. $\vdash F(x)$ (по условию).
2. $F(x) \to (G \to F(x))$ (аксиома II 1; в качестве G выбираем любую доказуемую формулу, не содержащую свободных вхождений x: ее доказуемость понадобится на шаге 5, а ограничение на x — на шаге 4).
3. $G \to F(x)$ (правило заключения к шагам 1 и 2).
4. $G \to \forall x F(x)$ (правило обобщения к шагу 3).
5. $\forall x F(x)$ (правило заключения к G и шагу 4).
6. $F(y)$ (правило заключения к шагу 5 и аксиоме Р1). .

б. В исчислении понятийных предикатов из выводимости $\forall x F(x)$ следует выводимость $\forall y F(y)$, а из выводимости $\exists x F(x)$ — выводимость $\exists y F(y)$ при условии, что $F(x)$ не содержит свободных

вхождений *y* и содержит свободные вхождения *x,* ни одно из которых не входит в область действия квантора по *y (правило переименования связанных переменных)*.

Докажем это правило для квантора общности.

1. $\vdash \forall x F(x)$ (по предположению).
2. $\forall x F(x) \rightarrow F(y)$ (аксиома Р1).
3. $\forall (x) F(x) \rightarrow \forall y F(y)$ (правило обобщения к шагу 2).
4. $\forall y F(y)$ (правило заключения к шагам 1 и 3).

Доказательство для 3 аналогично, но использует аксиому Р2 и правило $\exists$-введения.

Выводимость и истинность. Эквивалентные преобразования.

$\exists$ Всякая доказуемая формула исчисления понятийных предикатов тождественно-истинна (общезначима).

Эта теорема доказывается аналогично теореме 4.3: непосредственно проверяется общезначимость аксиом и показывается, что правила вывода сохраняют общезначимость, т.е. их применение к общезначимым формулам снова дает общезначимые формулы.

Теорема 4.6. Всякая общезначимая предикатная формула доказуема в исчислении понятийных предикатов.

Доказательство этой теоремы намного более сложно; здесь оно опускается.

В конце раздела об исчислении предписаний мы упоминали о том, что всякому эквивалентному соотношению $F — G$ в булевой алгебре соответствует доказуемая эквивалентность $\vdash F \sim G$ в исчислении предписаний. Из теорем 4.5 и 4.6 следует, что между соотношениями содержательной логики предикатов и формальными эквивалентностями в исчислении понятийных предикатов имеется аналогичное соответствие (напомним, что $F \sim G$ рассматривается как сокращение $(F \rightarrow G) \& (G \rightarrow F)$). На нем остановимся подробнее. Дело в том, что доказательства общезначимости в логике понятийных предикатов существенно сложнее, чем в логике предписаний, и поэтому формальный вывод эквивалентностей становится важным способом их получения.

Теорема 4.7. Пусть $F(A)$ — формула, в которой выделено вхождение формулы A; $F(B)$ — формула, полученная из $F(A)$ заменой этого вхождения A формулой B. Тогда если $\vdash A \sim B$, то $\vdash F(A) \sim F(B)$.

Эта теорема формулирует для исчисления понятийных предикатов правило, аналогичное правилу замены эквивалентных подформул в алгебрах. Благодаря ему можно получать доказуемые эквивалентности в исчислении, не строя их непосредственного вывода.

При доказательстве теоремы используются следующие производные правила:

1) если $\vdash A \sim B$, то $\vdash A \to C \sim B \to C$ и $C \to A \sim C \to B$;
2) если $\vdash A \sim B$, то $\vdash A \vee C \sim B \vee C$ и $\vdash C \vee A \sim C \vee B$;
3) если $\vdash A \sim B$, то $\vdash A \,\&\, C \sim B \,\&\, C$ и $\vdash C \,\&\, A \sim C \,\&\, B$;
4) если $\vdash A \sim B$, то $\vdash \neg A \sim \neg B$;
5) если $\vdash F(x) \sim G(x)$, то $\forall x F(x) \sim \forall x G(x)$;
6) если $\vdash F(x) \sim G(x)$, то $\exists x F(x) \sim \exists x G(x)$.

Первые четыре правила проверяются с помощью таблиц истинности.

С помощью этих шести правил теорема 4.7 доказывается индукцией по построению формулы $F(A)$: показывается, что если $A \sim B$, то эта эквивалентность сохраняется на всех шагах построения $F(A)$ из A и $F(B)$ из B.

Продемонстрируем это доказательство на примере.

Пусть $F(A) = \forall y (P_2(y) \vee \neg \exists P_2(x, y))$, $A = \neg \exists x P_2(x, y)$.

В качестве эквивалентности $A \sim B$ возьмем эквивалентность $\neg \exists x P_2(x, y) \sim \forall x \neg P_2(x, y)$, верную в логике предикатов и в силу теоремы 4.7 доказуемую в исчислении понятийных предикатов.

1. $\neg \exists x P_2(x, y) \sim \forall x \neg P_2(x, y)$ (исходная эквивалентность $A \sim B$).
2. $(P_1(y) \vee \neg \exists x P_2(x, y)) \sim (P_1(y) \vee \forall x \neg P_2(x, y))$ (правило 2).
3. $\forall y(\neg P_1(y) \vee \neg \exists x P_2(x, y)) \sim \forall y (P_2(y) \vee \forall x \neg P_2(x, y))$ (правило 5).

Формула из шага 3 и есть искомая эквивалентность $F(A) \sim F(B)$.

Приведем теперь без доказательства некоторые эквивалентности, выводимые в исчислении понятийных предикатов (в них A и B — формулы, не содержащие свободных вхождений x):

$$A \,\&\, \forall x F(x) \sim \forall x (A \,\&\, F(x)); \tag{4.5}$$
$$A \vee \exists x F(x) \sim \exists x (A \vee F(x)); \tag{4.6}$$
$$A \,\&\, \exists x F(x) \sim \exists x (A \,\&\, F(x)); \tag{4.7}$$
$$A \vee \forall x F(x) \sim \forall x (A \vee F(x)); \tag{4.8}$$
$$A \to \forall x F(x) \sim \forall x (A \to F(x)); \tag{4.9}$$
$$A \to \exists x F(x) \sim \exists x (A \to F(x)); \tag{4.10}$$
$$\forall x F(x) \to B \sim \exists x (F(x) \to B); \tag{4.11}$$
$$\exists x F(x) \to B \sim \forall x (F(x) \to B). \tag{4.12}$$

Эквивалентности (4.5)—(4.12), позволяют выносить кванторы вперед. Используя при этом соотношения, позволяющие заменять один квантор другим и «спускать» отрицание внутрь области действия квантора, а также правила переименования переменных (примеры 4.3, а, б), кванторы можно вынести вперед для любой формулы. Формула,

имеющая вид $Qx_1Q_2x_2 \ldots Q_nx_nF$, где $Q_1,\ldots,Q_n$ — понятийные кванторы, а F— формула, не имеющая понятийных кванторов (и являющаяся областью действия всех n понятийных кванторов), называется *предваренной формой,* или формулой в предваренной форме.

В исчислении понятийных предикатов для любой формулы F существует эквивалентная ей предваренная форма F', т. е. $\vdash F \sim F'$.

Пример 4.4. а. Приведем к предваренной форме формулу, которой была проиллюстрирована теорема 4.7.

1. $\forall y(P_1(y) \vee \neg \exists x P_2(x,y));$

2. $\forall y\,(P_1\,(y) \vee \forall x \neg P_2\,(x,y))$;

3. $\forall y \forall x\,(P_1\,(y) \vee \neg P_2\,(x,y))$ [по соотношению (4.8)].

б. Проделаем то же самое с несколько более сложной формулой:

1. $\forall x P_1\,(x) \to \neg \forall x\,(P_2\,(y) \vee \exists y P_3\,(x,y));$

2. $\forall x P_1\,(x) \to \neg \forall x\,(P_2\,(z) \vee \exists y P_3\,(x,\ y))$ (переименование свободной переменной).

3. $\forall x P_1\,(x) \to \neg \forall x \exists y\,(P_2\,(z) \vee P_3\,(x,y))$ [соотношение (4.6).

4,5. $\forall x P_1\,(x) \to \exists x \forall y \neg\,(P_2\,(z) \vee P_3\,(x,\ y))$ [здесь объединены два шага вывода].

6. $\forall u P_1\,(u) \to \exists x \forall y \neg\,(P_2\,(z) \vee P_3\,(x,y))$ (переименование связанной переменной).

7,8,9. $\exists x \exists u \forall y\,(P_1\,(u) \to \neg\,(P_2\,(z) \vee P_3\,(x,y))$ [здесь объединено три шага вывода: применения (4.10), (4.11), (4.9)].

Прикладные исчисления понятийных предикатов. Построенное ранее исчисление понятийных предикатов называется исчислением понятийных предикатов первого порядка. В исчислениях второго порядка возможны кванторы по предикатам, т. е. выражения вида $\forall P\,(P\,(x))$. Приложения таких исчислений встречаются гораздо реже; здесь исчисления второго порядка рассматриваться не будут.

Исчисление понятийных предикатов, не содержащее функциональных букв и предметных констант, называется чистым исчислением предикатов. По существу, до сих пор рассматривалось именно чистое исчисление предикатов, хотя язык исчисления (т. е. формулы) был определен с учетом его использования в прикладных исчислениях.

Прикладные исчисления (теории первого порядка) характеризуются тем, что в них к чисто логическим аксиомам добавляются собственные аксиомы, в которых, как правило, участвуют конкретные (индивидуальные) предикатные буквы (понятия), функциональные буквы и предметные константы. Типичные примеры индивидуальных

предикатных букв (понятий) — предикаты =, <, функциональных букв— знаки арифметических операций, предметных констант — натуральные числа, единица в теории групп, пустое множество в теории множеств.

Другая важная особенность прикладных исчислений заключается в том, что в схемах аксиом P1 и P2 участвуют уже не предметные переменные, а произвольные понятийные термы. Более точно, эти схемы аксиом принимают следующий вид:

P1'. $\forall xF(x) \rightarrow F(t)$;

P2'. $F(t) \rightarrow \exists xF(x)$,

где $F(t)$ — результат подстановки понятийного терма t в $F(x)$ вместо всех свободных вхождений x, причем все переменные t должны быть свободными в $F(t)$.

Большинство прикладных исчислений содержит предикат равенства = и определяющие его аксиомы. Аксиомами для равенства могут служить следующие:

El. $\forall x(x = x)$ (конкретная аксиома);

E2. $(x = y) \rightarrow (F(x, x) \rightarrow F(x, y))$ (схема аксиом), где $F(x, y)$ получается из $F(x, x)$ заменой некоторых (не обязательно всех) вхождений x на y при условии, что y в этих вхождениях также остается свободным. Теорию коммуникаций, в которой E1 и E2 являются теоремами или аксиомами, будем называть *теорией* (или исчислением) *с равенством.* Это связано с тем, что из E1 и E2 выводимы основные свойства равенства — рефлексивность, симметричность и транзитивность.

Теорема 4.8. В любой теории с равенством

1) $\vdash t = t$ для любого понятийного терма t;

2) $\vdash x = y \rightarrow y = x$;

3) $\vdash x = y \rightarrow (y = z \rightarrow x = z)$.

Доказательство:

1) Непосредственно следует из аксиом E1 и P1' (где $F(x)$ имеет вид $x = x$) по правилу заключения.

2) Из E2 имеем $(x = y) \rightarrow (x = x \rightarrow y = x)$. Отсюда $x = y,\ x = x \vdash y = x$ (двойное применение правила заключения). Но так как $\vdash x = x$, то $x = y \vdash y = x$ и, следовательно, по теореме дедукции $x = y \rightarrow y = x$.

(Теорема дедукции доказывалась ранее только для исчисления предписаний (для исчисления понятийных предикатов она верна лишь при некоторых ограничениях). Но данные формулы не содержат кванторов, поэтому эта теорема применима).

3) Поменяем местами в E2 x и y, в качестве $F(y, y)$ возьмем $y = z$, а в качестве $F(y, x)$ возьмем $x = z$. Получим $y = x \rightarrow (y = z \rightarrow x = z)$ и по правилу заключения $y = x \vdash (y = z \rightarrow x = z)$. Но так как в силу п.2

$x = y \vdash y = x$, то по транзитивности выводимости (см. п. 4.1) получаем $x = y \vdash (y = z \to x = z)$ и по теореме дедукции $x = y \to (y = z \to x = z)$.

Три свойства равенства, полученные этой теоремой, верны для любою отношения (и, следовательно, соответствующего предиката) эквивалентности. Схема аксиом Е2 выражает более сильное свойство, присущее лишь равенству: **неотличимость элементов, для которых выполняется равенство.**

Другой пример прикладного исчисления — теория частичного строгого порядка, содержащая две конкретные аксиомы для предиката <:

NE 1. $\forall x \neg (x < x)$;

NE 2. $\forall x \forall y \forall z \ (x < y \to (y < z \to x < z))$

Вызывает удивление, что два сходных утверждения о транзитивности даются в разной форме: п. 3 теоремы 4.8 без кванторов, в открытой форме, а NE2.— с кванторами, в замкнутой форме. В действительности эти два способа задания аксиом равносильны: от первого ко второму переходим по правилу обобщения, а от второго к первому — по аксиоме P1' и правилу заключения.

Если к NE1 и NE2 добавить аксиому с предметной константой NE3 $\forall x \neg (x < a)$, то получим теорию частичного порядка с минимальным элементом a.

Наиболее изученной формальной теорией, которая играет фундаментальную роль в основаниях математики, является формальная арифметика. Ее схемы аксиом:

Al. $F(0) \ \& \ \forall x (F(x) \to F(x')) \to F(x)$ (принцип индукции);

A2. $t'_1 = t'_2 \to t_1 = t_2$;

A3. $\neg (t'_1 = 0)$;

A4. $t_1 = t_1 \to (t_1 = t_3 \to t_2 = t_3)$;

A5. $t_1 = t_2 \to t'_1 = t'_2$;

A6. $t_1 + 0 = t_1$;

A7. $t_1 + t'_2 = (t_1 + t_2)'$;

A8. $t_1 \cdot 0 = 0$;

A9. $t_1 \cdot t'_2 = t_1 \cdot t_2 + t_1$.

В этих аксиомах использованы три функциональных символа +, •, ', один индивидуальный предикат (предикатная буква) = и одна предметная константа 0. Они придают схемам А2—А9 вполне конкретный вид: все понятийные предикатные и функциональные буквы в них зафиксированы и единственный способ их варьировать — это подставлять различные понятийные термы вместо метапеременных t_1, t_2. В частности, схемы А6—А9 — это просто предикат равенства, в который подставлены понятийные термы определенного вида. Схема

А1 имеет обычный вид: *F (x)* — метаобозначение, употребленное в том же смысле, в каком оно использовалось в чистом исчислении предикатов. Заметим, что схемы А2—А9 можно заменить конкретными аксиомами (т. е. формулами самой арифметики):
А2'. $x'_1 = x'_2 \rightarrow x_1 = x_2$ и т. д., из которых любые формулы вида А2—А9 можно получить с помощью правила обобщения и схемы аксиом Р1.
В заключение суммируем особенности построения прикладных исчислений понятийных предикатов и, в частности, посмотрим, в чем здесь проявляется различие между «схемно-аксиомным» и «подстановочным» подходами к построению исчислений.
1. При «схемно-аксиомном» подходе общелогические схемы аксиом, как уже отмечалось, описываются формулами в метапеременных; собственные аксиомы прикладных исчислений могут задаваться либо также схемами (например, А1), либо конкретными аксиомами, т. е. формулами самого исчисления (например, Е1). Однако в любом случае собственные аксиомы, как правило, содержат фиксированные функциональные и предикатные буквы (понятия), которые тем самым наделяются некоторыми свойствами, отличающими их от других букв исчисления (например, свойства предикатной буквы = определяются аксиомами Е1 и Е2).
2. При «подстановочном» подходе системы аксиом I (или II) и Р1, Р2 являются конечной системой формул самого исчисления. Буква *F* в аксиомах Р1 и Р2 — это предикатная буква исчисления (а не метапеременная, как в случае 1), являющаяся переменным предикатом, вместо которого по правилу подстановки подставляются формулы исчисления. В собственных аксиомах прикладных исчислений появляются постоянные предикаты (например, равенство), вместо которых подставлять ничего нельзя. Поэтому в языке исчисления предикатные и функциональные буквы приходится делить на две группы — переменные и постоянные, что не обязательно при первом подходе.

4.3. Метатеория логических исчислений в теории коммуникаций

Под метатеорией логических исчислений понимается изучение их общих свойств и соответствия этих свойств целям, ради которых исчисления создавались. Некоторые задачи такого рода (связь между доказуемостью и истинностью) уже рассматривались. Здесь они будут систематизированы и изучены более подробно.

Интерпретация и коммуникационные модели. Ценность формальной теории коммуникаций в конечном счете определяется ее способностью описывать какие-то объекты (энергии) и связи между ними. Поэтому один из первых для теории коммуникаций вопросов — это вопрос о том, для описания каких объектов пригодна данная теория коммуникаций. Конечно, если ставить его как общую проблему соответствия научных знаний о мире самому миру, то он не может обсуждаться средствами лишь математики (поскольку математика в отличие от естественных наук имеет дело не непосредственно с миром, а с формальными описаниями его различных фрагментов) и становится фундаментальной проблемой философии и методологии науки. Такого рода проблемы лежат за пределами этой работы. Здесь речь пойдет о более простом случае, когда множество объектов, для которого строится формальная теория коммуникаций, само по себе представляет достаточно строго описанный предмет исследований. Более точно, проблема адекватности формальной теории коммуникаций и описываемых ею объектов будет рассматриваться как математическая задача о соответствии между содержательно построенной теорией коммуникаций, рассматриваемой как множество объектов с операциями и отношениями на нем (т. е. как алгебраическая система —), и множеством высказываний об этой теории, построенном как формальное исчисление. При такой постановке задачи интуитивно ясное понятие интерпретации приобретает точный математический смысл.

Интерпретация формальной теории коммуникаций состоит из множества M и однозначного отображения, которое каждой предикатной букве (понятию) P_i^n ставит в соответствие n-местное отношение на M (интерпретирует ее как отношение на M), каждой функциональной букве f^n_j — n-местную операцию на M, каждой предметной константе — элемент M. Постоянные понятийные термы исчисления (не содержащие предметных переменных) при таком определении также отобразятся в элементы M. Таким образом, множество M (называемое областью интерпретации) рассматривается здесь как основное множество алгебраической системы.

Всякая замкнутая, т. е. не содержащая свободных переменных, формула теории коммуникаций представляет собой предписание о действиях над элементами, отношениями и функциями M, которое может быть истинным или ложным. Значения истинности составных формул вычисляются в соответствии с входящими в них логическими операциями. Открытая формула соответствует некоторому отношению на M, при подстановке предметных констант она превращается в предписание о том, что между элементами M, соответствующими

подставленным константам, должно выполняться данное отношение. Открытая формула называется *выполнимой* в данной интерпретации, если существует такая подстановка предметных констант, при которой она превращается в истинное предписание. Формула называется *истинной в данной интерпретации,* если она выполняется (т. е. превращается в истинное предписание) при любой подстановке констант. Формула называется ложной в данной интерпретации, если она невыполнима.

Интерпретация (а иногда область интерпретации *M)* называется *моделью* для множества формул Г, если любая формула Г истинна в данной интерпретации. Интерпретация называется *моделью теории T,* если она является моделью множества всех теорем теории *T,* т. е. если всякая формула, доказуемая в *T,* истинна в данной интерпретации. Если в *T* доказуема некоторая открытая формула *F* (которая, строго говоря, предписанием не является), то в модели теории *T* должны быть истинными все предписания, получающиеся из *F* всеми возможными подстановками констант на место свободных переменных формулы *F,* и, следовательно, должно быть истинно предписание $\forall x_1 \ldots \ldots \forall x_n F$, где $x_1, \ldots, x_n$— свободные переменные формулы *F*. Это обстоятельство вполне соответствует правилу обобщения в исчислении понятийных предикатов и использованию открытых аксиом (например, Е2) в прикладных исчислениях.

Пример 4.5. а. Для теории с равенством моделью является любая интерпретация, при которой предикатной букве (понятию) = поставлено в соответствие отношение равенства. Возможны и менее тривиальные интерпретации. Возьмем в качестве области интерпретации натуральный ряд *N,* в качестве интерпретации символа = — некоторое отношение *R* эквивалентности на *N* (например, сравнимость по mod 11), а все предикатные буквы теории (будем считать, что их конечное число) проинтерпретируем отношениями, которые не различают эквивалентные числа, т. е. по существу являются отношениями между классами эквивалентности. В данном конкретном случае такими отношениями будут отношения, сформулированные в терминах остатков от деления на 11, например: *«aR_1b,* если остаток от деления *a* на 11 меньше остатка от деления *b* на 11», *«a* обладает свойством R_2, если остаток от деления *a* на 11 не превосходит 5» и т. д. Значения истинности предписаний, содержащих только такие отношения, не будут меняться, если в них одно число заменить числом из того же класса эквивалентности, т. е. имеющим тот же остаток от деления на 11. Поэтому аксиома Е2 в такой интерпретации будет истинна, хотя интерпретация символа = не совпадает с обычным отношением равенства.

б. В теории строгого частичного порядка (с аксиомами NE1 и NE2) моделью будет любая интерпретация, при которой предикатная буква (понятие) < интерпретируется отношением «быть меньше». Однако почти столь же очевидно из аксиом NE1 и NE2 (хотя и выглядит парадоксально), что моделью этой теории будет и интерпретация, при которой символ < интерпретируется как отношение «быть больше»!

Последний пример иллюстрирует существо формального подхода: в символы исчисления (даже самые привычные) не вкладывается никакого смысла, пока не введена их явная интерпретация. Но и введенная интерпретация, вообще говоря, не относится к числу средств самого исчисления: она позволяет осмыслить формулы исчисления, но не участвует в формальном выводе теорем. О формальных свойствах самого исчисления, его формул и формальных преобразований над ними принято говорить, как о *синтаксисе исчисления;* свойства исчисления, описанные в понятиях его интерпоетаций, — это *семантика исчисления.* Например, мета-теоремы 4.1, 4.7, 4.8 являются теоремами о синтаксисе, а метатеоремы 4.2—4.6 — теоремами о семантике.

Непротиворечивость. Напомним, что формула называется *общезначимой,* если она истинна в любой интерпретации, и *противоречивой,* если она ложна в любой интерпретации, т. е. если ее отрицание общезначимо. Для любого множества общезначимых формул и, в частности, для чистого исчисления понятийных предикатов (по теореме 4.5) любая алгебраическая система и вообще любое множество является моделью. Напротив, для любого множества формул, содержащего хотя бы одну противоречивую формулу, моделей не существует.

Аксиома E1 теории с равенством не является общезначимой: существуют интерпретации (в смысле, указанном в начале этого параграфа), в которых она ложна. Примером такой интерпретации может служить всякое отображение, которое предикатной букве = ставит в соответствие отношение <. (Здесь символы = и < используются на разных уровнях: символ = — как формальная предикатная буква (понятие), не имеющая смысла до ее интерпретации, а символ < — как символ отношения на множестве, имеющий всем известный смысл «быть меньше».) Собственные аксиомы прикладных исчислений вообще не общезначимы — иначе по теореме 4.6 они были бы доказуемы в чистом исчислении предикатов.

Введенное ранее определение противоречивой формулы является семантическим, т. е. связывающим непротиворечивость с истинностью. Исходя из него, можно сформулировать понятие семантически непротиворечивой теории: **теория *семантически непротиворечива,***

если ни одна из ее теорем не является противоречивой, т. е. ложной в любой интерпретации. Исчисление предписаний и исчисление понятийных предикатов семантически непротиворечивы в силу теорем 4.3 и 4.5: все их теоремы общезначимы и, следовательно, непротиворечивы.

С помощью введенных понятий ответ на общий вопрос, поставленный в начале параграфа, формулируется так: **теория *T* пригодна для описания тех множеств, которые являются ее моделями; модель для теории *T* существует тогда и только тогда, когда *T* семантически непротиворечива.** Чисто логические теории — исчисление высказываний и исчисление предикатов — в силу теорем 4.3 и 4.5 пригодны для описания любых множеств, что соответствует общенаучному принципу универсальности законов логики (Лейбниц формулировал его как выполнимость логических законов «во всех мыслимых мирах»). Такой критерий пригодности теорий по существу известен уже давно; отыскание модели для теории до возникновения оснований математики было единственным общепризнанным методом доказательства законности теории. Одно из важных достижений оснований математики заключается в том, что аналог этого критерия сформулирован в понятиях самих формальных теорий, без привлечения семантических понятий. Таким аналогом является понятие формальной, или дедуктивной, непротиворечивости.

Теория *T* называется *формально непротиворечивой,* если не существует формулы *F*, такой, что *F* и ¬*F* являются теоремами теории *T,* т. е. в *T* не выводимы одновременно формула и ее отрицание. Аналогичное определение можно сформулировать и для произвольного множества формул.

Для теории коммуникаций, содержащей исчисление предписаний, из определения непосредственно следует, что если она формально противоречива, то в ней доказуема тождественно-ложная формула и, следовательно, она семантически противоречива. Действительно, если *F* и ¬*F* доказуемы, то, подставив в аксиому I 5 *F* вместо *A,* ¬ *F* вместо *B* и дважды применив правило заключения, получим |— *A* & ¬*A*. Более того, никакое множество формул, содержащее *F* и ¬ *F* (т. е. формально противоречивое), не может иметь модели, поскольку *F* и ¬ *F* не могут быть одновременно истинными ни в какой интерпретации. Тем самым доказано (от противного), что если множество формул семантически непротиворечиво, то оно формально непротиворечиво. Обратное утверждение (которое также оказывается верным), если понимать его конструктивно, гораздо глубже.

Смысл его в том, что по всякому формально непротиворечивому множеству формул можно построить его модель. Доказательство этого факта — его можно найти в литературе — довольно сложно и заключается в изложении метода такого построения. Вместе оба утверждения образуют важную метатеорему.

Теорема 4.9. Множество формул формально непротиворечиво, если и только если оно семантически непротиворечиво (т. е. имеет модель).

Таким образом, понятие формальной непротиворечивости оказывается эквивалентным более привычному в математике понятию семантической непротиворечивости; однако оно сформулировано в синтаксических понятиях и с конструктивной точки зрения более надежно (вспомним, что именно семантические трудности — парадоксы и привели к возникновению науки об основаниях математики и к концепциям формального подхода). Привыкнуть же к эквивалентности этих двух понятий в математике было нелегко. Неприятие современниками неевклидовых геометрий Лобачевского—Бойаи объясняется именно тем, что законность этих теорий обосновывалась отсутствием в них противоречий — аргументом, по существу совпадающим с современным понятием формальной непротиворечивости. Геометрические модели для этих теорий, доказывающие их семантическую непротиворечивость, были найдены позднее.

Полнота, разрешимость, аксиоматизируемость. Теперь можно приступить к обсуждению вопросов об адекватности формальных теорий, т. е. соответствии тем целям, ради которых они создавались. Пусть имеется, с одной стороны, содержательная теория *S*, сформулированная в семантических понятиях, т. е. совокупность истинных высказываний о некоторой алгебраической системе *M;* с другой стороны — формальная теория *T,* т. е. множество предписаний, выводимых из аксиом теории *T* с помощью правил вывода теории *T.* В каких случаях можно утверждать, что *T* является удовлетворительной формализацией S? Каковы признаки удовлетворительности формализации?

Очевидно, необходимым признаком является условие, чтобы *S* была моделью теории *T,* т. е. чтобы существовало отображение, при котором всякая теорема теории *T* отображается в истинное высказывание из *S*. Однако само по себе это условие явно недостаточно, иначе для формализации любой теории S хватило бы чистого исчисления предикатов: ведь для него любое множество является моделью. Исчерпывающей формализацией S будет служить такая теория *T,* для которой выполняется и обратное соответствие: каждое истинное высказывание теории *S* отображается в некоторую теорему теории *T.*

Теория *T* с таким свойством называется *полной* относительно *S*, а иногда — *адекватной S*.

Из теорем 4.3 и 4.4 следует, что исчисление предписаний полно относительно алгебры высказываний, а теоремы 4.5 и 4.6 образуют известную *теорему Геделя о полноте исчисления предикатов* относительно логики предикатов, т. е. множества всех общезначимых предикатных формул. Если для семантической (содержательной) теории S удается построить непротиворечивую и полную формальную теорию *T,* то S естественно назвать *аксиоматизируемой,* или *формализуемой,* теорией. Из предыдущих результатов следует, что **логика высказываний и логика предикатов аксиоматизируемы с помощью соответствующих исчислений**. Еще одна важная характеристика формальной теории — это ее *разрешимость.* Формальная теория *T* называется *разрешимой,* если существует алгоритм, который для любой формулы языка определяет, является она теоремой в *T* или нет, иначе говоря, если предикат «быть теоремой в теории *T*» общерекурсивен.

Теорема 4.10. Исчисление предписаний разрешимо.

Разрешающий алгоритм для формулы *F* исчисления предписаний заключается в вычислении значений *F* на всех наборах значений ее переменных. Ввиду полноты исчисления предписаний *F* является его теоремой, если и только если она истинна на всех наборах.

Теорема 4.11 (теорема Черча). Исчисление предикатов неразрешимо.

Несмотря на полноту исчисления предикатов, разрешающий алгоритм, связанный с вычислением значений истинности предикатных формул, построить не удается по причине бесконечности предметной области, которая приводит в общем случае к бесконечным таблицам истинности. Идея доказательства теоремы Черча (которое здесь не приводится состоит в том, чтобы в чистом исчислении предикатов описать предикат *Q (i, a, x):* «машина Тьюринга с номером *i,* будучи применена к исходным данным *a,* закончит вычисление в момент *x*». Предикат $\exists x Q\ (i, a, x)$ — это предикат остановки, а $\exists x Q\ (a, a, x)$ — предикат самоприменимости.

Отметим, что важный для приложений фрагмент исчисления предикатов разрешим: исчисление одноместных предикатов (т. е. исчисление, допускающее в формулах только одноместные предикатные буквы) разрешимо.

Еще тяжелее обстоит дело с формальной арифметикой, система аксиом которой приведена в конце 4.2.

Теорема 4.12 (первая теорема Геделя о неполноте в форме Клини). Любая формальная теория *T,* содержащая формальную арифметику, неполна: в ней существует (и может быть эффективно построена)

замкнутая формула *F*, такая, что ¬*F* истинно, но ни *F*, ни ¬F не выводимы в *T*.

Теорема 4.13 (вторая теорема Геделя о неполноте). Для любой непротиворечивой формальной теории *T*, содержащей формальную арифметику, формула, выражающая непротиворечивость *T*, недоказуема в *T*.

Таким образом, **арифметика и теория чисел оказываются неаксиоматизируемыми теориями**. В понятиях теории алгоритмов сформулированные ранее результаты можно резюмировать так:

1) множество всех истинных высказываний логики высказываний перечислимо и разрешимо;

2) множество всех истинных высказываний логики предикатов перечислимо (ввиду его полноты), но неразрешимо;

3) множество всех истинных высказываний арифметики неперечислимо: если бы для него существовала перечисляющая процедура (и, следовательно, соответствующая машина Тьюринга), то по ней можно было бы построить полную формальную теорию, **рассматривая правила перехода машины одной конфигурации к другой как правила вывода, процесс вычисления — как вывод, а результаты вычисления — как теоремы.**

Две знаменитые теоремы Геделя имеют важное методологическое значение. Из первой теоремы следует, что для достаточно богатых математических теорий не существует адекватных формализаций. Правда, любую неполную теорию *T* можно расширить, добавив, например, к ней в качестве новой аксиомы истинную, но не выводимую в *T* формулу. Однако по первой теореме Геделя новая теория *T* также будет неполна. Вторая теорема может быть истолкована как невозможность исследования метасвойств теории средствами самой формальной теории (опять невозможность самоприменимости!); иначе говоря, метатеория теории *T* для того, чтобы иметь возможность доказывать хотя бы непротиворечивость теории *T*, должна быть богаче *T*. По существу при этом ставится под сомнение первоначальная, «максималистская» программа финитного подхода: **нельзя построить математику как некоторую фиксированную совокупность средств, которые можно было бы объявить единственно законными и с их помощью строить метатеории любых теорий.**

С другой стороны, не следует истолковывать результаты Геделя (как это иногда делается, в основном среди непрофессионалов) как крах формального подхода. **Наличие алгоритмически неразрешимых проблем вовсе не бросает тень на теорию алгоритмов, а лишь сообщает «суровую правду» об устройстве мира, изучаемого этой**

теорией. Из этой правды не вытекает, что алгоритмический, конструктивный подход к решению проблемы непригоден; **хотя он чего-то и не может, но лишь потому, что этого не может никто**. Точно так же невозможность полной формализации содержательно определенных теорий — это не недостаток подхода или концепции, а объективный факт, не устранимый никакой концепцией. Формальный подход остается основным конструктивным средством изучения множеств высказываний (предписаний). Невозможность адекватной формализации теории означает, что надо либо искать формализуемые ее фрагменты, либо строить какую-то более сильную формальную теорию, которая, правда, снова будет неполна, но, быть может, будет содержать всю исходную теорию. В частности, методами, не формализуемыми в формальной арифметике, Гениен доказал непротиворечивость формальной арифметики.

4.4. Абстрактные формальные системы

Дальнейшие примеры формальных систем. Исторически формальные системы создавались с конкретной целью более точного обоснования методов построения математических теорий. Однако постепенно стало ясно, что на основе тех же принципов — **исходного набора аксиом, правил вывода и понятия выводимости** — можно описывать не только множества выражений, интерпретируемых как предписания, но и перечислимые множества объектов произвольного вида. Основы теории таких формальных систем были заложены Э. Постом.

(Начиная с этого момента понятие «теория» употребляется в обычном интуитивном смысле и не имеет прямого отношения к точному понятию «формальной теории», рассмотренному в предыдущих параграфах).

Эту теорию можно назвать абстрактной (хотя этот термин и не является общепринятым) или общей, так как она — в отличие от метатеории логических исчислений — не рассматривает свойства формальных систем относительно их конкретных интерпретаций, а изучает лишь их внутренние, синтаксические свойства.

Прежде чем перейти к самой теории, рассмотрим некоторые примеры формальных систем, не связанных с логическими интерпретациями.

Пример 4.6. а. Множество допустимых шахматных позиций можно описать как формальную систему, в которой **единственной аксиомой является начальная позиция, правилами вывода — правила игры, а теоремами — позиции, полученные по правилам игры из**

начальной. Однако эта идея требует уточнения**.** Пусть дана позиция, например, ее диаграмма или описание в шахматной нотации. Для того чтобы однозначно получить все позиции, достижимые из данной за один ход, недостаточно знать правила хода всех фигур, в том числе правила взятия и правило превращения пешки; может понадобиться информация, не содержащаяся явно в позиции: нужно знать: 1) чей ход; 2) ходил ли раньше король (если он стоит на своем начальном поле, позиция на этот вопрос не отвечает) — это нужно для определения допустимости рокировки; 3) не был ли последний ход ходом пешки через два поля — это нужно для определения взятия пешки на проходе. **Чтобы превратить множество шахматных позиций в формальную систему, нужно за позицию принять такое ее описание, которое в явном виде содержит ответы на все три вопроса**. Для исключения позиций, получающихся после взятия королей, надо ввести правила, запрещающие игнорировать шах, и понятие заключительных позиций (матовых и патовых), после которых никакие ходы не разрешаются.

б. Многие **индуктивные определения можно превратить в формальные системы, аксиомы которых перечислены в базисе (первом пункте) определения, а правила вывода — в индуктивном шаге.** Необходимым условием перехода к формальной системе является конструктивность задания аксиом и правил вывода, точнее, разрешимость множества аксиом и отношения непосредственной выводимости.

Рассмотрим, например, индуктивное определение абстрактной ориентированной двухполюсной схемы.

1) Пусть зафиксировано конечное число объектов, которые назовем элементами; в каждом элементе выделено двуполюса: вход и выход. Элементы являются схемами, полюсы которых совпадают с полюсами схем.

2) Пусть S_1 и S_2 — схемы. Тогда объекты, получающиеся:

а) путем отождествления выхода S_1 со входом S_2 (правило последовательного соединения) и

б) путем отождествления входа S_1 со входом S_2 и выхода S_1 с выходом S_2 (правило параллельного соединения), также являются схемами. В случае 2а входом схемы является вход S_1, выходом — выход S_2. В случае 2б входом является объединенный вход S_1, S_2, выходом — объединенный выход S_1, S_2.

Это определение описывает формальную систему, в которой **аксиомами являются элементы с выделенными полюсами, а правилами вывода — правила соединения схем.** Любой инженер даст этой системе обычную схемную (в инженерном смысле слова —

как правило, электротехническую) интерпретацию; однако такая интерпретация вовсе не вытекает из определения. В понятие элемента не вкладывается никакого содержания; единственное, что от него требуется — **возможность эффективно распознавать полюсы**. Элементом может быть ориентированное ребро, полюсы которого — вершины; тогда схемы— это просто графы с выделенным входом (источником, отправителем) и выходом (стоком, получателем, адресатом). (Вопрос к читателю: все ли такие графы порождаются данной формальной системой?) **Другая возможная интерпретация: элементы — одноместные функции; последовательное соединение—композиция функций, т.е. их последовательное вычисление, а параллельное соединение — параллельное вычисление двух функций с одинаковыми исходными данными и суммированием результатов в конце.**

При описании формальных систем в примере 4.6 была допущена одна вольность, которая, строго говоря, недопустима. Дело в том, что **ни в одном из этих описаний не зафиксирован алфавит; соответственно и правила вывода не сформулированы как операции над символами в словах.** Можно, конечно, сказать, что «примерно понятно, как это сделать; детали не уточняются», но способы уточнения могут быть различными и приведут к различным формальным системам. Это обстоятельство известно всем, кто занимался программированием игры в шахматы или машинными преобразованиями схем. Да и вообще любой, кто программировал содержательно сформулированные алгоритмы, знает, что **начинать приходится с выбора *формы представления данных,* т. е. символического кодирования объектов в понятиях языка программирования (а в случае автокода — просто машинными словами)**, причем различные выборы приводят к формальным системам, совершенно различным по своему виду и качествам. Этот выбор — предыстория формальной системы, возникающей лишь тогда, когда он уже сделан.

Общая теория формальных систем не рассматривает все возможные свойства конкретных систем, подобно тому, как теория алгоритмов не учит строить конкретные алгоритмы. Одной из ее главных целей является цель, в некотором смысле противоположная: выяснить, **каков необходимый минимум средств, с помощью которых можно описать любую формальную систему**. Сюда же примыкает вопрос, центральный для любой математической теории: каковы возможности объектов, изучаемых в данной теории? Для теории формальных систем он выглядит так: какие множества могут порождаться формальными системами? Опыт изучения теории алгоритмов говорит о том, что

такого рода задачи решаются путем выбора конкретных средств, приводящих к конкретным моделям, общность которых показывается затем путем сравнения их с другими моделями. **Средства формальных систем — это аксиомы и правила вывода**. В абстрактных формальных системах, в отличие от логических исчислений, не выделяется понятия формулы («осмысленного выражения»); их объекты — произвольные слова в фиксированном алфавите.

Итак, *формальная система FS* определяется:

1) алфавитом A (множество всех слов в алфавите A обозначим A^*);

2) разрешимым множеством $A_1 \subseteq A^*$, элементы которого называются аксиомами;

3) конечным множеством вычислимых отношений R_i (α_1,... ..., α_n, β) на множестве A^*, называемых правилами вывода (аксиомы также можно задать правилом вывода (одноместным отношением): R_l (β) ~ «β» — аксиома»); слово β называется выводимым из α_1,... ..., α_n по правилу R,.

Понятия вывода, выводимости и доказательства — те же, что и для формальных теорий (см. 4.1).

Иногда конкретное множество аксиом в системе не фиксируется, а рассматривается выводимость из произвольных разрешимых множеств слов; **в этом случае понятия вывода и доказательства не различаются. Конкретные виды формальных систем определяются в основном видом правил вывода**. Здесь будут рассмотрены два способа представления формальных систем: системы подстановок и системы продукций Поста. Однако уже общего определения формальной системы оказывается достаточным, чтобы получить простой, но важный факт.

Теорема 4.14. Для любой формальной системы FS множество всех доказуемых в ней слов перечислимо.

Рассмотрим множество A^{**} всех конечных последовательностей α_1,... ..., α_n, где α_i — слова в алфавите A. Множество A^{**}, очевидно, перечислимо. Ввиду разрешимости множества аксиом и вычислимости правил вывода по любой последовательности α_1,... ..., α_n можно эффективно узнать, является она выводом в FS или нет. Поэтому если в процессе перечисления A^{**} выбрасывать все последовательности α_1,... ..., α_n, не являющиеся выводами, то получим перечисление множества выводов. Следовательно, множество последних слов выводов, т. е. слов, выводимых в FS, перечислимо.

Отметим, что описанная процедура перечисляет выводимые слова, вообще говоря, с повторениями, поскольку одно и то же выводимое слово может иметь много выводов (например, **любая**

последовательность, заканчивающаяся аксиомой, есть вывод этой аксиомы).

Верно ли обратное — можно ли для перечислимого множества M построить перечисляющую его формальную систему, т. е. систему, в которой множество выводимых слов совпадает с M? Из предварительных соображений можно ответить утвердительно: представляется, например, правдоподобным, что машину Тьюринга можно проинтерпретировать как формальную систему. Точный ответ на этот вопрос будет дан при рассмотрении конкретных видов формальных систем.

Системы подстановок. *Система подстановок,* или *полусистема Туэ* — это формальная система, определяемая алфавитом A и конечным множеством подстановок вида $\alpha_i \to \beta_i$, где α_i, β_i — слова, возможно, пустые, в A. Подстановка $\alpha_i \to \beta_i$ интерпретируется как правило вывода R_i следующим образом: $\gamma \models \delta$ по правилу R_i, если слово δ получается из γ путем подстановки слова β_i вместо какого-нибудь вхождения α_i в слово γ. Выводом α из β в системе подстановок называется цепочка $\alpha \models \varepsilon_1 \models \varepsilon_2 \models \ldots \models \beta$, где каждое ε_j получается из ε_{j-1} некоторой подстановкой; как и обычно, наличие выводимости обозначаем $\alpha \vdash \beta$.

Такое определение вывода отличается от определения в начале п.4.1; предоставляем читателю убедиться, что для любой формальной системы, где все правила — бинарные и унарные отношения, эти два определения совпадают, т. е. $\alpha \vdash \beta$ в первом смысле, если и только если $\alpha \vdash \beta$ во втором смысле. (Заметим, что правило заключения в логических исчислениях — тернарное отношение!)

Зафиксируем множество аксиом системы и назовем слово δ заключительным, если оно доказуемо в системе и к нему неприменима ни одна из подстановок, т. е. если никакое его подслово не является левой частью никакой подстановки. Системе команд $\sum_T$ машины Тьюринга T нетрудно поставить в соответствие систему подстановок S_T над конфигурациями. Если в качестве аксиом выбрать слова $q_1\alpha$, то заключительными словами в системе S_T будут слова $q_z\beta$ (α, β — слова в алфавите ленты машины T). Если же к S_T добавить подстановку $q_z \to \lambda$, то в полученной системе S'_T множество заключительных слов совпадает с множеством значений функции, вычисляемой машиной T. Это дает для систем подстановок теорему, обратную теореме 4.14.

Теорема 4.15. Для любого перечислимого множества M существует система подстановок, множество заключительных слов которой совпадает с M.

Ассоциативное исчисление, или *система Туэ*— это формальная система, определяемая алфавитом A и конечным множеством

соотношений $\alpha_i \leftrightarrow \beta_i$, каждое из которых понимается как пара подстановок $\alpha_i \rightarrow \beta_i$ (левая подстановка) и $\beta_i \rightarrow \alpha_i$ (правая подстановка). Таким образом, ассоциативное исчисление всегда есть система подстановок; обратное, вообще говоря, неверно. При наличии таких парных правил вывода, если $\alpha \vdash \beta$, то и $\beta \vdash \alpha$; ввиду отмеченной ранее транзитивности выводимости получаем, что в ассоциативном исчислении выводимость является отношением эквивалентности и разбивает множество всех слов в A на классы эквивалентности. Заключительные слова в ассоциативных исчислениях возможны, однако они соответствуют классам эквивалентности, состоящим из одного слова, и особого интереса не представляют.

Аналогично тому, как это делалось для систем подстановок, поставим в соответствие каждой машине Тьюринга T (будем считать, что T работает на правой полуленте) ассоциативное исчисление $S(T)$. Опишем это соответствие более подробно. Если A_T — алфавит ленты машины T, то $A_S = A_T \cup \{q_1, \ldots, q_z\}$. Системе команд машины T соответствует система соотношений в S (T) (слева — команды, справа — соотношения):

$$q_i a_j \rightarrow q_k a_l R;\ q_i a_j \leftrightarrow a_l q_k; \qquad (4.13)$$

$$q_i a_j \rightarrow q_k a_l L;\ a_l q_i a_j \leftrightarrow q_k a_t a_l \text{ для всех } a_t \in A_T \qquad (4.14)$$

(число соотношений, соответствующих одной команде с движением влево, равно $|A_T|$).

Теорема 4.16. В исчислении S (T) слова $\alpha q_i a_j \beta$ и $\gamma q_z a_k \delta$ эквивалентны, если и только если машина (система) T из конфигурации $\alpha q_i a_j \beta$ за конечное число тактов переходит в конфигурацию $\gamma q_z a_k \delta$.

Одна половина теоремы («если») непосредственно следует из доказательства теоремы 4.15. Вторая половина («только если») может показаться несколько неожиданной: ведь в $S(T)$ допускаются подстановки в обе стороны, а в T— в одну и, следовательно, возможности $S(T)$ выглядят более сильными. Тем не менее покажем, что если существует вывод $\alpha q_i a_j \beta \vdash \gamma q_z a_k \delta$, то машина T из конфигурации $\alpha q_i a_j \beta$ переходит в конфигурацию $\gamma q_z a_k \delta$. Рассмотрим этот вывод, т. е. цепочку $\alpha q_i a_j \beta \vdash \varepsilon_1 \vdash \ldots \vdash \gamma q_z a_k \delta$. Каждое слово в этой цепочке получено из предыдущего либо левой, либо правой подстановкой. Кроме того, символ q в каждом слове — единственный. Последнее слово не может быть получено правой подстановкой, так как символ q_z отсутствует в левых частях команд машины T. Пусть ε_l — последнее слово в цепочке, полученное правой подстановкой:

$\varepsilon_l = \alpha_l q_r a_s \beta_l$. Слово ε_l получено из ε_{l-1} правой подстановкой, т. е. применением одного из соотношений (4.13), (4.14), содержащим $q_r a_s$ в левой части. Но таких соотношений столько, сколько команд T с $q_r a_s$ в левой части, т. е. ровно одно; обозначим его E_{rs}. Слово ε_{l+1} получено из

ε_l, левой подстановкой (по условию для ε_l); но единственная левая подстановка, применимая к ε_l — это то же соотношение E_{rs} (точнее, его левая половина). Итак, к ε_{l-1} применено E_{rs} слева направо, а к результату этого применения e_Γ применено то же E_{rs} справа налево. Так как все подстановки (4.13), (4.14) содержат q, а в любом слове цепочки q единственно, то любая подстановка $\alpha_i \to \beta_i$ к любому слову ε может быть применена единственным образом, т. е. в ε найдется не более чем одно вхождение α_i. Отсюда $\varepsilon_{l-1}=\varepsilon_{l+1}$; поэтому ε_l и ε_{l+1} из цепочки можно выбросить и построить вывод

$$\alpha q_i a_j \beta \vdash \ldots \vdash \varepsilon_{l+1} \vdash \varepsilon_{l+2} \vdash \ldots \gamma q_z a_k \delta,$$

в котором слов, полученных правой подстановкой, на единицу меньше, чем в прежнем выводе. Применяя к новому выводу те же рассуждения, из него также можно удалить слово, полученное правой подстановкой; в конечном счете будет построен вывод $\alpha q_i a_j \beta \to \ldots \to \gamma q_z a_k \delta$, содержащий только левые подстановки, т. е. в точности воспроизводящий последовательность конфигураций машины T.

Теорема 4.17 (теорема Маркова — Поста). Существует ассоциативное исчисление, в котором проблема распознавания эквивалентности слов алгоритмически неразрешима.

Возьмем какую-нибудь универсальную машину Тьюринга U, которая является правильно вычисляющей, т. е. начинает с конфигураций вида $q_1\alpha$ и заканчивает работу конфигурациями вида $q_z\beta$. Для машины U проблема остановки неразрешима (иначе ввиду универсальности U была бы разрешима общая проблема остановки для машин Тьюринга). Построим ассоциативное исчисление $S\ (U)$ и присоединим к нему систему соотношений вида $q_z a_i \leftrightarrow q_z$; получим новое исчисление $S'\ (U)$. В этом исчислении, так же как и в $S\ (U)$, можно имитировать вычислительный процесс U, однако благодаря новым соотношениям все заключительные конфигурации U в $S'\ (U)$ эквивалентны q_z. Поэтому теорема 4.16 для $S'\ (U)$ имеет следующий вид: в $S\ (U)$ слова $q_1\alpha$ и q_z эквивалентны, если и только если U, начав с $q_1\alpha$, остановится. Ввиду неразрешимости проблемы остановки для U проблема эквивалентности $q_1\alpha$ и q_z также неразрешима.

Ассоциативные исчисления имеют важную алгебраическую интерпретацию. Ранее говорилось о том, что всякую полугруппу можно получить из свободной полугруппы (т. е. просто из множества A^* всех слов в алфавите A) введением определяющих соотношений, т. е. равенств $\alpha_i=\beta_i$. Замена подслова α_i в слове α равным ему полсловом β_i, т. е. переход от слова $\alpha'\alpha_i\alpha''$ к слову $\alpha'\beta_i\alpha''$, называется эквивалентным преобразованием слова α. Слова считаются равными (или эквива-

лентными), если они могут быть получены друг из друга цепочкой эквивалентных преобразований.

Содержательно эквивалентность слов означает, что они задают один и тот же элемент полугруппы; иначе говоря, элементами полугруппы, заданной определяющими соотношениями, являются классы эквивалентности слов, порождаемые этими соотношениями. Формально такое описание полугруппы — это просто ассоциативное исчисление с соотношениями $\alpha_i \rightarrow \beta_i$; эквивалентные преобразования в полугруппе — это выводы в исчислении. Ранее была сформулирована проблема эквивалентности слов в полугруппе. Теперь становится ясным, что ответ на нее — это просто переформулировка теоремы 4.17.

Теорема 4.17'. Существует полугруппа, заданная определяющими соотношениями, в которой проблема распознавания эквивалентности (равенства) слов алгоритмически неразрешима.

Г С. Цейтин нашел ассоциативное исчисление с неразрешимой проблемой равенства — с алфавитом из пяти букв *{a, b, c, d, e}* и семью соотношениями:

$$ac \leftrightarrow ca;\ ad \leftrightarrow da;\ bc \leftrightarrow cb;\ bd \leftrightarrow db;$$
$$abac \leftrightarrow abace;\ eca \leftrightarrow ae;\ edb \leftrightarrow be.$$

Теорема 4.17' явилась **первым примером доказательства алгоритмической неразрешимости проблемы, не связанной с логикой и основаниями математики.**

Канонические коммуникационные системы.

Каноническая коммуникационнвя система определяется собственным алфавитом $A=\{a_1, ..., a_m\}$ (алфавитом констант), алфавитом $X=\{x_1, ..., x_n\}$ переменных, конечным множеством аксиом и конечным множеством правил вывода, имеющих вид $\gamma_1, ..., \gamma_k \Rightarrow \delta$ и называемых *продукциями* ($\gamma_1, ..., \gamma_k$ называются посылками (причинами), δ — следствием). Аксиомы, а также посылки и следствия продукций — это слова в алфавите $A \cup X$; иначе говоря, они содержат кроме собственных букв системы еще и переменные. В дальнейшем слова в алфавите $A \cup X$ будем называть понятийными термами, а слова в A — просто словами. Переменные канонической коммуникационной системы аналогичны метапеременпым в логических исчислениях; аксиомы канонической коммуникационной системы — это, по существу, схемы аксиом логических исчислений. Говоря о подстановке в понятийные термы слов вместо переменных, мы всегда будем иметь в виду, что сохраняется обычное ограничение: вместо всех вхождений одной и той же переменной подставляется одно и то же слово (быть может, пустое).

Слово α называется применением аксиомы ω, если α получается подстановкой в ω слов вместо переменных. Слово α непосредственно выводимо из слов α_1, ..., α_n, применением продукции P с n посылками, если найдется такая подстановка слов вместо переменных P, которая посылки P превратит в слова α_1, ..., α_n, а заключение P — в слово α. Например, слово bb непосредственно выводимо из слов $acab$, $cabb$ применением продукции $ax_1b, x_1bx_2 \Rightarrow bx_2$ при подстановке $x_1 = ca$, $x_2=b$. Последовательность слов называется доказательством в канонической системе, если каждое слово этой последовательности — либо применение аксиомы, либо непосредственно выводимо из предыдущих слов последовательности применением некоторой продукции системы. Слово (понятие) называется доказуемым (или теоремой), если оно является последним словом (понятием) некоторого доказательства.

Пример 4.7.а. Множество всех нечетных чисел в унарном представлении — это множество всех теорем канонической системы с алфавитами {|}, {x}, аксиомой | и продукцией $x\rightarrow\|$. Если эту продукцию заменить продукцией $x \rightarrow xx$, то получим каноническую систему, порождающую степени двойки (в унарном представлении):
|, ||, ||||, ...

б. Множество всех формул исчисления предписаний порождается канонической системой с алфавитами $\{a_u ..., a_n, \vee, \&, \neg, \rightarrow, (,)\}, \{x_1, x_2\}$, аксиомами $a_1, ..., a_n$ и продукциями

$$\left.\begin{array}{l} x_1,\ x_2 \Rightarrow (x_1 \vee x_2); \\ x_1,\ x_2 \Rightarrow (x_1 \,\&\, x_2); \\ x_1,\ x_2 \Rightarrow (x_1 \rightarrow x_2); \\ x_1 \Rightarrow \neg x_1. \end{array}\right\} \qquad (4.15)$$

Отметим, что здесь алфавит пропозициональных букв конечен. Для построения исчисления с бесконечным множеством пропозициональных букв сначала нужно построить формальную систему, порождающую это множество из конечного алфавита символов. Именно с этой целью в языках программирования вводится индуктивное определение идентификатора; оно представляет собой формальную систему, порождающую бесконечное множество переменных языка из конечного алфавита букв и цифр.

При построении машин Тьюринга часто оказывалось удобным (а иногда необходимым) вводить вспомогательные символы, которые участвуют в процессе вычисления, но не присутствуют ни в исходных данных, ни в результате. Аналогичные средства вводятся и в формальных системах.

Пусть дана каноническая система S с собственным алфавитом *A'*, в котором выделен подалфавит *A*. Если нас интересуют только те теоремы *S*, которые являются словами в *A*, то будем говорить, что система S является *канонической системой над A*. *A* назовем основным алфавитом, *A'* — расширением *A*, а символы из *A'\A* — вспомогательными.

Пример 4.8. а. Последовательность чисел 0, 1, 1, 2, 3, 5, 8..., в которой каждый член, начиная с третьего, равен сумме двух предыдущих, называется последовательностью Фибоначчи, а ее элементы— числами Фибоначчи. Числа Фибоначчи в унарном представлении порождаются канонической системой над {|} со вспомогательным символом *, аксиомой *1 и двумя продукциями:

$$x_1 * x_2 \Longrightarrow x_2 * x_1 x_2; \quad x_1 * x_2 \Longrightarrow x_1.$$

В этой системе символ * служит маркером, разделяющим два числа, аксиома задает первые два числа последовательности (0 изображен пустым словом слева от маркера), первая продукция из *n*-го и *(n* + 1)-го члена получает (*n* + 1)-й и *(n* + 2)-й члены последовательности, вторая продукция из пары чисел выделяет первое; только применение второй продукции дает слова в алфавите {|}.

б. Реализуем изложенный в примере 4.7б план построения формул исчисления предписаний с бесконечным множеством пропозициональных букв. Эти «буквы» будут обозначаться символом *a* с числовым индексом, т. е. будут представлять собой слова вида *a, a*01, *a*523 и т. д., играющие роль идентификаторов (имен переменных). Система содержит основной алфавит $A=\{a, 0, 1, 2, ..., 9, \vee, \&, \neg, \rightarrow, (,)\}$. вспомогательные символы *{i, f}*, аксиому *ai* и 16 продукций:

$$x_1 i \Longrightarrow x_1 0 i;$$
$$x_1 i \Longrightarrow x_1 1 i;$$
$$\cdots\cdots$$
$$x_1 i \Longrightarrow x_1 9 i,$$
$$x_1 i \Longrightarrow x_1 f;$$
$$x_1 f,\ x_2 f \Longrightarrow (x_1 \vee x_2) f;$$
$$x_1 f,\ x_2 f \Longrightarrow (x_1 \,\&\, x_2) f;$$
$$x_1 f,\ x_2 f \Longrightarrow (x_1 \rightarrow x_2) f;$$
$$x_1 f \Longrightarrow \neg x_1 f;$$
$$x_1 f \Longrightarrow x_1.$$

Первые десять продукций порождают идентификаторы; следующие пять продукций формализуют индуктивное определение формулы (из этих пяти четыре последних отличаются от продукций (4.15) только

символом *f),* последняя продукция служит для удаления символа *f.* Вспомогательные символы играют здесь несколько непривычную роль они, по существу, являются признаками (или метками) определенного класса слов: *i*— метка класса идентификаторов, *f*'— метка класса формул. Благодаря этому формальный вид продукции очень близок к тексту соответствующей части индуктивного определения: 11-я продукция означает «всякий идентификатор есть формула», а 12-я: «если x_1 и x_2 — формулы, то $(x_1 \vee x_2)$ — формула». Прием использования в формальных системах специальных символов как признаков классов широко используется в формальных системах, которые в данной работе не рассматриваются. Более подробно об использовании формальных систем такого рода для описания алгоритмических языков говорится в ряде работ.

Связь канонических систем с системами подстановок довольно ясна, по крайней мере, в одну сторону; очевидно, что всякой подстановке $\alpha_i \rightarrow \beta_i$ соответствует продукция $x_1\alpha_i x_2 \Rightarrow x_1\beta_i x_2$, поэтому для всякой системы подстановок легко построить эквивалентную ей каноническую систему. Соответствие в обратную сторону менее очевидно; однако о том, что оно существует, можно заключить, сопоставив теорему 4.14 (из которой следует, что множество теорем канонической системы перечислимо) с теоремой 4.15 (любое перечислимое множество можно представить как множество заключительных слов в некоторой системе подстановок). Правда, понятие заключительного слова может показаться несколько искусственным и слишком уж приближающим формальные системы к машинам Тьюринга; если рассматривать формальную систему как генератор элементов перечислимого множества, более естественным было бы описание перечислимого множества как множества теорем некоторой формальной системы. Это нетрудно сделать, используя введенное ранее понятие канонической системы с расширенным алфавитом.

Теорема 4.18. Для любого перечислимого множества M слов в алфавите A существует каноническая система над A, множество теорем которой совпадает с M.

Пусть M перечисляется машиной Тьюринга T с алфавитом ленты $A_T = \{ a_1, ..., a_n \}$, первые m букв которого образуют алфавит исходных данных $(m \leq n)$, множеством состояний $Q = \{ q_1, ..., q_z \}$ и системой команд $\sum_T$. Определим каноническую систему S над A следующим образом: $A = A_T$, $A' = A_T \bigcup Q_T$; аксиома системы *; командам машины T поставим в соответствие продукции аналогичнотому, как это делалось при доказательстве теорем 4.15 и 4.16: например, команде

$q_i a_j \to q_k a_l R$ соответствует продукция $x_1 q_i a_j x_2 \Rightarrow x_1 a_l q_k x_2$ [ср. (4.13)]. Кроме того, введем следующие $m + 2$ продукции:

$$x_1 * \Rightarrow x_1 a_1 *;$$
$$\vdots$$
$$x_1 * \Rightarrow x_1 a_m *;$$
$$x_1 * \Rightarrow q_1 x_1;$$
$$q_z x_1 \Rightarrow x_1.$$

Первые m продукций порождают из аксиомы все множество слов в алфавите исходных данных с маркером в конце; $(m + 1)$-я продукция (стартовая) порождает из любого исходного слова начальную конфигурацию; после этого работают продукции, соответствующие командам машины T; наконец, в заключительных конфигурациях (и только в них!) символ состояния выбрасывается и получаются искомые теоремы в алфавите A.

Итак, в понятиях канонических систем можно описать любые перечислимые множества. Правда, используемое определение канонической системы (в частности, вид ее продукций) является слишком общим: применение продукции трудно назвать элементарным шагом (хотя свойство применимости продукции, как нетрудно видеть, разрешимо). Поэтому возникает задача нормализации канонических коммуникационных систем, т. е. придание им более простого, «нормального» вида.

Каноническая коммуникационнпя система называется *нормальной,* если она имеет одну аксиому, а все ее продукции имеют видах $\alpha x \to x \beta$. Применение такой продукции к слову (если оно возможно) просто и стандартно: у слова вычеркивается начальный отрезок α и к концу приписывается β.

Теорема 4.19 (теорема Поста о нормальной форме). Для любой канонической системы CS с алфавитом A существует нормальная каноническая система NS над A, эквивалентная CS (т. е. множество теорем NS над A и множество теорем CS совпадают).

Прямое доказательство этой теоремы, содержащее метод построения NS по CS, довольно сложно. Ограничимся косвенным доказательством существования NS, а именно: покажем, что для любой системы подстановок S в алфавите A сущесшует нормальная каноническая система NS над A, эквивалентная S.

Пусть $A = \{a_1, ..., a_n\}$ и S содержит k подстановок $\alpha_i \to \beta_i$. Определим NS как систему с расширенным алфавитом $A' = \{ a_1, ..., a_n, a'_1, ..., a'_n)$ и $k + 2n$ продукциями:

$$\alpha_i x \Rightarrow x \beta'_i \qquad (i = 1, ..., k); \tag{4.16}$$

$$a_j x \Rightarrow x a'_j \quad (4.17)$$

$$a'_j x \Rightarrow x a_j, \quad (4.18)$$

где β'_i здесь и в дальнейшем обозначает слово β_i, в котором все буквы из A заменены соответствующими буквами со штрихом

Пусть к слову γ в S применима подстановка $\alpha_i \rightarrow \beta_i$: $\gamma = \gamma_1 \alpha_i \gamma_2$ и в S $\gamma_1 \alpha_i \gamma_2 \vdash \gamma_1 \beta_i \gamma_2$. Но тогда в NS

$$\gamma_1 \alpha_i \gamma_2 \vdash \alpha_i \gamma_2 \gamma'_1 \vdash \gamma_2 \gamma'_1 \beta'_i \vdash \gamma'_1 \beta'_i \gamma'_2 \vdash \gamma_1 \beta_i \gamma_2.$$

Так как любой вывод $\gamma \vdash \delta$ в S есть цепочка подстановок, то по выводу $\gamma \vdash \delta$ в S можно построить вывод $\gamma \vdash \delta$ в NS.

Пусть теперь в NS имеется вывод $\gamma \vdash \delta$, где γ, δ — слова в A. Разобьем этот вывод на отрезки $\gamma \vdash \varepsilon_{i_1} \vdash \varepsilon_{i_2} \vdash \delta$, такие, что ε_{i_j} и $\varepsilon_{i_{j+1}}$ (для всех j) — слова в A, а все слова между ними содержат вспомогательные буквы, и рассмотрим для определенности первый из них $\gamma \vdash \varepsilon_{i_1}$. Если γ — слово в A, то продукциями (4.17), (4.18) из него можно получить лишь слова вида $\gamma_2 \gamma'_1$, $\gamma'_2 \gamma_1$, γ', где слова γ_1, γ_2 таковы, что $\gamma_1 \gamma_2 = \gamma$. Так как ε_{i_1} — слово в A, то либо $\varepsilon_{i_1} = \gamma$ (но тогда этот отрезок из вывода можно удалить), либо на отрезке $\gamma \vdash \varepsilon_{i_1}$ была применена по крайней мере одна из продукций (4.16). Для любого вывода, содержащего между словами из A несколько применений продукций (4.16), в NS можно построить эквивалентный вывод, в котором между этими применениями вставлены слова из A. Строгое доказательство этого утверждения опускаем; приведем лишь пример: выводу $\alpha_1 \alpha_2 \vdash \alpha_2 \beta'_1 \vdash \beta'_1 \beta'_2 \vdash \beta'_2 \beta_1 \vdash \beta_1 \beta_2$, в котором применены продукции $\alpha_1 x \Rightarrow x \beta_1$ и $\alpha_2 x \Rightarrow x \beta_2$, соответствует вывод $\alpha_1 \alpha_2 \vdash \alpha_2 \beta'_1 \vdash \beta'_1 \alpha_2 \vdash \beta_1 \alpha_2 \vdash \alpha_2 \beta'_1 \vdash \beta'_1 \beta'_2 \vdash \beta_1 \beta_2$, в котором они разделены словом $\beta_1 \alpha_2$. Итак, можно считать, что в выводе $\gamma \vdash \varepsilon_{i_1}$ продукция (4.16) — пусть это будет $\alpha x \Rightarrow x \beta'$ —применена ровно один раз. Но тогда $\gamma = \gamma_1 \alpha \gamma_2$, применению продукции (4.-16) предшествовало несколько (быть может, ни одного) применений (4.17) и место вывода, где применена продукция (4.16), имеет вид $\alpha \gamma_2 \gamma_1' \Rightarrow \gamma_2 \gamma_1 \beta'$. Остаток $\gamma_2 \gamma_1 \beta'_1 \vdash \varepsilon_{i_1}$ рассматриваемого отрезка не содержит применений (4.16), поэтому в силу свойств продукций (4.17), (4.18) $\varepsilon_{i_1} = \gamma_1 \beta \gamma_2$. Но тогда ε_{i_1} получается из γ подстановкой $\alpha \rightarrow \beta$, т. е. $\gamma \vdash \varepsilon_{i_1}$ в S. Индукцией по числу слов ε_{i_j} доказываем, что $\gamma \vdash \varepsilon_{i_1}$ в S. Отсюда следует эквивалентность S и NS, после чего о справедливости теоремы нетрудно заключить из сопоставления теорем 4.15 и 4.18.

В заключение раздела приведем без доказательства один результат об алгоритмической неразрешимости, который доказан Постом с помощью канонических систем и который часто используется для доказательства других неразрешимостей (особенно в теории формальных грамматик).

Пусть дано конечное множество $(\alpha_1, \beta_1), ..., (\alpha_m, \beta_m)$ пар слов в алфавите *A*. Поставим следующую проблему: существует ли последовательность $i_1, i_2, ..., i_N$ индексов, такая, что $\alpha_{i_1}\alpha_{i_2}...\alpha_{i_N} = \beta_{i_1}\beta_{i_2}...\beta_{i_N}$?

Эта проблема называется комбинаторной проблемой или проблемой соответствия Поста. Имеются два варианта ее формулировки: общая комбинаторная проблема Поста (для произвольного множества пар) и ограниченная комбинаторная проблема (для множества пар с фиксированной мощностью *m)*.

Теорема 4.20. Ограниченная комбинаторная проблема Поста для достаточно больших *m* алгоритмически неразрешима.

Отсюда следует неразрешимость и общей комбинаторной проблемы.

Формальные системы и алгоритмы. Формальные системы оказываются столь же мощным средством для задания коммуникационных объектов, что и алгоритмы. С их помощью можно имитировать поведение машин Тьюринга, т. е. строить формальные коммуникационные системы, в некотором смысле аналогичные алгоритмам. С другой стороны, понятие перечислимого множества в понятиях формальных систем (опирающееся на теорему 4.18) выглядит более компактным, чем в понятиях машины Тьюринга. Поэтому возможны две концепции построения системы основных понятий, формализующих идеи эффективности и конструктивности коммуникационных систем. Концепция, описанная ранее и являющаяся исторически первой, кладет в основу понятие алгоритма. Вторая концепция, созданная Э. Постом, опирается на понятия формальной (конкретнее, канонической) системы и перечислимого множества, которое определяется просто как множество теорем формальной системы. Нормальную каноническую коммуникационную систему над алфавитом *A* можно представить как граф с одной выделенной вершиной — аксиомой, остальные вершины которого помечены словами в *A* — теоремами, ребра — это применения продукций, а пути из выделенной вершины в данную — возможные выводы данного слова. Множество слов в *A,* порождаемое системой, — это множество всех вершин графа, помеченных словами в *A*. **Алгоритм—это формальная система особого, детерминированного вида, характеризующаяся тем, что в ней к каждой теореме применима не более чем одна продукция.** Соответствующий граф

представляет собой цепочку, изображающую коммуникационный процесс; аксиома в таком графе — это просто исходные данные алгоритма.

Другой способ детерминизации формальных систем — это фиксация порядка применения правил вывода. Например, нормальный алгоритм Маркова — это упорядоченная система подстановок с двумя дополнительными соглашениями: 1) *i*-я подстановка может быть применена, только если неприменимы 1, ..., *(i*—1)-я подстановки;

2) если подстановка $\alpha \rightarrow \beta$ применима к слову γ, то β подставляется вместо самого левого вхождения α в γ.

Основной акцент «алгоритмической» концепции —в ее детерминизме. Поэтому она удобна при описании коммуникационных процессов и устройств. Основной акцент «формально-системной» концепции — в компактности конструктивного описания множеств. Примеры такого описания — формальные теории (п.4.1 и 4.2). Другая группа примеров, крайне важных в теории коммуникаций, — это алгоритмические языки программирования. Методы описания языков и построения компиляторов для них опираются на *теорию формальных грамматик,* представляющих собой еще один вид абстрактных формальных систем.

4.5. Гиперсети как высшая форма организация коммуникационных сетей

В настоящем разделе рассматривается классификация коммуникационных гиперсетей, приведены определения маршрутов, отделимости и соединимости вершин в коммуникационных гиперсетях и алгоритмы их вычисления в том случае, когда указанные задачи не *NP*-полные. Исследуются методы синтеза некоторых классов коммуникационных гиперсетей с заданной связностью. Гиперсети в качестве математических моделей связности сложных систем — наиболее адекватные модели структур коммуникационных сетей связи, энергетических, транспортных, нефте- и газопроводных и т. п. В процессе синтеза их структур важно учитывать не только оптимизацию, но и структурную надежность коммуникационных систем. Этим объясняется актуальность исследования характеристик связности коммуникационных гиперсетей.

4.5.1. Основные понятия и определения

1.1. Абстрактные гиперсети. Шестерку $S = (X, V, R; P, F, W)$ назовем абстрактной гиперсетью, если

$X = (x_1, x_2, ..., x_n)$— множество вершин;

$V = v_1, v_u ..., v_q)$— множество ветвей;

$R = (r_1, r_2, .. ., r_m)$— множество ребер;

$P : V \to 2^X$ — отображение, сопоставляющее каждому элементу $v \in V$ множество $P(v) \in X$ его вершин. Тем самым отображение P определяет гиперграф $PS = (X, V; P)$;

$F:R \to 2^V_{PS}$ — отображение, сопоставляющее каждому элементу $r \in R$ множество $F(r) \in V$ его ветвей. Причем семейство подмножеств ветвей 2^V_{PS} содержит только такие, ветви которых составляют связную часть гиперграфа PS. Отображение F определяет гиперграф $FS = (V, R; F)$;

$\forall r \in R \;\; W : r \to 2^{P(F(r))}$ — отображение, сопоставляющее каждому элементу $r \in R$ подмножество $W(r) \subseteq P(F(r)) \subseteq X$ его вершин, где $P(F(r))$—множество вершин в PS, инцидентных ветвям $F(r) \in V$. Таким образом, отображение W определяет гиперграф $WS = (X,R; W)$.

Гиперграф PS назовем первичной коммуникационной сетью гиперсети S, а гиперграф WS — вторичной коммуникационной сетью гиперсети S. Обратные отображения определяются следующим образом:

$$P^{-1}(x) = \{v : x \in P(v)\}, \; F^{-1}(v) = \{r : v \in F(r)\},$$
$$W^{-1}(x) = \{r : x \in W(r)\}.$$

В некоторых случаях для удобства будем использовать следующие обозначения гиперсетей: S, $S = (X, V, R)$, $S = (PS, WS; \Phi)$. В первом случае указывается только «имя» абстрактной гиперсети, во втором — образующие множества, а в третьем — «имена» первичной и вторичной коммуникационных сетей гиперсети S и отображение $\Phi : WS \to PS$.

Абстрактные гиперсети допускают одно важное для моделирования коммуникационных систем сетевой структуры обобщение. Пусть заданы гиперграфы

$$PS = (X, V; P) = WS_0 = (X, R_0; W_0), \;\; WS_1 = (Y_1, R_1; W_1), \ldots$$
$$\ldots, WS_k = (Y_k, R_k; W_k).$$

Тогда последовательность отображений

$$\Phi_i : WS_k \xrightarrow{\Phi_k} WS_{k-1} \xrightarrow{\Phi_{k-1}} \ldots \xrightarrow{\Phi_2} WS_1 \xrightarrow{\Phi_1} WS,$$

определяет иерархическую абстрактную k-гиперсеть

$$S = (PS, \; WS_1, \; \ldots, \; WS_k; \; \Phi_1, \; ., \; \Phi_k),$$

если $Y_k \subseteq Y_{k-1} \subseteq \ldots \subseteq Y_1 \subseteq X$ и $\forall i \in 1, \ldots, \; k, \; \forall r^i \in R_i$ $P_i(r^i) = \{r^{i-1}\} \subseteq R_{i-1}, \; W_i(r^i) \subseteq W_{i-1}(\{r^{i-1}\})$ и $\{r^{i-1}\}$ образуют связную часть в гиперграфе WS_{i-1}.

1.2. Инцидентность. Смежность. Отображения *P, F, W* вместе с обратными отображениями являются отношениями инциденции в соответствующих гиперграфах *PS, FS, WS* и, следовательно, они определяют инцидентность элементов в абстрактной гиперсети *S*.

Фундаментальным понятием в теории гиперсетей является отношение слабой инциденции. Два элемента из различных множеств слабо инцидентны, если найдется элемент из третьего множества, инцидентный им обоим. Например, вершина $x \in X$ слабо инцидентна ребру $r \in R$ (т. е. $x \in W^*(r)$), если только существует ветвь $v \in V$ такая, что $x \in P(v)$ и $v \in F(r)$, т. е. $x \in P(F(r))$.

Слабо инцидентные элементы могут оказаться инцидентными. Поэтому в некоторых случаях необходимо рассматривать понятие строгой слабой инцидентности, т. е. такие слабо инцидентные элементы, которые не могут быть инцидентными.

Для элементов абстрактных гиперсетей можно определить шесть понятий смежности и столько же — слабой смежности. Действительно, по аналогии с графами и гиперграфами два элемента из одного множества смежны тогда и только тогда, когда найдется элемент из другого множества, инцидентный им обоим. Но так как в абстрактных гиперсетях для пары элементов из любого множества можно найти инцидентные элементы из двух различных множеств, то в этом случае для конкретных элементов имеют место соответственно два понятия смежности. Например, вершины x и y из X V-смежны, если $\exists v \in V : x \in P(v)$ и $y \in P(v)$, и R-смежны, если $\exists r \in R : x \in W(r)$ и $y \in W(r)$. Аналогично определяются смежность других элементов S и слабая смежность этих элементов. Степень (слабая степень) элементов из S равна числу смежных (слабо смежных) ему элементов, т. е. для каждого элемента имеем четыре определения его степени.

4.5.2. Классификация гиперсетей

2.1. Из определений непосредственно следует, что абстрактная гиперсеть обобщает такие математические объекты, как ультраграфы, гиперграфы, гиперсхемы (по зарубежным публикациям — гиперсети), графы, графы связей и т. п. Вводя различные ограничения на множества *X, V, R* и отображения *P, F, W*, можно получить различные классы гиперсетей и перечисленные математические модели связности.

Характерной особенностью гиперсетей является то, что рассматриваются **три независимых множества элементов и отношения инциденции между ними**. Поэтому при классификации гиперсетей эффективным подходом будет рассмотрение подклассов,

инициируемых только ограничениями на *P, F, W* и типом связной части гиперграфа *PS* при отображении $F : R \to 2^V_{PS}$. Иерархические гиперсети нами не рассматриваются.

2.2. Если предположить, что первичная *PS = (X, V; P)* и вторичная *WS = (X, R; W)* коммуникационные сети абстрактной гиперсети *S* являются графами, а связные части 2^V_{PS} — всевозможными маршрутами в *PS*, то имеет место обычное определение гиперсети. Когда маршрут есть коммуникационная цепь в *PS*, то гиперсеть называется обыкновенной, в случае простой цепи имеем простую гиперсеть. Основное внимание в данном разделе уделено простым гиперсетям.

2.3. В определении абстрактной гиперсети рассматриваются два понятия: **слабая инцидентность и инцидентность**. Таким образом, вводя ограничения на соответствующие отношения, будем получать различные классы абстрактных гиперсетей. Например, пусть отображение $P:V \to 2^X$ двузначно, тогда первичная коммуникационная сеть *PS* абстрактной гиперсети является графом. Этот факт будем обозначать $\{X \to V\}$, т. е. любому элементу $v \in V$ инцидентны не более чем две вершины $x, y \in X$. В том случае, когда накладывается ограничение на слабую инцидентность двух элементов, $\{R>X\}$ означает, что любой вершине $x \in X$ слабо инцидентно не более двух ребер, причем здесь рассматривается строгая слабая инцидентность (см. п. 1.2).

2.4. Число инцидентных или слабо инцидентных элементов для заданного элемента из другого множества — основной параметр классификации абстрактных гиперсетей. Обозначим скобками гиперсеть, в которой соответствующий элемент имеет не более чем один инцидентный (слабо инцидентный) элемент. Фигурные скобки обозначают инцидентность двум элементам, а $\{\ \}^k$ — инцидентность k элементам $(k \geq 3)$. Например, в $(R \to V)$ - гиперсети каждой ветви $v \in V$ инцидентно не более чем одно ребро $r \in R$. Классификация гиперсетей завершена, если для каждой упорядоченной пары множеств из *X, V, R* определены численные ограничения на отношения инциденции (слабой инциденции) и тип связной структуры в первичной коммуникационной сети *PS=(X, V; P)* при отображении *F* (см. п. 2.2).

Легко заметить, что некоторые классы абстрактных гиперсетей включают в себя другие классы (например, *(R> X)*-гиперсеть простая, а *(R > V)* -гиперсеть обыкновенная; обратное неверно) или абстрактная гиперсеть может принадлежать различным классам. Некоторые классы могут иметь пустое пересечение. Из сказанного следует, что система

классов абстрактных гиперсетей имеет нетривиальную структуру, установить зависимость между классами в некоторых случаях сложно.

2.5. Так как в коммуникационных сетях, моделируемых абстрактными гиперсетями, первичная коммуникационная сеть в основном является графом, то исследование характеристик связности будет осуществляться только для гиперсетей (вторичная коммуникационная сеть может быть гиперграфом). В п. 2.2 перечислены некоторые типы связных частей в 2^V_{PS}. Необходимо добавить такие важные структуры, как деревья (*T*-гиперсети), циклы (*C*-гиперсети) и *k*-полюсники (*H*-гиперсети). В последнем случае в 2^V_{PS} рассматриваются всевозможные части графа $PS=(X, V; P)$ с выделенными *k* вершинами (например, подграф с заданной связностью между парами заданных вершин).

4.5.3. Маршруты и метрика в гиперсетях

Понятие «маршруты» играет фундаментальную роль в анализе связности коммуникационных гиперсетей и исследовании их метрических свойств. То что маршруты в коммуникационных гиперсетях задаются по-разному, способствует расширению изобразительных и операционных средств теории коммуникаций. В этом разделе рассмотены принципы решения некоторых задач, связанные с поиском маршрутов в коммуникационных гиперсетях.

3.1. Маршрутом в коммуникационной гиперсети $S=(X, V, R)$ называется конечная последовательность $\mu=(x_1, r_1, x_2, \ldots, x_{k-1}, r_{k-1}, x_k)$, составленная из элементов *X, R* таким образом, что вершины и ребра чередуются, а всякие два соседних элемента инцидентны. Квазимаршрут в коммуникационной гиперсети $S=(X, V, R)$ — это конечная последовательность μ, в которой пара соседних элементов x_i, r_i инцидентна, а r_i, x_{i+1} слабо инцидентна. Если в определении маршрута заменить «инцидентность» на «слабую инцидентность», то получим определение слабого маршрута. Ориентированные маршруты определяются аналогично, с учетом ориентации ребер.

3.2. Рангом $\nu(\mu)$ маршрута μ (квазимаршрута, слабого маршрута) называется число ребер (или их частей), принадлежащих этому маршруту. Отдаленность (квазиотдаленность, слабая отдаленность) между вершинами $x, y \in X$ численно равна рангу кратчайшего маршрута (квазимаршрута, слабого маршрута), соединяющего эти вершины, и обозначается через $\nu(x, y)(\overline{\nu}(x, y), \tilde{\nu}(x, y))$.

Отдаленность и слабая отдаленность удовлетворяют аксиомам метрики, квазиотдаленность — аксиомам орметрикп.

Длиной ребра (или его части) называется число ветвей, инцидентных этому ребру (части ребра). Длина $\rho(\mu)$ маршрута μ (квазимаршрута, слабого маршрута) равна суммарной длине ребер (их частей), входящих в маршрут μ. Расстояние (квазирасстояние, слабое расстояние) между вершинами $x, y \in X$ в гиперсети S равно длине кратчайшего маршрута (квазимаршрута, слабого маршрута), соединяющего эти вершины, и обозначается через $\rho(x,y)(\bar{\rho}(x,y), \tilde{\rho}(x,y))$. Расстояние и слабое расстояние удовлетворяют аксиомам метрики, квазирасстояние — аксиомам орметрики. Аналогично определяются эти понятия для ориентированных гиперсетей.

Рассмотрим теперь методы сводимости задач поиска кратчайших маршрутов в гиперсетях к аналогичным задачам на графах и гиперграфах.

3.3. Кратчайшие маршруты. Из определения маршрута в гиперсети $S = (X, V, R)$ непосредственно следует, что каждому маршруту μ в S между вершинами x и y соответствует маршрут μ в графе $WS = (X, R)$.

Теорема 4.21. *Пусть $S = (X, V, R)$—простая гиперсеть, заданная матрицами M^{xv}, M^{xr} и M^{rv}. Тогда (i, j)-й элемент матрицы A^p равен числу маршрутов ранга p из x_i в x_j, где*

$$A = M^{xr} \circlearrowright M^{rx}, \text{ а } M^{xr} = (M^{rx})^T.$$

Следствие. *Если S — связная гиперсеть, то $v(x_i, x_j)$, $i \neq j$, равно наименьшему из целых чисел p, для которых (i, j)-й элемент матрицы A^p отличен от нуля.*

Для ориентированных маршрутов следуют аналогичные утверждения.

Расстояние $\rho(x_i, x_j)$ между парой вершии x_i и x_j в гиперсети S находится по взвешенному графу $WS^* = (X, R)$. Каждому ребру r_k графа WS^* ставится в соответствие длина, равная $|\{v_i\}|$, где $\{v_i\} = F(r_k)$. Таким образом, кратчаншие маршруты в гиперсетях можно найти с помощью известных алгоритмов по графу WS^*. Для ориентированной гиперсети граф WS^* ориентируется.

3.4. Кратчайшие квазимаршруты. Отношении квазиотдаленности и квазирасстояния несимметричны для неориентированных гиперсетей, тем более для ориентированных. Рассмотрим способы вычисления кратчайших квазимаршрутов между различными упорядоченными парами вершин $x_i, x_j \in X$.

Теорема 4.22. *Пусть $S = (X, V, R)$ — простая гиперсеть, заданная матрицами M^{xv}, M^{xr} и M^{rv}. Тогда (i, j)-й элемент матрицы B^p равен числу квазимаршрутов ранга p из x_i в x_j, где $B = M^{xr} \circlearrowright M^{rv} \circlearrowright M^{vx}$.*

Следствие. *Если S — связная гиперсеть, то* $\overline{v}(x_i, x_j)$, $i \neq j$, *равно наименьшему из целых чисел p, для которых* (i, j)*-й элемент матрицы* B^p *отличен от нуля.*

Доказательства теоремы 4.22 и следствия следуют из того факта, что матрица B является матрицей смежности некоторого ориентированного графа $BS = (X, U)$, в котором из вершины x_i идет дуга $u \in U$ в вершину x_j, если и только если в гиперсети S *существует* ребро r_k, инцидентное вершине x_i и слабо инцидентное x_j.

Для ориентированных гиперсетей также можно вычислить квазиотдаленность, если построить ориентированный граф $\overrightarrow{BS} = (X, E)$, в котором из x_i идет дуга $e \in E$ в вершину x_j, если и только если в гиперсети S существует ориентированное ребро r_k, исходящее из x_i и слабо инцидентное x_j. Для матрицы смежности $\vec{B}$ графа $\overrightarrow{BS}$ справедливы теорема 4.22 и следствие этой теоремы.

Квазиотдаленность $\overline{v}(x_i, x_j)$ между парой вершин $x_i, x_j \in X$ можно найти с помощью ультраграфа $US = (X, R; f, g)$, который строится по гиперсети S. Каждой вершине (ребру) гиперсети S ставится в соответствие вершина (ребро) ультраграфа $US = (X, R; f, g)$. Отображения $f: X \rightarrow R$ и $g : R \rightarrow X$ для US определяются следующим образом:

$\forall x_i \in X$ $r_j \in f(x_i)$ тогда и только тогда, когда вершина x_i инцидентна ребру r_j в гиперсети S;

$\forall r_j \in R$ $x_i \in g(r_j)$ тогда и только тогда, когда вершина x_i слабо инцидентна ребру r_j.

Нетрудно показать, что $\overline{v}(x_i, x_j)$ в гиперсети S равно длине кратчайшего сильного маршрута в ультраграфе US между соответствующими вершинами.

Ориентированной гиперсети ставится в соответствие ориентированный ультраграф $\overrightarrow{US} = (X, R; f, g)$, в котором

$\forall x_i \in X$ $r_j \in f(x_i)$ тогда и только тогда, когда ориентированное ребро $r_j \in S$ выходит из вершины $x_i \in S$;

$\forall r_j \in R$ $x_i \in g(r_j)$ тогда п только тогда, когда вершина x_i слабо инцидентна ребру r_j и $r_j \in f(x_i)$.

Между парой вершин x_j и $x_j \in S$ существует кратчайший ориентированный квазимаршрут тогда и только тогда, когда существует кратчайший сильный маршрут между соответствующими вершинами в ориентированном ультраграфе $\overrightarrow{US}$.

Квазирасстояние $\overline{\rho}(x_i, x_j)$ между парой вершин x_i и x_j гиперсети S можно найти по взвешенному орграфу $BS^* = (X, U)$, в котором длина

дуги *u* равна числу ветвей в части ребра r_k, соединяющего вершины x_i и x_j в гиперсети *S*, т. е. числу элементов в соответствующей части кортежа $F(r_k)=\{x_i, .. ., x_j; ...\}$. Для ориентированных гиперсетей квазирасстояние от вершины x_i к вершине x_j находится с помощью взвешенного орграфа $\overrightarrow{BS} = (X, E)$, в котором длины дуг определяются аналогично.

Заметим, что вычисление кратчайших квазимаршрутов с помощью специально построенных графов и ультраграфов не накладывает ограничений на класс гиперсетей.

3.5. Кратчайшие слабые маршруты. Отношения слабой отдаленности и слабого расстояния играют большую роль при решении коммуникационных задач на гиперсетях. Так же как и в предыдущих случаях, справедлива следующая теорема.

Теорема 4.23. *Пусть S = (X, V, R) — простая гиперсеть, заданная матрицами* M^{xv}, M^{xr} *и* M^{rv}. *Тогда* (i, j)*-й элемент матрицы* D^p *равен числу слабых маршрутов ранга p из* x_i *в* x_j, *где*

$$D = M^{xv} \ominus M^{vr} \ominus M^{rv} \ominus M^{vx}.$$

Следствие. *Если S — связная гиперсеть, то* $\tilde{\nu}(x_i, x_j)$, $i \neq j$, *равно наименьшему из целых чисел p, для которых* (i, j)*-й элемент матрицы* D^p *отличен от нуля.*

Доказательства теоремы 4.23 и следствия вытекают из свойств графа $DS = (X, U)$, соответствующего матрице смежности *D*.

Множество вершин графа *DS* совпадает с множеством вершин гиперсети *S* и x_i смежна с x_j в *DS*, если и только если в гиперсети *S* найдется ребро r_k, слабо инцидентное x_i и x_j. В графе *DS* любому маршруту взаимно однозначно соответствует слабый маршрут в гиперсети *S*. Матрице *D* соответствует также гиперграф $HS = (X, R)$, который задается матрицей инциденций $N^{xr}=M^{xv} \ominus M^{vr}$, следовательно, кратчайшие слабые маршруты в гиперсети можно находить по гиперграфу *HS*. Кроме того, слабые маршруты в ультраграфе *US* соответствуют таковым в *S*.

Если гиперсеть *S* ориентированная, то ей можно поставить в соответствие орграф $\overrightarrow{DS} = (X, \vec{U})$, $|\vec{U}| = |U|$. Из вершины x_i идет дуга *u* в вершину x_j, если и только если существует ребро $r_k \in S$, слабо инцидентное x_i и x_j и ориентированное от x_i к x_j. Любому ориентированному маршруту в $\overrightarrow{DS}$ взаимно однозначно соответствует ориентированный слабый маршрут в *S*.

Слабое расстояние $\tilde{\rho}(x_i, x_j)$ в гиперсетях (ориентированных гиперсетях) между вершинами x_i и x_j, вычисляется с помощью

взвешенного графа $DS^*=(X, U)$ ($\overrightarrow{DS}^* = (X, \vec{U})$), в котором ребру (дуге) u ставится в соответствие длина, равная числу ветвей в части ребра r_k, соединяющего вершины x_i и x_j в гиперсети S, т. е. числу элементов в соответствующей части кортежа $F(r_k) = \{...; x_i,...; x_j;...\}$.

3.6. О поиске коммуникационных цепей в гиперсетях. Маршрут μ в гиперсети $S= (X, V, R)$ называется коммуникационной r-цепью, если каждое ребро используется не более одного раза, и коммуникационной v-цепыо, если каждая ветвь используется не более одного раза. Отсюда следует, что всякая коммуникационная v-цепь одновременно является коммуникационной r-цепью. Обратное неверно. Маршрут μ называется простой коммуникационной цепью, если в нем все вершины различны.

Лемма 1. *Всякий маршрут (квазимаршрут, слабый маршрут) гиперсети S содержит коммуникационную r-цепь (r-квазицепъ, слабую r-цепь), соединяющую ту же пару вершин.*

Теорема 4.24. *Всякий кратчайший маршрут (квазимаршрут, слабый маршрут) между двумя вершинами гиперсети S является коммуникационной r-цепъю (r-квазицепъю, слабой r-цепъю).*

Следствие. *Задачи поиска коммуникационных r-цепей (r-квазщепей, слабых r-цепей) между двумя вершинами в гиперсети S полиномиально вычислимы.*

Легко показать, что задачи поиска слабой простой коммуникационной цепи и слабой коммуникационной v-цепи эквивалентны соответственно задачам поиска простой коммуникационной цепи и коммуникационной цепи в графе PS.

ПКЦ: пусть задана гиперсеть $S = (X, V, R)$ и пара выделенных вершин $s, t \in X$. Существует ли простая коммуникационная цепь между вершинами s и t в гиперсети S?

Теорема 4.25. *Задача ПКЦ является NP-полной.*

Следствие. *Задача поиска простой квазицепи в гиперсети S между вершинами s и t является NP-полной.*

ВКЦ: пусть задана гиперсеть $S = (X, V, R)$ и пара выделенных вершин $s, t \in X$. Существует ли коммуникационная v-цепь между вершинами s и й гиперсети S?

Теорема 4.26. *Задача ВКЦ является NP-полной.*

Следствие. *Задача поиска коммуникационной v-квазицепи в гиперсети S между вершинами s и t является NP-полной.*

Для ориентированных гиперсетей справедливы аналогичные результаты.

4.5.4. Независимость и соединимость

4.1. Два маршрута, соединяющих пару вершин $x, y \in X$, называются внутренне независимыми (внешне независимыми), если не существует вершины $z \neq x, y$, инцидентной (строго слабо инцидентной) ребрам этих маршрутов. Маршруты называются независимыми, если они одновременно внутренне и внешне независимы. Два маршрута, соединяющих пару вершин, V-независимы (R-независимы), если не существует ветви (ребра), принадлежащей обоим маршрутам.

Аналогичным образом определяется независимость для квази- и слабых маршрутов. Поскольку в них отдельные ребра включены частично, введем еще одно определение независимости этих маршрутов.

Два квазимаршрута (слабых маршрута) частично R-независимы, если не существует участка ребра (части ребра, инцидентной одной ветви), принадлежащего обоим квазимаршрутам (слабым маршрутам).

4.2. Две вершины $x, y \in X$ в гинерсети S k-соединимы (k-квазисоединимы, слабо k-соединимы), если эти вершины соединены k-независимыми по вершинам маршрутами (квазимаршрутами, слабыми маршрутами). Аналогично определяются внутренняя и внешняя k-соединимость, k — V-соединимость и k — R-соединимость (**маршруты должны быть соответственно внутренне, внешне независимы по вершинам, незанисимы по ветвям или ребрам**).

Для квазимаршрутов и слабых маршрутов k-квазисоединимость и слабая k-соединимость по ветвям и ребрам определяются так же, как и в предыдущем случае, а частичная k-квазисоединимость и слабая k-соединимость по ребрам определяются следующим образом. Две вершины $x, y \in X$ в гиперсети $S=(X, V, R)$ k — R-квазисоединимы (слабо k — R-соединимы), если эти вершины соединены k частично R-независпмыми по ребрам квазимаршрутами (слабыми маршрутами).

4.3. Из определения соединимости непосредственно следует существование 17 задач вычисления k-соединимости пары вершин в гиперсети S. В табл. 1 приведена классификация задач поиска k-независимых (x, y)-маршрутов в смысле их принадлежности к полиномиально-вычислимым или NP-полным задачам.

Таблица 1.

Классификация задач вычисления k-соединимости в гиперсетях

Подклассы задач вычисления k-соединимости	Маршруты	Квазимаршруты	Слабые маршруты
Соединимость	*NP*-полная	*NP*-полная	*P*
Внутренняя соединимость	*P*	*P*	*P*
Внешняя соединимость	*NP*-полная	*NP*-полная	*P*
V-соединимость	*NP*-полная	*NP*-полная	*P*
R-соединимость	*P*	*P*	*P*
Частичная *R*-соединимость	—	*P*	*P*

4.4. Задачи вычисления k-соединимости и k — V-соединимости пары вершин в произвольной гиперсети $S = (X, V, R)$ являются *NP*-полными. Покажем, что остальные принадлежат классам (по сложности вычисления), указанным в табл. 1. Отметим, что задача вычисления частичной k — R-соединимости для маршрутов не имеет смысла, так как в маршруте каждое ребро полностью ему принадлежит.

Из определения внутренней k-соединимости следует независимость соответствующих маршрутов на первичной сети $PS = (X, V)$ гиперсети $S=(X, V, R)$, т. е. задача вычисления внутренней k-соединимости принадлежит классу *P*. *NP*-полная задача вычисления k-соединимости сводится к задаче вычисления внешней k-соединимости пары вершин в гиперсети S. Действительно, по гиперсети $S = (X, V, R)$ построим гиперсеть $S' = (X' \bigcup X, V' \bigcup V, R)$, применив к каждой вершине $x \in X$ операцию выделения инцидентной вершины (см. рис. 1).

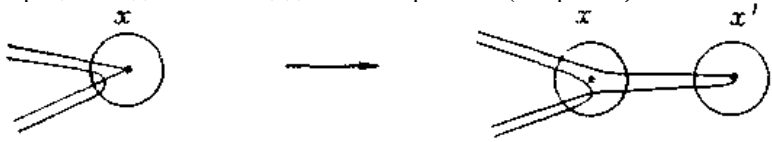


Рис. 1. Выделение инцидентной вершины.

Легко увидеть, что любым k-независимым (s, t)-маршрутам в гиперсети S соответствуют внешне k-независимые $(s, t$-маршруты, и наоборот. Поскольку задача поиска внешне k-независимых маршрутов принадлежит классу *NP*, тем самым доказана *NP*-полнота указанной задачи. Очевидно, что задача вычисления k — R-соединимости полиномиально вычислима, так как непосредственно сводится к задаче поиска k-независимых по ребрам (s, t)-маршрутов во вторичной коммуникационной сети $WS = (X, R)$.

4.5. Для квазимаршрутов имеют место следующие результаты.
Задача *XK(VK)*. Пусть заданы гиперсеть $S = (X, V, R)$, пара вершин $x, y \in X$ и целое положительное число k. Верно ли, что вершины x и y k-квазисоединимы (k—V-квазисоединимы) ?

Теорема 4.27. *Задачи XK и VK являются NP-полными.*

Доказательство. Очевидно, что задачи *XK* и *VK* принадлежат классу *NP*. Осталось показать, что *NP*-полная задача полиномиально сводима к задаче *XK* (для задачи *VK* доказательство строится аналогично).
Пусть заданы гиперсеть $S=(X, V, R)$ и пара вершин $x,y \in X$. Преобразуем гиперсеть $S = (X, V, R)$ в $S' = (X', V', R)$ следующим образом. К каждой ветви $v=(a, b) \in V$ добавим две инцидентные вершины a' и b', и каждую из вершин a и b разобьем на две $a^+, a^-; b^+, b^-$ (рис. 2) так, чтобы ребра, инцидентные вершинам a и b, стали инцидентными a^+ и b^+, а ребра, слабо инцидентные a и b, оказались слабо инцидентными a^- и b^-.

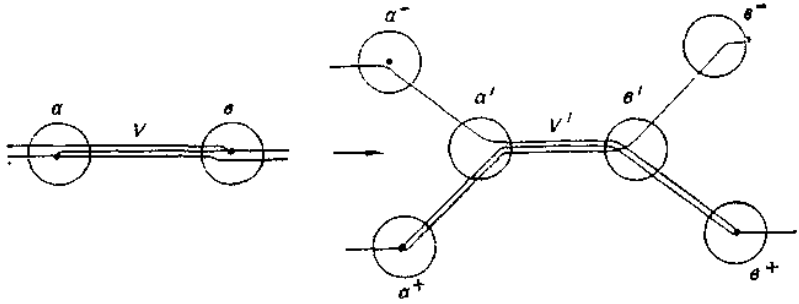


Рис. 2. Подразбиение вершины на инцидентную и слабо инцидентную.

В полученной гиперсети $S' = (X', V', R)$ каждому квазимаршруту в точности соответствует один маршрут в S, и наоборот. Действительно, в гиперсети S' содержится только два типа вершин: (a^-, a', b^-, b') слабо инцидентны ребрам, (a^+, b^+) инцидентны ребрам. Поэтому других квазимаршрутов (кроме совпадающих с маршрутами) в этой гиперсети нет.
Таким образом, решив задачу *XK*, этим решим и *NP*-полную задачу вычисления k-соединимости вершин в гиперсети S. Аналогично доказывается *NP*-полнота задачи *VK*, так как каждой ветви v в гиперсети S сопоставлена в точности одна ветвь v' в гиперсети S'. Теорема доказана.

Теорема 4.28. *Задача вычисления k — R-квазисоединимости пары вершин $x, y \in X$ в гиперсети $S = (X, V, R)$ полиномиально вычислима.*

Доказательство. Гиперсети $S = (X, V, R)$ сопоставим ультраграф

US = (X, R; f, g) (см. п. 3.4), тогда *k*-независимым по ребрам квазимаршрутам между вершинами *x* и *y* в гиперсети *S* будут взаимно однозначно соответствовать *k*-независимые по ребрам сильные маршруты в ультраграфе *US*. Но задача *k*-соединимости по ребрам в ультраграфе полиномиально вычислима. Теорема доказана.

Теорема 4.29. *Задача вычисления частичной k— R-квазисоединимости пары вершин в гиперсети S =(X, V, R) полиномиально вычислима.*

Доказательство. Гиперсети *S = (X, V, R)* сопоставим смешанный граф *G*=(*X* ⋃ *Y, E),* полученный из данной гиперсети по следующему правилу. К каждой вершине $x \in X$ добавляются $y_1, ..., y_{k_x}$, где k_x — слабая *k*-степень вершины *x*. Каждая вершина y_i подразбивает соответствующее ребро, слабо инцидентное *x*. От y_i к вершине *x* добавляются две дуги (y_i, x). Пример такого преобразования показан на рис. 3.

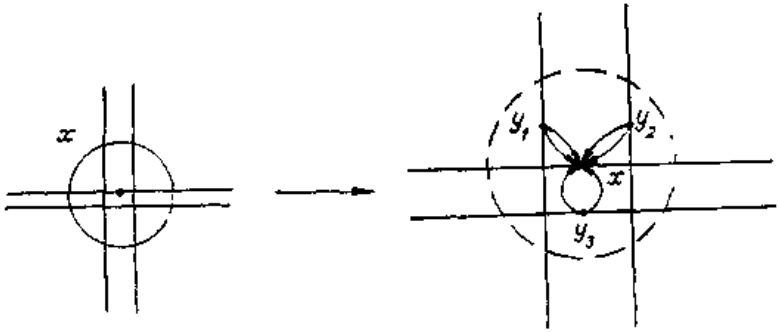


Рис.3. Локальное преобразование гиперсети в смешанный граф.

Можно показать, что любому квазимаршруту в гиперсети *S =(X, V, R)* взаимно однозначно соответствует ориентированный маршрут в смешанном графе *G*=(*X* ⋃ *Y, E)* между вершинами из множества *X*. Но в смешанном графе задача поиска *k*-независимых по ребрам и дугам маршрутов между парами вершин полиномиально вычислима. Теорема доказана.

Используя преобразование гиперсети *S*, изложенное в доказательстве теоремы 4.29, можно показать, что задача поиска внутренне *k*-независимых (s, t)-квазимаршрутов полиномиально вычислима.

В п. 4.4 гиперсеть *S'*, построенная для доказательства *NP*-полноты задачи внешней *k*-соединимости, может быть использована для доказательства *VP*-полноты задачи внешней *k*-квазисоединимости. Действительно, в гиперсети *S'* каждый квазимаршрут является маршрутом, и наоборот.

Теперь перейдем к рассмотрению слабых маршрутов и соответствующих задач соединимости.

4.6. Для слабых маршрутов справедливы следующие теоремы.

Теорема 4.30. *Задачи вычисления слабой k-соединимости и слабой k — V-соединимости пары вершин x, y ∈ X в гиперсети S = (X, V, R) полиномиально вычислимы.*

Доказательство. Из определения слабых маршрутов ясно, что любому из них в гиперсети однозначно соответствует маршрут в первичной коммуникационной сети *PS' =(X, V')*, которая получается из *PS* удалением ветвей, которым не инцидентно ни одно ребро. Отсюда и следует утверждение теоремы.

Теорема 4.31. *Задача вычисления слабой k — R-соединимости пары вершин x, y ∈ X в гиперсети S = (X, V, R) полиномиально вычислима.*

Доказательство. Из определения слабой *k — R*-соединимости следует, что любое ребро $r \in R$ входит в слабые маршруты не более одного раза. Поэтому гиперсети *S=(X, V, R)* сопоставили гиперграф *HS=(X, R)* (см. п. 3.5), в котором любым *k*-независимым по ребрам *(x, y)* -маршрутам в точности соответствуют *k*-независимые по ребрам *(x, y)* -маршруты в гиперсети *S*. Так как задача *k*-соединимости по ребрам в любом гиперграфе полиномиально вычислима, то из вышесказанного следует доказательство теоремы.

Теорема 4.32. *Задача вычисления частичной слабой k — R-соединимости пары вершин x, y∈X в гиперсети S = (X, V, R) полиномиально вычислима.*

Доказательство. Так как в частично *R*-независимых слабых маршрутах любое ребро может по частям войти в различные слабые маршруты, то для доказательства теоремы достаточно построить такой граф *L=(X, E)*, в котором любому маршруту взаимно однозначно соответствует слабый маршрут в гиперсети *S*. Такой граф строится по гиперсети *S = (X, V, R)* следующим образом. Все слабо инцидентные вершины для любого ребра сделать инцидентными, в полученной гиперсети *S** рассмотреть вторичную коммуникационную сеть *WS* = (X, E)* и отождествить граф *L* со вторичной коммуникационной сетью *WS**. Очевидно, что *L* является мультиграфом и *k*-соединимость по ребрам между вершинами $x, y \in X$ в *L* равна частичной *k — R*-соединимости этой пары вершин в гиперсети *S*. Теорема доказана.

Теорема 4.33. *Задача вычисления слабой внутренней k-соединимости пары вершин x, y∈X в гиперсети S=(X, V, R) полиномиально вычислима.*

Доказательство. Преобразуем гиперсеть *S = (X, V, R)* в граф *G(S) = (X* $\cup$ *X', E)* по следующему правилу. Каждой вершине $x \in X$ сопоставим $2\sigma_R(x)+1$ вершин *Y(x)*, где $\sigma_R(x)$ - слабая *R*-степень

вершины *x*. На множестве *Y(x)* строится полный граф. Все ребра, инцидентные вершине $x \in X$, остаются инцидентными одной и той же вершине $z \in Y(x)$. Ребра гиперсети, слабо инцидентные *x*, подразбиваются парой смежных вершин из *Y(x)*. Пример такой операции приведен на рис. 4.

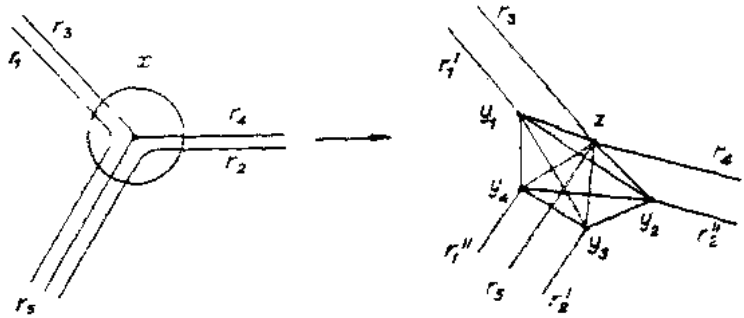


Рис. 4. Преобразование вершин гиперсети в клики слабой инциденпии.

В построенном графе *G(S)* между любой парой вершин z_i и z_j любому множеству непересекающихся цепей взаимно однозначно соответствует множество слабых внутренне независимых маршрутов между соответствующими вершинами в гиперсети *S*. Следовательно, по графу *G(S)* за полиномиальное время можно вычислить слабую внутреннюю *k*-соединимость пары вершин в гиперсети *S*. Теорема доказана.

Аналогичным образом доказывается полиномиальная вычислимость задачи поиска слабой внешней *k*-соединимости пары вершин в гиперсети *S*. В этом случае гиперсеть *S* преобразуется в граф *L(S)* с помощью операции, показанной на рис. 5.

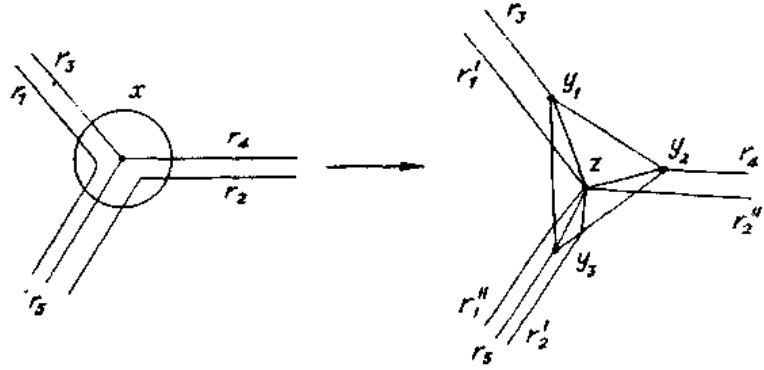


Рис. 5. Преобразование вершин гиперсети в клики инциденции.

4.7. Некоторые *NP*-полные задачи поиска *k*-соединимости пары вершин в гиперсети *S* можно решить за полиномиальное время на важных и нетривиальных классах гиперсетей. Рассмотрим *(R>X)*-гиперсети, в которых каждой вершине строго слабо инцидентно не более одного ребра. Тогда для таких гиперсетей справедлива следующая теорема.

Теорема 4.34. *В (R>X)-гиперсетях задача поиска внешней k-соединимости (k-квазисоединимости) пары вершин полиномиально вычислима.*

Доказательство. Так как каждой вершине $x \in X$ слабо инцидентно не более одного ребра, то любые независимые по ребрам *(s, t)*-маршруты будут также внешне независимы и, следовательно, для решения задачи можно воспользоваться графом вторичной коммуникационной сети *WS* гиперсети *S*. В случае квазимаршрутов по гиперсети *S* можно построить смешанный граф *G* (теорема 4.29) и тогда любым непересекающимся по ребрам (*s, t)*-цепям в *G* взаимно однозначно соответствуют внешне независимые (*s, t)*-маршруты в гиперсети *S*. Теорема доказана.

Можно предположить, что для *(R > X)* -гиперсетей задачи поиска *k*-соединимости и *k*-квазисоединимости также полиномиально вычислимы. Используя преобразования гиперсети *S*, приведенные в теореме 4.34, легко можно показать, что задачи вычисления *k — V*-соединимости и *k — V*-квазисоединимости пары вершин в *(R→ V)* -гиперсети *S* решаются за полиномиальное время.

4.5.5. Отделимость и связность коммуникационных гиперсетей

Связность любой коммуникационной структурной модели определяется, с одной стороны, способом достижимости вершин (т. е. типом маршрута), а с другой — типом и характером удаления элементов из структурной модели.

5.1. Коммуникационная гиперсеть называется связной, если и только если между любой парой вершив гиперсети *S* существует соединяющий их маршрут. Отсюда следует, что графы *PS* и *WS* связны. Коммуникационная гиперсеть называется односторонне квазисвязной, если и только если любая пара вершин из *S* соединима хотя бы одним квазимаршрутом. Коммуникационная гиперсеть квазисвязна, если и только если для любых вершин $x, y \in X$ существуют квазимаршруты $\mu(x, y)$ и $\mu(y, x)$. Коммуникационная гиперсеть называется слабо

связной, если и только если любая пара вершин S соединима слабым маршрутом.

Рассмотренные отношения связности упорядочиваются по включению, т. е. из связности → квазисвязность → слабая связность. Обратное неверно.

Коммуникационная гиперсеть S называется насыщенной, если гиперграф FS связен, и ненасыщенной — в противном случае. Коммуникационная гиперсеть S называется полносвязной, если и только если связны графы PS, WS и гиперграф FS. Очевидно, что несвязность S влечет несвязность WS.

5.2. Понятия k-отделимости в гиперсетях связаны со способом удаления элементов. Можно ввести различные способы удаления элементов. Здесь рассмотрим лишь те, которые отражают возможные преобразования структур коммуникационных систем, моделируемых гиперсетями.

1. Удаление ребер. Удаляются ребра в гиперсети $S = (X, V, R)$ без инцидентных вершин и ветвей.
2. Частичное удаление ребер заключается в удалении части ребра, инцидентной некоторой ветви. При этом ребро разделяется по крайней мере на две части.
3. Удаление ветвей. Удаляются ветви с инцидентными ребрами без инцидентных вершин.
4. Частичное удаление ветвей определяется аналогично, но ребра удаляются частично.
5. Внешнее удаление вершин. Удаляются ребра, слабо инцидентные данным вершинам.
6. Внутреннее удаление вершин. Удаляются ребра, инцидентные данным вершинам.
7. Удаление вершин. Удаляются вершины вместе с инцидентными ветвями (ребра, инцидентные удаленным ветвям, также удаляются).

Из введенных способов удаления вершин можно рассмотреть частичное; остановимся на одном способе удаления.

8. Частичное удаление вершины. Определяется аналогично предыдущему способу, но ребра, инцидентные удаляемым ветвям, удаляются частично.

5.3. Теперь можно привести возможные определения k-отделимости пары вершин в гиперсети $S = (X, V, R)$. Две вершины $x, y \in X$ в гиперсети $S=(X, V, R)$ $k_R(l_R, m_R)$-отделимы (квази-, слабо отделимы), если $k_R(l_R, m_R)$ — наименьшее число ребер, удаление которых (для других способов удаления элементов определения отделимости аналогичны) приводит к разрушению всех маршрутов (квази-,

слабых маршрутов) между вершинами x и y в гиперсети S. Таким образом, **даны определения 24 понятий отделимости пары вершин**. k-связность гиперсетей определяется через приведенные понятия отделимости.

Пример. Гиперсеть $S = (X, V, R)$ k_X-связна, если k_X принимает наименьшее значение из k_X-отделимости всех пар вершин гиперсети S.

Часть гиперсети S' назовем остовной, если из исходной гиперсети $S = (X, V, R)$ удалено некоторое подмножество ребер R', т. е. $S' = (X, V, R—R')$. Часть гиперсети S' назовем усеченной остовной, если из исходной гиперсети $S = (X, V, R)$ удалено некоторое множество ветвей V' (а следовательно, и инцидентные им ребра R'), т. е. $S' = (X, V—V', R — R')$. При внешнем или внутреннем удалении вершин, очевидно, будет получена остовная гиперсеть. Часть гиперсети S' назовем подгиперсетью (или просто подсетью), если из исходной гиперсети $S = (X, V, R)$ удалено некоторое множество вершин X' (а следовательно, и инцидентные им ветви V' и ребра R'), т.е. $S' = (X—X', V—V', R—R')$.

5.4. Для получения оценок связности гиперсети $S = (X, V, R)$ будут полезны некоторые неравенства, частично упорядочивающие численные значения характеристик связности.

Обозначим: $\Theta := k, l, m$, тогда $\Theta_R(S)$—реберная связность гиперсети S при $\Theta = k$, реберная квазисвязность гиперсети S при $\Theta = l$, реберная слабая связность гиперсети S при $\Theta = m$. Для остальных характеристик связности Θ также принимает три значения. $H := X, V, R$, тогда $k_H(S)$ — связность гиперсети S при $H=X$, связность по ветвям гиперсети S при $H = V$, связность по ребрам при $H = R$. $\sigma = 1, 2, ..., 8$ — номер способа удаления элементов из гиперсети S (см. п. 5.2). Тогда, например, $l_3(S)$ означает l_V-квазисвязность гиперсети S по ветвям, а $m_6(S)$— внутреннюю слабую связность гиперсети S. Черта над буквой означает частичную отделимость (см. также табл. 2).

Справедливы следующие неравенства:

$$\forall \sigma = \overline{1,8} \qquad m_\sigma(S) \geqslant l_\sigma(S) \geqslant k_\sigma(S), \tag{4.19}$$

$$\forall \Theta = k, l, m \quad \Theta_R(S) \geqslant \Theta_V(S) \geqslant \Theta_X(S), \tag{4.20}$$

$$\bar{\Theta}_R(S) \geqslant \bar{\Theta}_V(S) \geqslant \bar{\Theta}_X(S), \tag{4.21}$$

$$\Theta^0_X(S) \geqslant \Theta_X(S), \tag{4.22}$$

$$\Theta^I_X(S) \geqslant \Theta_X(S), \tag{4.23}$$

$$\Theta_R(S) \geqslant \Theta^I_X(S), \tag{4.24}$$

$$\forall H = X, V, R; \Theta = k, l, m \ \bar{\Theta}_H(S) \geqslant \Theta_H(S)$$
(при $\Theta = k$ имеет место равенство). (4.25)

На рис. 6 значения характеристик связности гиперсети частично упорядочены согласно приведенным неравенствам.

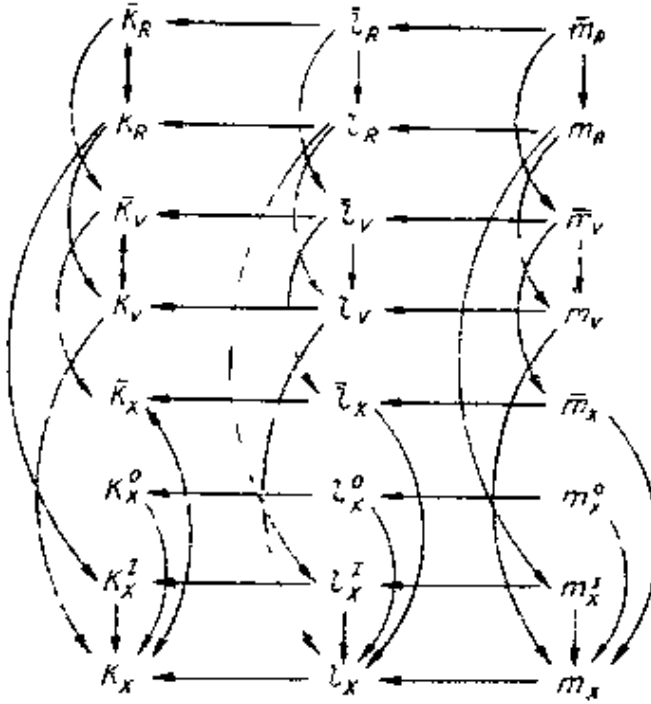


Рис. 6. Граф частичной упорядоченности характеристик связности.

4.5.6. О сложности вычисления отделимости в гиперсетях

Так же как в случае соединимости пары вершин в гиперсети *S*, задачи их отделимости имеют различную сложность. Некоторые из этих задач решаются за полиномиальное время, другие принадлежат к классу *NP*-полных задач. В табл. 2 указаны идентификаторы задач отделимости и их соответствующий класс сложности вычисления.

Таблица 2.

Класификация задач вычисления *k*-отделимости в гиперсетях

Удаление элементов	Маршрут		Квазимаршрут		Слабый маршрут	
Ребра	k_R	P	l_R	P	m_R	P
Ребра, частичное	$\overline{k}_R$	P	$\overline{l}_R$	P	$\overline{m}_R$	P
Ветви	k_V	NP-полная	l_V	NP-полная	m_V	NP-полная
Ветви, частичное	$\overline{k}_V$	NP-полная	$\overline{l}_V$	NP-полная	$\overline{m}_V$	P
Вершин, внешнее	k^o_X	NP-полная	l^o_X	NP-полная	m^o_X	NP-полная
Вершин, внутреннее	k^I_X	P	l^I_X	NP-полная	m^I_X	NP-полная
Вершин	k_X	NP-полная	l_X	NP-полная	m_X	NP-полная
Вершин, частичное	$\overline{k}_X$	NP-полная	$\overline{l}_X$	NP-полная	$\overline{m}_X$	P

6.1. Отделимость. Напомним, две вершины $x, y \in X$ в гиперсети S отделимы, если при удалении некоторых элементов гиперсети между вершинами x и y отсутствует маршрут, их соединяющий.

Задача k_R-отделимости решается за полиномиальное время, к этому же классу относится задача внутренней k^I_X - отделимости. Так же показана NP-полнота задач: k_V-, k^0_X-, k_X-отделимости. Действительно, полиномиальность первых двух задач следует из равенств $k_R(S) = \lambda(WS)$, $k^I_X = \omega(WS)$ соответственно, где $\lambda(WS)$—отделимость по ребрам соответствующей пары вершин в графе вторичной сети WS гиперсети S, а $\omega(WS)$—отделимость по вершинам этой же пары вершин в WS. Справедливость утверждений (см. табл. 2) о сложности вычисления частичной отделимости пары вершин в гиперсети S следует из справедливости равенства (7) для $\Theta = k$.

6.2. Квазиотделимость. Задача вычисления l_R-квазиотделимости может быть решена за полиномиальное время. По гиперсети $S=(X,V,R)$ строится ультраграф $US = (X, R; f, g)$ (см. п. 3.4), в этом ультраграфе любому сильному маршруту взаимно однозначно соответствует квазимаршрут в гиперсети $S = (X, V, R)$. Но k-соединимость по ребрам в ультраграфе US равна k-отделимости по ребрам (при слабом удалении ребер). Отсюда следует полиномиальность вычисления l_R-квазиотделимости пары вершин в гиперсети S. Аналогично доказывается полиномиальная вычислимость частичной $\overline{l}_R$-квазиотделимости, но вместо ультраграфа US рассматривается граф $G = (X \cup Y, E)$ (см. теорему 4.29).

Существует доказательство NP-полноты задач вычисления l_V-, l^0_X-, l^I_X-l_X -отделимости пары вершин в гиперсети $S = (X, V, R)$. Доказана NP-

полнота задач частичной $\overline{l}_V$ - и $\overline{l}_X$ -квазиотделимости. Действительно, при доказательстве *NP*-полноты задач l_V- и l_X-квазиотделимости *x* и *y* фактически можно рассматривать удаление ребер, хотя и частичное, но разрушающее все квазимаршруты из вершины *x* в вершину *y*.

6.3. Слабая отделимость. Для задач слабой m_V-, m^0_X- ,m^I_X-, m_X-отделимости двух вершин в гиперсети *S = (X, V, R)* их *NP*-полнота доказана В.К.Попковым. Им же показано, что задача слабой m_R-отделимости решается за полиномиальное время путем преобразования гиперсети *S* в гиперграф *HS = (X, R)* (см. п. 3. 5). Можно показать, что задача слабой $\overline{m}_R$ -отделимости также решается за полиномиальное время. Для этой цели достаточно перейти от гиперсети *S* к графу *L* (см. п. 4.6, теорема 4.22). Если рассмотреть первичную сеть *PS = (X, V)* гиперсети *S* и удалить из нее ветви, не инцидентные ребрам, то в полученном графе *PS' =(X, V')* задачи определения реберной λ-отделимости и вершинной ω-отделимости пары вершин *x, y* $\in$ *X* эквивалентны задачам слабой $\overline{m}_V$ -отделимости и слабой $\overline{m}_X$ -отделимости соответствующих вершин в гиперсети *S*. Таким образом, эти задача также полиномиально вычислимы.

6.4. В теории графов справедливы теоремы менгеровского типа, т. е. численное значение отделимости пары вершин равно численному значению их соединимости. Для гиперсетей это положение не всегда выполняется. Например, на рис. 7 приведена гиперсеть *S = (X, V, R)*, в которой вершины s и *t* 1-соединимы, но 2-отделимы.

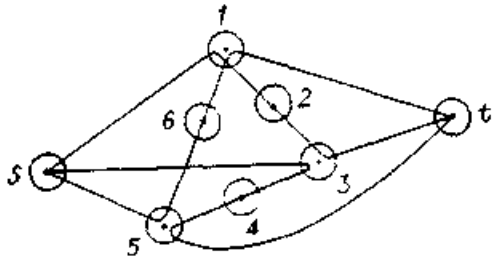


Рис. 7. Пример двухполюсной гиперсети.

Из примера ясно, что квази- и слабая соединимость отличаются от квази- и слабой отделимости. Действительно, *s* и *t* слабо 3-соединимы, но слабо 2-отделимы. Можно показать, что для полиномиально вычислимых задач соединимости и отделимости в гиперсетях теорема

Менгера справедлива. Причем при исследовании $\overline{\Theta}_H$ (S)-отделимости вершин необходимо рассматривать при $H = X$ k-соединимость, при $H=V$ k — V-соединимость, при $H = R$ частичную k — R-соединимость.

Покажем, что для NP-полных задач k_V-, $k^0{}_X$-, k_X-отделимости справедливы следующие неравенства:

$$k_V \geqslant z_V, \quad k_X^0 \geqslant z_X^0, \quad k_X \geqslant z_X, \tag{4.26}$$

где z_V равно k — V-соединимости, $z^0{}_X$— внешней k-соединимости, z_X — k-соединимости. В самом деле, так как при удалении элементов в гиперсети S должны быть разрушены (s, t)-маршруты, а по определению k-соединимости пары вершин элементы маршрутов полностью не пересекаются, то очевидно, что k-отделимость не меньше k-соединимости.

В случае слабой связности пары вершин в гиперсети $S =(X, V, R)$ имеют место обратные неравенства:

$$m_V \leqslant \tilde{z}_V, \quad m_X^0 \leqslant \tilde{z}_X^0, \quad m_X^I \leqslant \tilde{z}_X^I, \quad m_X \leqslant \tilde{z}_X, \tag{4.27}$$

где m — слабая отделимость, а $\tilde{z}$ — слабая соединимость одной и той же пары вершин s, t в гиперсети S. Справедливость этих неравенств следует из того факта, что при удалении одного элемента в гиперсети S может быть разрушено несколько независимых слабых (s, t)-маршрутов.

Соотношения квазиотделимости и квазисоединимости почти не исследованы, но можно показать, что l_V-, $l^0{}_X$-, l_X-квазиотделимость не оценивается через соответствующую квазисоединимость. Так как слабая соединимость легко вычисляется, то с учетом неравенств (4.25) и (4.19) — (4.25) можно оценивать сверху различные типы отделимости пары вершин в гиперсети S.

6.5. Сложность вычисления большинства задач k-отделимости в гиперсетях не позволяет исследовать характеристики связности гиперсетей большой размерности уже при $k > 3$. Однако для некоторых классов гиперсетей задачи вычисления k-отделимости пары вершин решаются за полиномиальное время.

Теорема 4.35. *Задача вычисления внутренней k-квазиотделимости пары вершин в $(X \to R)$-гиперсети $S = (X, V, R)$ полиномиально вычислима.*

Доказательство. По $(X \to R)$-гиперсети $S=(X, V, R)$ построим ориентированный граф $GS=(X, E)$ следующим образом. Множество вершин орграфа GS совпадает с множеством вершин в S. Пара вершин x и y в GS соединяется дугой (x, y) в GS, если и только если в гиперсети S существует ребро $r \in R$, сильно инцидентное вершине x и слабо инцидентное вершине y. Так как в $(X \to R)$-гиперсети каждому ребру

сильно инцидентна не более чем одна вершина, то при внутреннем удалении произвольной вершины x удалению инцидентных ей ребер в S соответствует удаление дуг, исходящих из вершины x в орграфе GS. С другой стороны, любому квазимаршруту в S взаимно однозначно соответствует ормаршрут в GS между той же парой вершин. Следовательно, разделяющее множество вершин в GS является разделяющим множеством (внутреннее удаление вершин) для тех же квазидостижимых вершин в гиперсети S. Теорема доказана.

Теорема 4.36. *Задача вычисления k — V-отделимости пары вершин в $(R \to V)$-гиперсети $S=(X, V, R)$ полиномиально вычислима.*

Доказательство. В $(R \to V)$-гиперсети S каждой ветви инцидентно не более чем одно ребро, поэтому $k_V(S) \geq \lambda(WS)$. С другой стороны, пусть $\lambda(WS)$—минимальное разделяющее множество ребер вторичной сети $WS = (X, B)$ $(R \to V)$-гиперсети $S = (X, V, R)$. Тогда в силу утверждения теоремы каждому ребру r из разделяющего множества R' можно сопоставить единственную инцидентную ему ветвь $v \in V$, удаление которой разрушает данное ребро r. Отсюда следует $\lambda(WS) \geq k_V(S)$. Теорема доказана.

Можно предположить, что задачи вычисления k — V-квазиотделимости и слабой k — V-отделимости пары вершин в $(R \to V)$-гиперсети также решаются за полиномиальное время. Например, если $(R \to V)$-гиперсеть S одновременно является $\{V \to R\}$-гиперсетью, то полиномиальность вычисления k — V-квазиотделимости пары вершин в S можно показать, переходя от гиперсети S к смешанному графу G из п. 4.5.

Теорема 4.37. *Задача вычисления внешней k-отделимоста пары вершин в $(R > X)$-гиперсети решается за полиномиальное время.*

Доказательство. Аналогично доказательству теоремы 4.36 за тем лишь исключением, что разделяющему (внешнее удаление вершины) множеству вершин в S взаимно однозначно соответствует множество разделяющих ребер в графе $\overline{WS} = (X', R')$, который получается из вторичной сети $WS = (X, R)$ гиперсети S отождествлением вершин, инцидентных такому ребру r, для которого не существует слабо инцидентных вершин из X. Очевидно, что в общем случае некоторые вершины внешне k-неотделимы при любом k. В этом случае в графе $\overline{WS}$ эти вершины будут отождествлены. Теорема доказана.

Можно показать, что для некоторых тривиальных классов гиперсетей (например, для $(V \to R)$-гиперсетей) все задачи вычисления k-отделимости решаются за полиномиальное время.

4.5.7. Задачи синтеза оптимальных гиперсетей с заданной связностью

Эти задачи широко используются в теории коммуникаций. Поэтому, несмотря на *NP*-полноту большинства задач синтеза гиперсетей, в работе приводятся некоторые эвристические и точные алгоритмы синтеза гиперсетей, удовлетворяющих заданным характеристикам связности. Кроме того, анализируются наиболее общие схемы методов, используемых при составлении различных алгоритмов синтеза.

7.1. Рассмотрим основные параметры задач синтеза.

1°. Исходные данные. В задачах синтеза могут быть заданы: множества вершин X и Y, первичная сеть $PS = (X, V)$, вторичная сеть $WS = (Y, R)$, однозначное отображение f: $Y \rightarrow X$ ($|X| \geq |Y|$). Каждое из приведенных данных может либо присутствовать в постановке задачи, либо отсутствовать. Наличие множеств X и Y обязательно. Кроме того, ветви, ребра и вершины гиперсети могут быть взвешены. Последние данные обычно фигурируют в задачах оптимального синтеза.

2°. Характеристики связности (отделимости) описаны в п. 5. Таким образом, синтезируемая гиперсеть должна обладать заданными значениями одной или нескольких характеристик связности.

3°. Отделимость вершин устанавливается выделенными вершинами или парой вершин.

4°. Значение связности устанавливается максимальное или заданное.

5°. В том случае, когда заданы первичная сеть $PS=(X, V)$ и вторичная $WS=(Y, R)$, для достижения необходимого значения связности синтезируемой гиперсети возможны добавления ребер, ветвей или распараллеливание ребер.

6°. В тех случаях, когда неизвестны первичная или вторичная сети, возможны ограничения на число их ветвей, ребер, диаметр и другие характеристики. Кроме того, в синтезируемой гиперсети могут быть ограничения на число элементов, инцидентных другим элементам гиперсети. Например, число ребер, инцидентных любой ветви, не превосходит заданной величины.

7°. В процессе синтеза гиперсети с заданной связностью часто возникает необходимость в оптимизации значения некоторой целевой функции, в частности, она может отсутствовать. Таким образом, предложенная классификация задач синтеза гиперсетей позволяет сформулировать весь круг проблем, связанных с оптимизацией гиперсетей, удовлетворяющих данным характеристикам связности.

7.2. Для любой задачи синтеза необходим собственный алгоритм, однако можно предложить общие схемы решения поставленных задач,

которые могут оказаться полезными при составлении частных алгоритмов. Все методы синтеза оптимальных структур гиперсетей можно подразделить на точные и приближенные. К приближенным методам прибегают в том случае, когда точные методы недостаточно эффективны (не дают решение за приемлемое время).

Точные методы:

1°. Метод ветвей и границ. Применительно к синтезу гиперсетей ветвление можно осуществлять, задавая для каждой ветви v либо $v \in F(R)$, либо $v \notin F(R)$. Такая схема может применяться в тех случаях, когда знание множества $F(R)$-ветвей, вошедших в оптимальную гиперсеть, позволяет эффективно построить ее самое.

2°. Метод направленного перебора. Суть метода направленного перебора заключается в следующем. Для каждого ребра $r \in R$ фиксируется множество цепей, которые могут фигурировать в оптимальной гиперсети в качестве $F(r)$. После этого осуществляется перебор всех вариантов структур гиперсетей. Применимость метода в значительной степени ограничивается необходимостью хранения в памяти ЭВМ множества допустимых цепей.

3°. Метод сведения задачи к уже известной. В некоторых случаях задачу синтеза оптимальной структуры гиперсети удается свести к уже известной, хорошо исследованной задаче. Ограниченность сферы применения этого метода очевидна.

Приближенные методы:

1°. Метод кратчайшего пути. Для каждого ребра $r \in R$ в качестве $F(r)$ выбирается кратчайший путь в графе PS между концевыми вершинами ребра r.

2°. Метод независимых путей. Если в графе вторичной сети WS имеется k кратных ребер между вершинами x, y и, кроме того, локальная связность вершин x, y в графе первичной сети PS не меньше k, то эти k кратных ребер можно реализовать по k независимым путям. Для того чтобы суммарная длина этих путей была минимальна, необходимо каждой ветви графа PS приписать «пропускную способность», равную длине этой ветви; затем воспользоваться алгоритмом нахождения потока минимальной стоимости.

3°. Метод кратчайшего пути с адаптацией. От метода кратчайшего пути он отличается тем, что после нахождения очередной трассы $F(r)$ длины ветвей, входящих в $F(r)$, изменяются по некоторому правилу. Если длины ветвей уменьшаются, то получающаяся гиперсеть имеет тенденцию к «сжатию» (уменьшению числа ветвей, по которым реализованы ребра). Если же длины ветней увеличиваются, то получающаяся гиперсеть имеет тенденцию к «расширению» (каждое

последующее ребро реализуется по пути, который должен иметь как можно меньше общих ветвей с путями реализации предыдущих ребер).

4°. Метод случайного поиска с адаптацией. Каждой ветви первичной сети $PS = (X, V)$ ставится в соответствие некоторая вероятность удаления данной ветви из PS. Разыгрывается t вариантов удаления ветвей из PS согласно заданным вероятностям. В полученных t графах $\{PS\}$ производится трассировка ребер WS (например, методом кратчайшего пути). Найденные гиперсети $\{S_i\}$ оцениваются с помощью целевой функции. Ветви PS_i, для которых значение целевой функции наилучшее, поощряются, т. е. вероятность удаления соответствующих ветвей из PS уменьшается, а для ветвей, входящих в PS_i с наихудшим значением целевой функции, она увеличивается. Процедура построения t случайных суграфов PS повторяется до тех пор, пока относительная погрешность не станет меньше заданной величины или процесс не стабилизируется.

5°. Метод гомеоморфного отображения. Сущность метода заключается в том, что отдельные фрагменты WS или вторичная сеть в целом отображаются в PS так, чтобы часть первичной сети PS (ветви этой части инцидентны ребрам WS) была гомеоморфной вторичной сети WS. Если такая трассировка возможна, то $\omega(S) = \omega(WS)$, причем здесь имеется в виду связность вершин, принадлежащих только WS.

6°. Метод замены трасс. Сначала трассировка ребер из WS осуществляется в PS произвольным способом (например, методом кратчайших путей), затем осуществляется перебор по всем ребрам вторичной сети WS, т. е. берется произвольное ребро, соответствующая ему трасса уничтожается и осуществляется новая трассировка с учетом оптимизации целевой функции и заданных ограничений. При такой трассировке можно использовать различные эвристические приемы или точные методы.

7°. Метод локальной оптимизации. Сначала строится некоторое приближенное решение задачи синтеза оптимальной структуры гиперсети. Затем по некоторым правилам перестраиваются отдельные фрагменты гиперсети. Если получающаяся гиперсеть оптимальнее старой, то она берется в качестве текущего приближенного решения и процедура повторяется заново уже по отношению к ней. Процесс прекращается, когда не удается построить более оптимальную гиперсеть.

8°. Усеченные точные методы. На базе точных методов (метод ветвей и границ, метод направленного перебора) можно строить приближенные методы решения задачи синтеза оптимальной структуры гиперсети, если в точном методе проводить не весь объем вычислений, а некоторую заранее обусловленную его часть. Например,

в методе ветвей и границ можно осуществлять ветвления до тех пор, пока наилучшая из построенных гиперсетей не окажется достаточно близкой к оптимальной.

7.3. Об одном алгоритме синтеза оптимальной гиперсети. Перейдем к описанию эвристического алгоритма решения задачи синтеза, дающего неплохие результаты на достаточно широком классе коммуникационных задач. Сформулируем необходимые определения. Пусть известны графы первичной сети $PS=(X, V)$, вторичной сети $WS=(Y, R)$. Обозначим: $\rho(v)$ или $\rho_{PS}(v)$ — длина ветви $v \in V$, $\rho(r)$ или $\rho_{WS}(r)$ — длина ребра $r \in R$, $\alpha(r)$—пропускная способность (емкость) ветви $v \in V$, $\alpha(r)$ — емкость ребра $r \in R$, $\omega(S)$ — вершинная связность гиперсети S.

Требуется построить гиперсеть $S=(PS, WS; \Phi)$, для которой выполняются условия

$$\omega(S) \geqslant k, \tag{4.28}$$

$$\forall v \in V \quad \sum_{r \in \Phi^{-1}(v)} \alpha(r) \leqslant \alpha(v) \tag{4.29}$$

и которая минимизирует функционал $\varphi(S)$ — стоимость гиперсети S.

Относительно вида функционала $\varphi(S)$ не формулируется никаких ограничений, поскольку он явным образом не фигурирует в предлагаемом ниже эвристическом алгоритме. Фактически алгоритм осуществляет поиск некоторого допустимого (удовлетворяющего выше описанным ограничениям) решения. При практическом использовании рекомендуется убедиться в том, что для данного конкретного функционала $\varphi(S)$ алгоритм дает приемлемые результаты.

Алгоритм А1:

Шаг 1. Если $\omega(PS) \geq k$ и $\omega(WS) \geq k$, то к шагу 2, иначе к шагу 15.

Шаг 2. Между всеми парами вершин $x, y \in X$ найти максимальные коммуникации в графах PS и WS, т. е. вычислить $\mu_{PS}(x, y)$ и $\mu_{WS}(x, y)$. Если существует пара вершин $x, y \in X$, для которой $\mu_{PS}(x, y) < \mu_{WS}(x, y)$, то к шагу 15, иначе к шагу 3.

Шаг 3. Реализуем ребра графа WS по кратчайшим путям в графе PS. Если выполняется условие (11), то к шагу 8, иначе к шагу 4.

Шаг 4. Для всех ветвей v_i вычислить коэффициент $\Delta_i = \sum_{r \in \Phi^{-1}(v_i)} \alpha(r) / \alpha(v_i)$. Если для некоторой ветви v_i $\Delta_i > 1$, то она называется перенасыщенной.

Шаг 5. Среди перенасыщенных ветвей выберем ветвь v_j с максимальным значением Δj и найдем ребро $r=(x, y) \in \Phi^{-1}(v_j)$ с минимальным значением $\alpha(r)$. Для каждой ветви v_j вычислим значение δ_j по формуле

$$\delta_j = \begin{cases} \Delta_j, \text{ если } v_j \in \Phi(r) \text{ и } \sum\limits_{r' \in \Phi^{-1}(v_j)} \alpha(r') \leqslant \alpha(v_j) - \alpha(r), \quad r' \neq r; \\ 1, \text{ если } v_j \notin \Phi(r) \text{ и } \sum\limits_{r' \in \Phi^{-1}(v_j)} \alpha(r') > \alpha(v_j) - \alpha(r), \quad r' \neq r; \\ \Delta_j - \dfrac{\alpha(r)}{\alpha(v_j)}, \text{ если } v_j \in \Phi(r) \text{ и ветвь } v_j \text{ не перенасыщена}; \\ 1, \text{ если } v_j \in \Phi(r) \text{ и } v_j \text{ перенасыщена}. \end{cases}$$

Также для каждой ветви v_j положим $\rho^*(v_j) = \rho(v_j)/(1 - \delta_j)$. Если $\delta_j = 1$, то очевидно, $\rho^*(v_j) = \infty$.

Шаг 6. Ищем кратчайший путь между вершинами x, y в графе PS с весами ветвей $\rho^*(v)$. Если этот путь имеет конечный вес, то к шагу 7, иначе к шагу 15.

Шаг 7. Осуществляем перетрассировку ребра r по найденному в шаге 6 кратчайшему пути. Заново вычисляем все Δ_j по формуле

$$\Delta_j = \sum_{r \in \Phi^{-1}(v_j)} \alpha(r)/\alpha(v_j).$$

Если в получившейся гиперсети отсутствуют перенасыщенные ветви, то к шагу 8, иначе к шагу 5.

Шаг 8. Если $\omega(S) \geq k$ (проверка осуществляется с помощью переборного алгоритма), то к шагу 14, иначе к шагу 9.

Шаг 9. С помощью переборного алгоритма находим минимальное сечение $\{x_1, ..., x_p\}$, $p < k$, гиперсети по вершинам.

Шаг 10. Найдем все ребра r, коммуникации которых содержат вершины из сечения $\{x_1, ..., x_p\}$, а концевые вершины лежат в разных компонентах связности S_1 и S_2 гиперсети $S\backslash\{x_1, ..., x_p\}$. Хотя бы одно такое ребро найдется, так как WS k-связен. Выберем среди них ребро $r = (x, y)$ с наибольшим значением $\alpha(r)$.

Шаг 11. Для каждой ветви v_j вычислим δ_j по формуле

$$\delta_j = \begin{cases} 1, \text{ если } v_j \notin \Phi(r) \text{ и } \sum\limits_{r' \in \Phi^{-1}(v_j)} \alpha(r') > \alpha(v_j) - \alpha(r), \quad r' \neq r; \\ \dfrac{1}{\alpha(v_j)} \sum\limits_{r' \in \Phi^{-1}(v_j)} \alpha(r) \text{ во всех остальных случаях.} \end{cases}$$

Также для каждой ветви δ_j положим $\rho^*(v_j) = \rho(v_j)/(1 - \delta_j)$. Если $\delta_j = 1$, то $\rho^*(\delta_j) = \infty$.

Шаг 12. Найдем кратчайший путь между вершинами x, y в графе $PS\backslash\{x_1, ..., x_p\}$ с весами ветвей $\rho^*(v)$. Если этот путь имеет конечный вес, то к п. шагу, иначе к шагу 15.

Шаг 13. Осуществляем перетрассировку ребра связност r по найденному в п. шагу кратчайшему пути. Если исчерпаны все ребра, коммуникации которых содержат вершины из сечения $\{x_1, ..., x_p\}$, а

концевые вершины лежат в разных компонентах связности S_1 и S_2 гиперсети $S\backslash\{x_1, ..., x_p\}$, то к шагу 9, иначе к шагу 10. Повторяем шаги 9—13 до тех пор, пока не будет найдена гиперсеть, удовлетворяющая условиям (10), (11), или не повторится минимальное сечение. В первом случае к шагу 14, во втором —к шагу 15.

Шаг 14. Гиперсеть, удовлетворяющая условиям (10), (11), найдена.

Шаг 15. Допустимое решение не найдено. Применение переборного алгоритма здесь оправдано тем, что связность гиперсетей для практических целей небольшая.

4.5.8. Алгоритмы синтеза гиперсетей с заданной вершинной связностью

Задачи синтеза гиперсетей имеют различную сложность решения. Покажем на примерах, как незначительное изменение в постановке задачи влияет па сложность ее решения. Пусть заданы первичная сеть $PS = (X, V)$, для которой $\omega(PS) \geq 2$, и вторичная сеть $WS = (Y, R)$— простой цикл, и пусть $|X|=|Y|$. Необходимо найти гиперсеть $S=(PS, WS; \Phi)$, у которой вершинная связность равна 2. Если известно отображение $f:Y\rightarrow X$, то легко решается вопрос о наличии допустимого решения, в противном случае к поставленной задаче сводится задача поиска гамильтонова цикла, т. е. задача синтеза становится *NP*-полной.

Очевидным фактом является то, что k-связность пары вершин в гиперсети S не может превосходить связности этой же пары вершин в первичной и вторичной сетях гиперсети $S = (PS, WS; \Phi)$. Например, на рис. 7 вершины s и t 2-связны в S, но 3-связны в PS и WS. Возникает вопрос, всегда ли можно найти гиперсеть S', для которой $\omega(S') = \min(\omega(PS), (\omega(WS))$? Оказывается, что такая гиперссть не всегда существует (рис. 8).

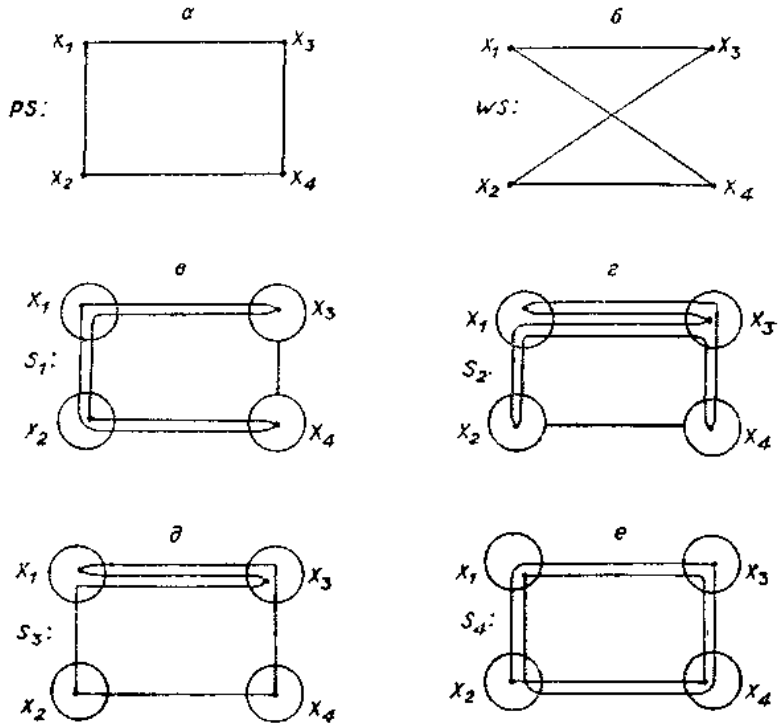


Рис. 8. Односвязные гиперсети.

Гиперсети S_1, S_2 очевидно, односвязны; S_3 разрушается при удалении вершины x_1 или x_3, а S_4 становится несвязной при удалении любой вершины. Из приведенных примеров видно, что для данных *PS* и *WS* не существует гиперсети, имеющей связность, равную двум. В связи с этим возникает следующая задача синтеза гиперсети *S*.

Пусть заданы первичная сеть $PS = (X, V)$ и вторичная $WS = (Y, R)$. Требуется найти гиперсеть $S=(X, V, R')$ такую, что

$\omega(S) = \min(\alpha(PS), (\omega(WS))$. Причем ко вторичной сети *WS* разрешается добавлять ребра. Критерий эффективности — минимум суммарной длины ребер. На рис. 9 приведены примеры гиперсетей S'_1 и S'_3, которые становятся двусвязными при добавлении новых ребер к S_1 и S_3.

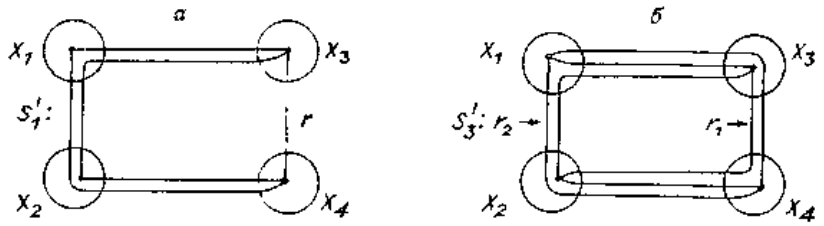


Рис. 9. Двухсвязные гиперсети.

Причем к S_1 (см. рис. 8, *в)* добавлено ребро между вершинами x_3, x_4, несмежными в *WS;* к S_3 (см. рис. 8,*д)* — ребра (x_1, x_4) и (x_2, x_3), соединяющие уже смежные в *WS* вершины, т. е. некоторые ребра в *WS* распараллеливаются и реализуются по независимым маршрутам в *PS*.

Поставленная задача, очевидно, имеет решение, так как, добавив ребра ко всем парам *V*-смежпых вершин в гиперсети *S,* получим гиперсеть *S'*, для которой $\omega(S')=\omega(PS)$, но такое решение будет, видимо, неоптимальпым. Прием повышения связности гиперсети за счет добавления новых ребер не всегда пригоден, так как изменяется структура вторичной сети. Поэтому возникает задача увеличения связности гиперсети за счет распараллеливания ребер вторичной сети. В этом случае структура последней остается неизменной.

8.1. Пусть заданы первичная сеть $PS = (X, V)$ и вторичная $WS = (Y, R)$. Предположим, что при оптимальной (с точки зрения максимума связности) реализации *WS* в *PS* получится гиперсеть *S,* для которой $\omega(S) < \min(\omega(PS), (\omega(WS))$. Тогда имеет место

Теорема 4.38. *Для любых PS и WS всегда можно построить гиперсеть S (распараллеливая некоторые ребра WS) такую, что* $\omega(S)= \min(\omega(PS), (\omega(WS))$.

Доказательство. Предположим противное. Тогда в построенной гиперсети *S* найдется минимальное сечение $\{x_i\}$ такое, что $|\{x_i\}|< \min(\omega(PS), (\omega(WS))= k$. Так как $\omega(WS)\geq k$, то среди вершин $\{x_i\}$ найдется такая, которая слабо инцидентна некоторому ребру из *WS*. Причем концевые вершины x_l, x_t этого ребра будут лежать в разных компонентах связности гиперсети $S\backslash\{x_i\}$. Распараллелим найденное ребро (x_l, x_t) на два: $(x_l, x_t)^1$ и $(x_l, x_t)^2$. Первое из них реализуется по старой трассе, а второе — в графе $PS\backslash\{x_i\}$. Очевидно, что вершины x_l, x_t будут связны в $PS\backslash\{x_i\}$, так как $\omega(PS)\geq k$. Следовательно, во вновь полученной гиперсети *S'* вершины $\{x_i\}$ не являются минимальным сечением,— противоречие. Тем самым теорема доказана.

Теперь рассмотрим следующую задачу. Для заданных первичной *PS* и вторичной *WS* сетей найти гиперсеть $S' = (PS, WS;$ Ф), которая

удовлетворяет условиям (10), (11), а граф WS' получается из WS в результате распараллеливания некоторых ребер.

Алгоритм *А*2:

Шаг 1. Реализовать алгоритм *А*1. Если полученная гиперсеть удовлетворяет условиям (10), (11), то на шаг 6, иначе на шаг 2.

Шаг 2. Если гиперсеть S удовлетворяет условию (11), но не удовлетворяет условию (10) или оба условия нарушаются, то на шаг 3, иначе на шаг 4.

Шаг 3. В построенной гиперсети S найти минимальное сечение $\{x_i\}$. По множеству $\{x_i\}$ найти слабо инцидентное одной из вершин $\{x_i\}$ ребро $(x_l, x_t) \in R$ и с помощью известных алгоритмов найти в PS две независимые цепи между вершинами x_l, x_t. Емкость $\alpha(r_{l,t})$ данного ребра распределяется по цепям в PS так, чтобы удовлетворялось условие (11). Если это невозможно, то на шаг 5, иначе повторить шаг 3. Как только связность S станет равной k, то на шаг 4.

Шаг 4. В гиперсети S найти перенасыщенную ветвь v (для которой $\alpha(v) > \sum_{r \in \Phi^{-1}(v)} \alpha(r)$). Найти инцидентные ребра $\Phi^{-1}(v)$ данной ветви v и выбрать среди них такое подмножество $R'' \subset \Phi^{-1}(v)$, чтобы

$$\alpha(v) \leqslant \sum_{r \in \Phi^{-1}(v) \setminus R''} \alpha(r).$$

Последовательно распараллеливая эти ребра и распределяя их емкости по дубликатам, попытаемся удовлетворить условию (11). Если это не удается, то на шаг 5, иначе шаг 4. Повторяется до тех пор, пока не будут проверены все ветви, переход на шаг 6.

Шаг 5. Допустимое решение не найдено.

Шаг 6. Гиперсеть S построена.

Сходимость алгоритма очевидна. Если ограничение (11) не является критическим, то гиперсеть с заданной связностью будет обязательно найдена. Кроме того, алгоритм позволяет находить гиперсеть с квазиминимальной стоимостью. Поставленная задача, вообще говоря, не всегда позволяет построить гиперсеть S со связностью, равной $\omega(PS)$, т. е. связность вторичной сети существенно влияет на связность S. Поэтому актуальна задача синтеза гиперсети S, для которой $\omega(S) = \omega(PS)$. Этого можно добиться добавлением новых ребер в WS.

8.2. Пусть заданы первичная сеть $PS = (X, V)$ и вторичная $WS = (Y, R)$. Требуется найти гиперсеть $S = (PS, WS; \Phi)$ такую, чтобы $\omega(S) = k \leq \omega(PS)$ и выполнялось условие (11), где $WS' = (Y, R \cup U)$, а U — множество добавленных ребер. Причем при добавлении очередного ребра емкости ребер WS' пересчитываются.

Алгоритм А3:

Шаг 1. Реализовать алгоритм А1. Если полученная гиперсеть удовлетворяет условию (11) и $\omega(S) = k$, то на шаг 5 иначе на шаг 2.

Шаг 2. Если гиперсеть S не удовлетворяет условию (11), то на шаг 4, иначе на шаг 3.

Шаг 3. В S найти минимальное сечение $\{x_i\}$. Выделить в $S\backslash\{x_i\}$ компоненты связности WS^1 и WS^2. Найти в них пару ближайших несмежных вершин $x_l \in WS^1$ и $x_t \in WS^2$. Соединить вершины x_l и x_t ребром. Пересчитать емкости ребер с помощью заданной процедуры расчета емкостей. Перейти на шаг 1.

Шаг 4. Допустимое решение не существует.

Шаг 5. Гиперсеть S найдена.

Сходимость алгоритма очевидна. Отсутствие допустимого решения определяется только условием (11).

В заключение раздела рассмотрим быстрый алгоритм синтеза гиперсети с достаточно большой связностью.

8.3. Пусть заданы первичная сеть $PS = (X, V)$ и вторичная $WS = (Y, R)$. Требуется найти гиперсеть $S=(PS, WS; Ф)$ с максимально возможной связностью. Вес каждой ветви и вершины равен единице.

Алгоритм А4:

Шаг 1. Упорядочим ребра $r = (x_i, x_j) \in R$ по убыванию расстояния $\rho(x_i, x_j)$ в PS. Получим список ребер $\{r_i\}$, r=1, 2, ..., m; i:=l.

Шаг 2. Реализуем r, в PS по кратчайшему пути (кратчайший по суммарному весу вершин и ребер).

Шаг 3. Увеличим вес вершин и ребер найденной трассы ребра r_i. Вес увеличивается на некоторую величину $z(S)$, которая зависит от текущего состояния гиперсети S.

Шаг 4. Если все ребра WS' реализованы, то при $\omega(S)=\min(\omega(PS), (\omega(WS))$ — на шаг 10, иначе на шаг 5, если не реализованы — $i := i + 1$, переход на шаг 2.

Шаг 5. С помощью переборного алгоритма найти минимальное сечение по вершинам $\{x_l\}$.

Шаг 6. Найти все ребра r, слабо инцидентные вершинам из $\{x_l\}$, такие, что их концевые вершины лежат в разных компонентах связности S_1 и S_2 гиперсети $S\backslash\{x_l\}$. Хотя бы одно такое ребро обязательно найдется, так как WS k-связен.

Шаг 7. В первичной сети $PS\backslash\{x_l\}$ между всеми парами концевых вершин x_i, x_j найдем кратчайшие пути. (Путь обязательно найдется, так как $\omega(PS) \geq k$.)

Шаг 8. Выберем такую пару x_i и x_j для которой следующая разность минимальна:

$$\rho_{PS\setminus\{x_l\}}(x_i, x_j) - \rho_{PS}(x_i, x_j).$$

Шаг 9. Реализуем ребро (x_i, x_j) по новой трассе, тогда сечение, разделяющее компоненты S_1 и S_2, увеличится. Повторяем шаги 3—6 до тех пор, пока связность $\omega(S)$ увеличивается, переход на шаг 10.
Шаг 10. Гиперсеть с квазимаксимальной связностью найдена.

4.5.9. О построении гиперсетей с заданной квазисвязностью

Рассматриваются задачи синтеза в классе $(X \to R)$-гиперсетей с заданной частичной квазисвязностью и ограничением на ранг связывающих квазимаршрутов. Во всех этих задачах считается заданной первичная сеть $PS=(X, V)$ синтезируемой гиперсети.

9.1. Рассмотрим T-гиперсеть, в первичной сети $PS = (X, V)$ которой выделен корень $x_0 \in X$. Вторичная сеть $WS = (X, R)$ содержит то же множество вершин и $|R|$ ребер, причем каждое ребро $r \in R$ инцидентно x_0 и слабо инцидентно остальным вершинам $X\setminus\{x_0\}$. Из определения T-гиперсети следует, что каждому ребру $r \in R$ сопоставлено некоторое корневое дерево T_r в графе $PS=(X, V)$.

Обозначим $\lambda_i(\omega_i)$— реберную (вершинную) связность вершины $x_i \in X$ с вершиной x_0 в графе PS. Тогда корневая реберная и вершинная связности графа PS определяются следующим образом:

$$\lambda = \max_{x_i \in X - x_0} \lambda_i, \quad \omega = \max_{x_i \in X - x_0} \omega_i. \tag{4.30}$$

Два ребра r_1 и r_2 гиперсети $S=(X, V, R)$ называются T_λ-независимыми, если для любой вершины $x \in X$ существуют два независимых по ветвям квазимаршрута из x_0 в x, принадлежащих r_1 и r_2. Соответствующие квазимаршруты назовем T_λ-независимыми. T_ω-независимость определяется аналогично, т. е. с учетом независимости квазимаршрутов по вершинам. В дальнейшем будем говорить просто о T-независимости, если соответствующие утверждения справедливы для частичной квазисвязности T-гиперсети как по ветвям, так и по вершинам.

Множество ребер T-гиперсети, инцидентных вершине x_0, будем называть T-покрытием в $S = (X, V, R)$, если они попарно T-независимы. Полным T-покрытием гиперсети S будем называть такое T-покрытие $\Theta(S)$, что $|R|$ равно корневой связности PS и каждая вершина $x \in X$

слабо инцидентна k ребрам, где k равно числу связности этой вершины с корнем x_0 в PS.

Теорема 4.39. *Для любой первичной сети $PS = (X, V)$ с корнем x_0 существует T-гиперсеть $S = (X, V, R)$, в которой множество R является полным T-покрытием.*

Существуют доказательство теоремы как для T_ω-, так и для T_λ-покрытия. В последнем случае приведены два алгоритма синтеза. Здесь рассмотрен алгоритм, который позволяет находить полное T-покрытие, т. е. алгоритм синтеза T-гинерсети, где частичная квазиотделимость (как по ветвям, так и по вершинам) любой вершины x от вершины x_0 будет максимально возможной, а ранг любого (x_0, x)-квазимаршрута равен единице.

Алгоритм А5.

Шаг 1. Для каждой вершины $i \in X$, $x \neq x_0$ вычислим $k(x)$-связность вершины x с x_0, $k(x_0) := \max\limits_{x \in X - x_0} k(x)$, $i = 1$, $X_i := x_0$, $V_i := \varnothing$, $T_i := (X_i, T_i)$, $\Theta(S) := \{T_1, \ldots, T_{k(x0)}\}$, $\forall x$ $l(x) := 0$. Граф $PS = (X, V)$ преобразуется в орграф $PS' = (X, V')$ заменой каждой ветви на пару противоположно ориентированных дуг, а ветви, инцидентной x_0, на дуги, ориентированные из x_0.

Шаг 2. В графе PS' выбираем дугу $v = (x, \overline{x})$ из X_i в $X - X_i$ такую, что дерево $T_i = (X_i \cup \overline{x}, V_i \cup v)$ не нарушает T-покрытие $\Theta(S)$. Причем количество независимых цепей из x_0 в $\overline{x}$ в графе $PS'' = (X, V' - V_i)$ уменьшится по сравнению с графом PS' ровно на единицу и степень вершины x_0 в графе PS'' станет не меньше $k(x_0) - 1$. Если такая дуга не найдена, то на шаг 4, иначе на шаг 3.

Шаг 3. $V' := V' - v$, $T_i := (X_i \cup \overline{x}, V_i \cup v)$, переход на шаг 2.

Шаг 4. Проверяем последовательно все существующие и вновь получаемые висячие вершины дерева T_i на выполнение неравенства

$$k(x) - l(x) < k(x_0) - i. \qquad (4.31)$$

Если это неравенство для некоторой висячей в T_i вершины x выполняется, то $V' := V' \cup (y, x)$, $((y, x) \in T_i)$, $T_i := (X_i - x, V_i - (y, x))$, иначе переходим к следующей висячей вершине. Процесс заканчивается, если висячих вершин в T_i, удовлетворяющих неравенству (4.31), не остается.

Шаг 5. $\forall x \in T_i$ $l(x) := l(x) + 1$, $i := i + 1$, если $i > k(x_0)$, то на шаг 6, иначе на шаг 2.

Шаг 6. Полное T-покрытие найдено.

Проверка правильности подключения новой вершины в T_i в шаге 2 осуществляется путем проверки числа независимых путей из x_0 в x в графе PS''. После того как найдено $\Theta(S)$-покрытие, каждому ребру

$r \in R \subset WS$ будет сопоставлено дерево из $\Theta(S)$. Таким образом, найдена T-гиперсеть, удовлетворяющая условиям задачи синтеза.

9.2. Применение теории гиперсетей в задачах синтеза коммуникационных сетей инициирует еще несколько важных постановок задач синтеза гиперсетей с заданной частичной квазисвязностью. Пусть задана первичная сеть $PS = (X, V)$ и некоторое подмножество вершин $x_0 \bigcup X'$, где $X' = (x_1, .. ., x_k) \subset X$. Требуется найти гиперсеть $S = (X, V, R)$ такую, чтобы между вершинами x_0 и $x_i \in X'$ имела место частичная k_i-квазинеотделимость по ветвям (вершинам), где $k_i \leq \lambda_i$ ($k_i \leq \omega$,). Решение поставленной задачи можно получить с помощью алгоритма А5. Действительно, положим $k(x_0) = \max\limits_{x \in X'} k(x)$ и найдем $\Theta(S)$-покрытие. Затем, поочередно рассматривая висячие вершины $x_3 \in T_i$, будем их удалять из T_i, если они не принадлежат X' или число слабо инцидентных ребер превышает k_j.

Часто может возникнуть ограничение на число слабо инцидентных вершин любому ребру $r \in R$. Данное ограничение может быть учтено в предыдущих задачах путем разбиения каждого ребра r на две части r' и r'' по некоторой вершине x так, что x_0 инцидентно r', а x инцидентно r'' и слабо инцидентно r'. Затем процесс разбиения можно продолжить.

В предыдущих задачах синтеза T-гиперсетей предполагалась частичная k-квазинеотделимость корня x_0 от других вершин гиперсети S. Если потребовать попарную частичную k-квазинеотделимость в синтезируемой гиперсети, то задача решается тривиально. Достаточно потребовать, чтобы гиперссть одновременно принадлежала к классам $(V \rightarrow R)$- и $\{X — R\}$-гиперсетей, т. е. вторичная сеть WS будет изоморфна некоторому суграфу PS' графа PS, причем PS' должен удовлетворять заданным характеристикам связности. Данная задача представляет особый интерес, когда есть ограничение на ранг независимых квазимарштрутов, соединяющих различные вершины гиперсети. В этом случае, поочередно объявляя каждую вершину гиперсети корнем, решаем первую задачу (алгоритм А5), затем последовательно удаляем висячие ребра в покрывающих деревьях, отслеживая выполнение ограничения и условия задачи.

4.5.10. Заключение

Развитие теории гиперсетей имеет фундаментальное значение для исследования и проектирования иерархических коммуникационных

систем сетевой структуры, т. е. таких, в которых иерархия определяется вложением одной подсистемы в другую. Перечислим некоторые важные направления развития гиперсетей.

10.1. Маршруты в гиперсетях. В разд. 4.5.3 сформулированы и решены некоторые задачи поиска оптимальных маршрутов. Представляют интерес задачи, связанные с поиском обходов элементов гиперсетей, в частности бесповторных обходов ребер, ветвей или вершил, а также обобщения указанных задач. Важны также задачи, связанные с метрикой гиперсетей (поиск центров, медиан и т. п.). По-видимому, задачи данного направления в основном разрешимы за полиномиальное время.

10.2. Упаковки и покрытия. Так же как и в теории гиперграфов, для гиперсетей определяются упаковки и покрытия, т. е. для любой пары множеств из *X, V, R* и соответствующих отношений инцидепции или смежности можно определить понятия упаковки и покрытия. Кроме того, в гиперсетях возможно определение этих понятий, когда одно множество сопоставляется двум другим. В этом направлении возникают принципиально новые задачи.

10.3. Частичные гиперсети. Удаляя различные элементы из гиперсетей, можно получать всевозможные частичные гиперсети с теми или иными свойствами. Например, возможны следующие задачи: найти минимальную (по числу вершин, ветвей или ребер) частичную гиперсеть, связывающую заданное множество вершин или ветвей; найти компоненты связности, блоки и части гиперсетей с заданной связностью.

10.4. Эквивалентные гиперсети. Две гиперсети $S' = (X', V', R')$ и $S'' = (X'', V'', R'')$ эквивалентны, если и только если изоморфны PS', PS'' и WS', WS'', кроме того, $X' \leftrightarrow X''$. Две гиперсети S' и S'' равносильны, если WS' изоморфен WS'' при $X' \leftrightarrow X''$, и равнозначны, если PS' изоморфен PS''. Основные задачи по этому направлению связаны с поиском эквивалентных (равносильных, равнозначных) гиперсетей с заданным свойством (набором характеристик).

10.5. Планарность. Гиперсеть *S* называется *R*-планарной, если ее можно расположить на плоскости так, чтобы ребра не пересекались (пересечение ребер запрещается как по ветвям, так и внутри вершин; см. геометрическое представление гиперсети на рис. 6. Здесь ребра пересекаются в вершине *1* и не пересекаются в других вершинах. Кроме того, ребро *(s,* 4) пересекается с ребром (s, 6) по ветви (5, 6)).

С понятием планарности гиперсетей связан ряд задач: найти необходимые и достаточные условия *R*-планарности гиперсети; сформулировать алгоритм укладки *R*-планарной гиперсети; найти эквивалентную данной гиперсеть, которая была бы *R*-планарной, если

планарны *PS* и *WS*. Возможны и другие задачи, связанные с минимизацией числа пересечений, и т. п.

5. Решение коммуникационных задач формирования предписаний методами эвристического поиска

5.1. Вводные замечания

Основой всех стратегий решения коммуникационных задач в области формирования предписаний является метод поиска. Для коммуникационных задач практической степени сложности поиск должен направляться с помощью некоторого механизма управления, причем мы отвергаем возможность использования исчерпывающего поиска или поиска, управляемого случайным механизмом. Решающим аргументом в пользу такого решения является тот факт, что в принципе пространство поиска может быть бесконечным.

Эвристически эффективная стратегия представляет собой совокупность двух механизмов: *механизма генерации элементов* пространства поиска — кандидатов в решающую последовательность и *механизма управления генерацией,* работа которого основывается на информации о пространстве поиска, а также свойствах цели и уже построенных элементов.

Степень эвристической эффективности стратегий определяется прежде всего тем, как извлекаются указанная информация и свойства и как они используются механизмом управления в поиске решения формирования коммуникационных предписаний. Таким образом, мы можем выделить в каждой стратегии ***синтаксическую,* или структурную компоненту (генерация), и *семантическую,* или смысловую компоненту (управление поиском)**. Важно отметить, что семантическая компонента вносится в процесс решения коммуникационной задачи *стратегией* во всех предписаниях, кроме семантических коммуникационных сетей (2.3)

При построении стратегий формирования коммуникационных предписаний мы стоим перед необходимостью разрешить противоречие между *качеством решения* и ею *эффективностью,* где под *качествам* мы понимаем свойства решения (длина решающего пути, количество вершин в решающем графе и т. д.), а под

эффективностью — количество ресурсов, затраченное при поиске решения (общее число генерированных вершин и т. д).
Поскольку нашей задачей является построение эвристически эффективных стратегий формирования коммуникационных предписаний, мы не рассматриваем стратегии типа поиска в ширину, которые, хотя и позволяют найти все решения и, следовательно, наиболее качественные, представляют собой модель исчерпывающего поиска, а потому эвристически неэффективны. План настоящего раздела состоит в том, чтобы для каждого из предписаний в виде коммуникационной задачи эвристического поиска построить стратегию формирования коммуникационных предписаний, позволяющую найти решение наилучшего качества, если оно имеется, а затем исследовать пути повышения эвристической эффективности, возможно, с риском потерять решение наилучшего качества.

5.2. Стратегии формирования коммуникационных предписаний, основанные на поиске в графе вывода

5.2.1. Алгоритм поиска решающего графа

В п.2.6 мы рассмотрели одно обобщенное декларативное предписание в виде графа вывода, из которого в качестве частных случаев вытекают предписания в виде пропозиционального графа, графа доказательства предписаний и графа пространства состояний. В обобщенном декларативном предписании решение коммуникационной задачи сводится к поиску *беспосылочного редукционного вывода (решающего графа) в графе вывода*. Мы опишем алгоритм поиска решающего графа в графе вывода как механизм генерации элементов пространства поиска, а затем рассмотрим некоторые общие характеристики этого алгоритма, в том числе касающиеся механизма управления.
Этот алгоритм носит скорее иллюстративный характер: во-первых, он демонстрирует стиль построения алгоритмов в частных теоретико-графических представлениях, во-вторых, на примере этого алгоритма мы покажем общий сценарий, по которому в последующих параграфах будут рассматриваться стратегии формирования коммуникационных предписаний в этом классе предписаний. Мы не утверждаем, что этот алгоритм окажется наиболее эффективным (в отношении быстродействия и требуемой памяти) для всех частных случаев.
Введем два направления поиска — *прямое,* т. е. от посылок к заключению *(F),* и *обратное (B)*. Будем называть *вывод*

генерированным, если генерированы все его предписания и акты вывода предписаний, даже если это произошло в процессе генерации другого вывода. На каждом шаге мы будем выбирать одно из направлений поиска, причем, если не окажется ни одного вывода предписания — кандидата на генерацию, то алгоритм терпит неудачу. В противном случае он выбирает наилучшего кандидата в текущем направлении и генерирует все ранее негенерированные предписания применением одного ранее негенерированного акта вывода, принадлежащего этому направлению. Если при этом не получено решение, то алгоритм вновь выбирает направление, выбирает кандидата на вывод предписания, и т. д. до тех пор, пока не будет получено решение. Соответственно направлениям поиска мы выделяем два множества *F* и *B*. *F* содержит все уже генерированные предписания и акты вывода предписаний, принадлежащие беспосылочным выводам (БВ). *B* содержит все уже генерированные предписания и акты вывода предписаний, принадлежащие редукционным выводам (РВ). Заметим, что пересечение *F* и *B* не обязательно должно быть пустым до получения решения.

Определим формально кандидаты на генерацию в направлениях *F* и *B*. Пусть существует акт вывода, не принадлежащий *F,* но все посылки которого принадлежат *F*. Тогда любой БВ, состоящий из этого акта вывода, его заключения и, для любой посылки акта вывода, точно одного БВ, содержащегося в *F* и имеющего эту посылку как заключение, называется *кандидатом на генерацию в направлении F*. Пусть D_0 — РВ, содержащийся в *B*, и каждая посылка D_0 есть заключение некоторого акта вывода, не принадлежащего *B*. Тогда любой РВ, состоящий из точно одного такого акта вывода, его посылок и D_0, называется *кандидатом на генерацию в направлении B*.

Предположим, что на выводах определена мера качества предписаний, так что для любой совокупности выводов можно выбрать вывод наилучшего качества (ни один из выводов, принадлежащий этой совокупности, не имеет лучшего качества, чем выбранный).

Тогда алгоритм поиска беспосылочного редукционного вывода может быть представлен следующими шагами:

1. *F* и *B* в начальном состоянии — пустые множества.
2. Выбрать направление *X* генерации *(F* или *B)*.
3. Если нет кандидатов на генерацию в направлении *X,* неудача. В противном случае продолжать.
4. Выбрать и генерировать вывод-кандидат *D* для *X,* имеющий наилучшее качество в направлении *X*. Генерация производится путем добавления к множеству *X* одного акта вывода и всех определений в *D,* еще не принадлежащих *X*.

5. Добавить к *F* и *B* все акты вывода и предписания, принадлежащие уже сгенерированному БВ предписания в *B*.
6. Успешное решение, если а) решающий вывод содержится в *F* или *B*, б) нет кандидата на генерацию в направлении *F* или *B*, имеющего лучшее качество, чем *D*. В противном случае перейти к шагу 2.

На рис. 5.1 представлен пример состояния пространава поиска в начале цикла работы алгоритма.

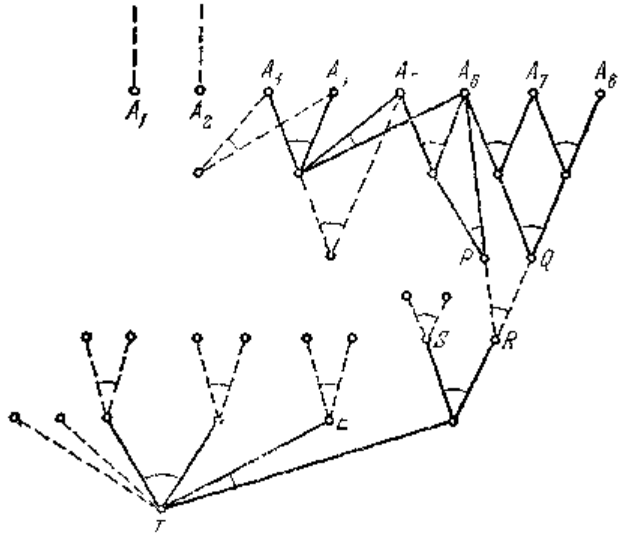


Рис. 5.1. Состояние пространства поиска в начале цикла работы алгоритма.

Сплошные линии — дуги, принадлежащие уже сгенерированным выводам. Пунктирные линии — дуги, принадлежащие выводам, которые являются кандидатами на генерацию. Остальные выводы на рисунке не показаны. У аксиом A_1 и A_2 пунктиром показаны фиктивные акты вывода (по определению, A_1 и A_2 являются их заключениями). У A_3—A_8 фиктивные акты вывода не показаны. Акт вывода с посылками *P* и *Q* и заключением *R* является частью кандидата на генерацию как в направлении *F*, так и в направлении *B*. Множества *F* и *B* пока не имеют общих элементов.

Пусть на шаге 2 алгоритма выбрано направление *B* и выбранным кандидатом на генерацию является РВ с посылками *E, S, P, Q*. На шаге

4 к *B* добавляются предписания *P, Q* и акт вывода, содержащий их как посылки. Этот же акт вывода добавляется к *F* вместе с заключением *R*. Одновременно к *B* добавляются все предписания и акты вывода в выводах, имеющих *P* и *Q* как заключения.

5.2.2. Свойства алгоритма

Полнота. *Стратегия поиска (формирования) предписаний* называется *полной,* если она найдет решение всегда, когда оно существует. *Поисковая стратегия* будет *исчерпывающей* для направления *X,* если она генерирует все предписания и акты вывода предписаний, которые могут быть генерированы в направлении *X*. Очевидно, что любая исчерпывающая поисковая стратегия полна. Однако и в случае бесконечного количества альтернатив вывода (бесконечного дизъюнктивного ветвления) поисковая стратегия может быть полна.

Упорядочение выводов по качеству называется *квазиконечным* для направления *X,* если для любого вывода *D* существует только конечное число выводов, имеющих качество, лучшее или равное качеству *D*.

Теорема 5.1. *Любая двунаправленная поисковая стратегия, использующая квазиконечное упорядочение выводов по качеству для направления X, полна при условии, что она не становится однонаправленной в противоположном X направлении.*

Доказательство. Пусть D^* — решение, не порожденное поисковой стратегией. Тогда существует некоторый $D \subset D^*$, который на каком-то шаге стал кандидатом на генерацию в направлении *X,* но не был генерирован. Тогда, если поисковая стратегия не становится однонаправленной в направлении *X* и не нашла другого решения, она должна генерировать в направлении *X* кандидаты на вывод D_1, D_2, ..., D_n, D_{n+1}, . . ., не оканчивая работы. Но по определению квазиконечности один из них, например D_n, должен иметь качество, худшее, чем *D,* что противоречит нашему предположению.

Допустимость. Поисковая стратегия называется *допустимой,* если она находит наилучшее решение всегда, когда оно имеется. Определим понятие наилучшего решения. Пусть f — действительная неотрицательная функция, определенная на множестве выводов. Назовем f *оценочной функцией,* если D_1 имеет *лучшее качество,* чем D_2, когда

$$f(D_1) < f(D_2). \qquad (5.1)$$

D_1 и D_2 *равнокачественны,* когда

$$f(D_1) = f(D_2). \qquad (5.2)$$

Оценочная функция *монотонна,* если из $D' \subseteq D$ следует

$$f(D') \le f(D). \qquad (5.3)$$

Монотонная оценочная функция характеризует меру сложности решающего вывода. Существует несколько вариантов меры сложности:

— *размер* — число предписаний (директив) в выводе;

— *уровень* — наибольшее число предписаний (директив), измеряемое вдоль какой-либо ветви вывода;

— *суммарная сложность (стоимость)* — сумма всех сложностей (стоимостей), каждая из которых связана с одним из предписаний (директив) в выводе;

— *максимальная сложность (стоимость)* — наибольшая сумма сложностей (стоимостей), измеренная для всех предписаний (директив) вдоль какой-либо ветви вывода.

Решение называется *наилучшим,* если оно обладает наименьшей мерой сложности среди всех решающих выводов на графе вывода.

Диагональной поисковой стратегией (ДПС) называется стратегия, использующая такую оценочную функцию, что

$$h(D)=f(D)-g(D)\geq 0 \qquad (5.4)$$

для всех выводов D, $g(D)$ — мера сложности вывода D такая, что $g(D')\leq g(D)$, если $D'\subseteq D$. Функция h называется *эвристической функцией* и оценивает сложность части наилучшею решения D^*, дополнительную к D. Тогда оценочная функция может рассматриваться как оценка сложности наилучшею решения D^*, содержащего D.

Таким образом, ДПС выбирает и генерирует всегда тот из кандидатов на вывод, который имеет наименьшую оценочную функцию вида

$$f(D)=g(D)+h(D). \qquad (5.5)$$

Геометрическая интерпретация ДПС иллюстрируется на рис. 5.2.

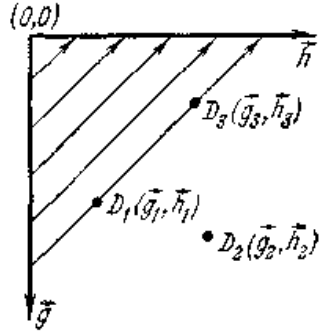


Рис. 5.2. Геометрическая интерпретация диагональной поисковой стратегии

Здесь точки в системе координат $(\vec{g},\vec{h})$ представляют собой акты вывода, порожденные поисковой стратегией в процессе выбора кандидата на вывод D с $g(D)=\vec{g}$ и $h(D)=\vec{h}$. В силу монотонности функции f и свойств ДПС поиск решения идет от *более коротких диагоналей к более длинным,* так что D_2 будет всегда порождаться после D_1. ДПС называется *ограниченной,* если для всех кандидатов на вывод D

$$f(D)\leq g(D^*), \qquad (5.6)$$

где $D\subseteq D^*$, D^* - решающий вывод.

Из приведенного определения очевидно, что оценочная функция для ограниченной ДПС монотонна и $h(D^*)=0$, т. е. в интерпретации рис. 5.2 все решающие выводы лежат на оси $\vec{g}$.

Теорема 5.2. *Ограниченная ДПС допустима.*

Доказательство. Предположим, что ограниченная ДПС нашла решение D^*, но существует решение D_0, лучшее, чем D^*, а окончание работы алгоритма произошло потому, что ни один кандидат на вывод в направлении X не имеет качества, лучшего, чем D^*. Тогда в момент окончания работы некоторый $D'_0\subset D_0$ является кандидатом на генерацию в направлении X. Однако в силу свойств ограниченной ДПС

$$f(D'_0)\leq g(D_0)<g(D^*)=f(D^*), \qquad (5.7)$$

что противоречит условию окончания работы алгоритма.

Следствие. *Двунаправленная ограниченная ДПС, использующая квазиконечное упорядочение выводов по качеству для направления X, допустима при условии, что она не становится однонаправленной в противоположном направлении.*

Оптимальность. *Допустимая стратегия* с эвристической функцией h_1 называется *оптимальной,* если она никогда не порождает большего количества предписаний (директив), чем другая допустимая стратегия с такой эвристической функцией h_2, что $h_1(D)\geq h_2(D)$ для всех выводов и $h_1(D)>h_2(D)$ для некоторых из них.

Оптимальность алгоритма поиска решающего вывода в графе вывода может быть доказана лишь для некоторых его частных случаев. Поэтому рассмотрение вопроса об оптимальности мы выносим в параграфы, посвященные описанию алгоритмов поиска решения в частных предписаниях.

Итак, в описанных стратегиях в качестве механизма генерации используется абстрактное преобразование, определенное на множестве предписаний в графе; в качестве механизма управления поиском используется оценочная функция, содержательно интерпретируемая количеством усилий, которые надо затратить для получения решения.

5.3. Поиск коммуникационных решений в пространстве состояний

5.3.1. Алгоритм и его свойства

Решение коммуникационной задачи в продукционной системе сводится к поиску *решающего пути* в графе пространства состояний. Таким образом, механизм генерации в этом предписании определяется преобразованием Г(*x*), порождающим все дочерние вершины *x*. Выбор альтернативных путей поиска, или вершин-кандидатов на генерацию, связывается с классом оценочных функций *f(x)*, где *x* — вершины графа. Мы будем выбирать для генерации ту из вершин-кандидатов, для которой функция *f* принимает *минимальное значение*. Мы рассматриваем вариант алгоритма для одной начальной вершины $x_0 \in X_o$.

Пусть *S* — множество уже выбранных вершин, $\overline{S}$ — множество вершин-кандидатов, $S \cap \overline{S} = \varnothing$, x_0 — начальная вершина, X_t — множество конечных вершин, $x_t \in X_t$, x — текущая вершина, $x \in X$, x_i — ее дочерние вершины. Тогда алгоритм поиска в графе пространства состояний представляется следующими шагами:

1) Поместить x_0 в $\overline{S}$ и вычислить для нее $f(x_0)$.

2) Если $\overline{S} = \varnothing$, неудача, иначе продолжать.

3) Выбрать такую $x \in \overline{S}$, что $f(x) = \min\limits_{y \in \overline{S}} f(y)$, и поместить ее в *S*, изъяв из $\overline{S}$.

4) Если $x \in X_t$, путь найден, окончание, иначе продолжать.

5) Найти *все* $x_i = \Gamma(x)$. Если $\Gamma(x) = \varnothing$, к шагу 2. Иначе вычислить все $f(x_i)$.

6) Для каждого x_i:

а) Если $x_i \notin S \bigcup \overline{S}$, то поместить x_i в $\overline{S}$.

б) Если $x_i \in \Gamma(x)$ n *S*, то сопоставить x_i наименьшую из старой и вновь полученной оценку $f(x_i)$.

в) Если $x_i \in \Gamma(x) \cap \overline{S}$, то сопоставить x_i наименьшую из старой и вновь полученной оценку $f(x_i)$, поместить x_i в $\overline{S}$ (изъяв ее из *S*).

г) В остальных случаях не изменять *S* и $\overline{S}$.

7) Перейти к шагу 2.

Пункты 6б и 6в отражают действия алгоритма в случае, когда оператор Г порождает уже рассмотренные ранее вершины, которые к этому моменту находятся либо в $\overline{S}$, либо в S. Естественно, что мы хотим приписать этим вершинам наименьшие из возможных оценочные функции. Этой проблемы не возникало бы при поиске в дереве, поскольку в этом случае имеется точно один путь из начальной вершины в данную.

Рассмотрим свойства этого алгоритма для класса ДПС, а затем укажем на некоторые возможные обобщения.

Поскольку представление в виде графа в пространстве состояний является частным случаем графа вывода, получаемым при исключении конъюнктивных вершин и ограничении числа дочерних вершин конечным числом, мы заключаем, что представленный выше алгоритм, управляемый ДПС, обладает полнотой.

Более того, в случае ограниченной ДПС алгоритм является допустимым, т. е. всегда находит наилучший в смысле минимума f решающий путь, если он существует.

Из определения оптимальности (п.5.2) ясно, что нам необходимо показать, что допустимая стратегия с эвристической функцией h_2 будет порождать любую вершину, которую будет порождать допустимая стратегия с функцией h_1.

Теорема 5.3. *Ограниченная ДПС оптимальна.*

Доказательство. Пусть нам даны ограниченная ДПС $\sum_1$ с эвристической функцией h_1 и другая допустимая стратегия $\sum_2$ с эвристической функцией h_2. Предположим, что существует такая вершина x, которую генерирует $\sum_1$, но не генерирует $\sum_2$. Тогда для $\sum_1$ оценка в вершине x

$$f(x)=g(x)+h_1(x). \tag{5.8}$$

В силу свойства ограниченности стратегии

$$f(x)\le f(x_t), \tag{5.9}$$

где $x_t \in X_t$, откуда

$$h_1(x)\le f(x_t)-g(x) \tag{5.10}$$

Стратегия $\sum_2$ в силу допустимости может не генерировать вершину x только по одной причине: существует некий менее сложный путь к x_t, чем путь через вершину x. Поэтому для $\sum_2$ действительная стоимость пути к x_t через x

$$f^*(x)\ge f(x_t). \tag{5.11}$$

Подставляя (5.10) в (5.11), получаем

$$h_1(x)\le g^*(x)-g(x)+h_2(x) \tag{5.12}$$

Однако из анализа шага 6 алгоритма ясно, что оценка $g(x)$ в результате повторного возврата к вершине x может, в крайнем случае,

уменьшиться, т. е. для всех $x \in X$ и на любом шаге работы алгоритма $g^*(x) \leq g(x)$. Отсюда

$$h_1(x) - h_2(x) \leq 0, \tag{5.13}$$

что противоречит определению оптимальности. Отсюда заключаем, что $\sum_2$ генерирует любую вершину, которую генерирует $\sum_1$ что доказывает теорему.

Замечание. Условие (5.3) монотонности ДПС позволяет вывести свойство меры сложности $g(x)$. Если $x \in S$, то это означает по построению алгоритма, что в этот момент $f(x) \leq f(y)$ для всех $y \in \overline{S}$. Поэтому и в силу монотонности, если мы вновь придем к вершине x из любой другой вершины, то новая оценка $f'(x) \geq f(x)$. Это означает, что при первой же генерации вершины x мы находим ее истинную меру сложности или, другими словами, при первом же включении x в S она уже находится на оптимальном пути из начальной вершины в x. Отсюда

$$g(x) = g^*(x) \tag{5.14}$$

и, следовательно, для алгоритма поиска, управляемого монотонной оценочной функцией, т. е. для допустимого алгоритма, шаги 6б и 6в могут быть без ущерба изъяты.

Необходимо указать на ряд важных особенностей алгоритма поиска, управляемого ограниченной ДПС. Допустимость стратегии гарантирует нахождение оптимального пути, однако при этом никак не оговаривается число генерируемых при этом вершин, т. е. фактически вычислительные ресурсы, необходимые для получения решения. Оптимальный путь будет найден с минимальными затратами, но лишь среди других допустимых стратегий.

Вполне вероятно, что путем отказа от допустимости мы могли бы, теряя гарантию найти оптимальный решающий путь, сократить число генерируемых вершин, получив, таким образом, некоторый компромисс между качеством получаемого решения и ресурсами, необходимыми для его получения, т. е. эффективностью алгоритма.

Далее, существенным дефектом ДПС является исследование всех альтернативных путей, имеющих одинаковое качество. Выбор вершины среди множества вершин с одинаковой оценочной функцией в рассмотренном выше алгоритме произволен, однако, в силу монотонности оценочной функции, алгоритм все время будет возвращаться к исследованию вершин в множестве $\overline{S}$, обладающих равным качеством с уже занесенной в S вершиной. В случае значительного количества равнокачественных текущих решений управление поиском с помощью ДПС окажется чрезвычайно слабым и алгоритм по своим вычислительным свойствам будет приближаться к полному

перебору вершин из $\overline{S}$, т. е. будет скорее поиском в ширину, чем в глубину. Между тем в практически важных случаях пространство поиска весьма часто имеет вблизи решения мезаструктуру, т. е. форму плато с мало отличающимися в точках пространства значениями оценочной функции. Если же это плато находится вблизи локального оптимума (пространство поиска не унимодально), то это дополнительно ухудшает качество работы алгоритма.

Для улучшения положения можно наложить еще одно ограничение на ДПС. Если два кандидата на генерацию обладают равной оценочной функцией, но неодинаковой мерой сложности, т. е. для $x, y \in \overline{S}$

$$\left.\begin{aligned} f(x) = f(y), \\ g(x) > g(y) \end{aligned}\right\} \qquad (5.15)$$

то выбирается для включения в S та вершина x, которая обладает большей мерой сложности, или большей стоимостью уже пройденного пути. Основанием для введения условий (5.15) является предположение, что путь от этой вершины к цели окажется короче $(h(x) < h(y))$. ДПС, удовлетворяющая условиям (5.15), называется *восходящей ДПС*. Геометрическая интерпретация этой стратегии очевидна из рис. 5.2. Дополнительно к направлению генерации, определяющему ДПС, добавляется направление генерации вдоль диагонали снизу вверх, предшествующее переходу с диагонали на диагональ, так что последовательность рассмотрения точек в координатах (g, h) будет D_1, D_3, D_2.

На качество работы алгоритма существенное влияние оказывает метод получения эвристической функции h.

Теорема 5.4. *Для любой ограниченной ДПС и для всех $x \in X$, лежащих на некотором пути к целевой вершине,*

а) $h(x) \leq h^*(x)$, *где $h^*(x)$ — истинная мера сложности оптимального пути от вершины x до целевой вершины;*

б) *стратегия с функцией $h^*(x)$ генерирует вершины только на оптимальном пути.*

Таким образом, можно предположить, что с приближением h к h^* алгоритм, управляемый ограниченной ДПС, будет генерировать все меньшее число вершин. Однако эвристическая функция определяется на основании специфической информации о решаемой задаче, т. е. на основании знания глобальных свойств пространства поиска. Слабость знаний решателя задач об этих свойствах является отличительной чертой всех систем предписаний, поэтому мы не можем рассчитывать на сколько-нибудь точное знание эвристической функции (почему она и называется эвристической). Попытка более точно определить

эвристическую функцию, в лучшем случае, приведет к более сложным вычислениям на каждой вершине, что отрицательно скажется на качестве работы алгоритма.

Заканчивая анализ ДПС как механизма управления поиском в пространстве состояний, мы хотим еще раз подчеркнуть, что

1) Оптимальная стратегия гарантирует эффективность алгоритма только в классе допустимых стратегий.
2) Наличие равнокачественных решений ухудшает направленность поиска.
3) Выбор эвристической функции может серьезно повлиять на качество работы алгоритма.

5.3.2. Методы повышения эффективности поиска

Рассмотрим некоторые методы отказа от допустимости, которые могут привести к повышению эффективности алгоритма поиска. Опишем следующие возможности:

1) Более общее задание оценочной функции.
2) Динамическое изменение оценочной функции в ходе поиска решения.
3) Динамическое переупорядочение множества операторов Г.
4) Динамическое отсечение вершин.

Общее задание оценочной функции. Можно задать оценочную функцию в виде

$$f(x) = (1-\omega)g(x) + \omega h(x), \quad 0 \le \omega \le 1, \qquad (5.16)$$

где $f(x)$ —монотонная оценочная функция.

Отметим ряд частных случаев этой функции:

1) $\omega=0$ — алгоритм поиска в ширину, или *стратегия насыщения уровня*.
2) $\omega = \frac{1}{2}$ — только что рассмотренный класс ДПС.
3) $\omega=1$ — алгоритм, управляемый эвристической функцией.

Стратегия с оценочной функцией вида (5.16) обладает следующими свойствами:

1) *Стратегия полна при* $\omega < 1$.
2) *Стратегия с истинной эвристической функцией* h^* *оптимальна при* $^1/_2 \le \omega \le 1$.

Чтобы проиллюстрировать повышение эффективности поиска для $^1/_2 \le \omega \le 1$ по сравнению с ДПС, приведем таблицу 5.1 результатов, экспериментально полученных для одной из задач (действительная минимальная длина пути равна 32).

Таблица 5.1

ω	Количество генерированных вершин	Длина пути
0,5	1000	Отказ от продолжения поиска
0,75	317	38
0,89	431	62
0,94	279	62
1,00	514	80

Динамическое изменение оценочной функции. Рассмотрим два способа динамического изменения оценочной функции: *аналитический* и *эвристический*.

В первом случае задаем оценочную функцию в виде

$$f(x)=g(x)+\omega(x)h(x). \qquad (5.17)$$

Весовой коэффициент является функцией вершины. Желательно, чтобы он был убывающей функцией расстояния вершины от начальной вершины. В этом случае в близких к начальной вершине уровнях осуществлялся бы поиск в глубину, избегающий рассмотрения равнокачественных вершин вдали от целевой вершины. По мере приближения к цели поведение алгоритма могло бы изменяться в сторону стратегий с насыщением уровня, т. е. к более подробному исследованию окрестности текущей вершины $x \in \overline{S}$. Положим

$$\omega(x) = 1 + e - \frac{e \cdot l(\mu(x_0, x))}{n}, \qquad (5.18)$$

где $0<e<1$, $l(\mu(x_0,x))$ — длина пути от начальной вершины к вершине x.

Оценочная функция (5.17) с весовым коэффициентом (5.18) будет обладать описанным выше поведением, если $g(x)$ меняется без больших скачков.

Стратегия, управляемая функцией (5.17), обладает одним важным свойством.

Теорема 5.5. *Стратегия с оценочной функцией* (5.17), *взвешивающим коэффициентом* (5.18) *и эвристической функцией* $h(x) \le h^*(x)$ *для всех*

$x \in X$ находит решающий путь, стоимость которого не превышает стоимость оптимального пути более чем в $(1+e)$ *раз.*

Доказательство. Пусть m — стоимость пути, найденного стратегией при данном e, а L — стоимость оптимального пути. Пусть $x \in \overline{S}$ — вершина на оптимальном пути. Тогда из (5 17) и (5.18)

$$f(x) < g(x) + (1+e)h(x). \tag{5.19}$$

Так как $h(x) \leq h^*(x)$, то

$$f(x) < g(x) + h^*(x) + eh^*(x). \tag{5.20}$$

Так как x лежит на оптимальном пути, то

$$f(x) < L + eh^*(x), \tag{5.21}$$

откуда

$$f(x) < L(1+e). \tag{5.22}$$

Но по оценке нашей стратегии $m \leq f(x)$, так что

$$m < L(1+e), \tag{5. 23}$$

что доказывает теорему.

В таблице 5.2 приведены экспериментальные результаты поиска в графе, сравнивающие работу стратегии с $\omega(x)h(x)$ и ДПС (известная оптимальная длина решающего пути равна 246).

Таблица 5.2

Стратегия	Длина решающего пути	Количество генерированных вершин
Стратегия со взвешиванием, $e=0,6$	260	353
Стратегия со взвешиванием, $e=0,4$	253	474
Диагональная поисковая стратегия	Отказ от продолжения поиска	500

Идея *эвристического* метода динамического изменения оценочной функции заключается в том, чтобы обновлять ее в процессе решения задачи на основе анализа уже построенных частичных поисковых деревьев и с помощью соответствующей техники усреднения и оптимизации. Мы называем этот метод эвристическим, потому что нет никакой гарантии, что поисковое пространство достаточно однородно, чтобы частичные деревья являлись представительной выборкой.

Кроме того, успех этого метода в значительной степени зависит от того, насколько удачно выбрана оценочная функция (с точки зрения ее близости к действительной стоимости пути).

Динамическое отсечение вершин. Несмотря на те или иные меры, принятые для улучшения качества работы алгоритма, возникает вопрос, что делать, когда вычислительные ресурсы (память или время) исчерпаны, а решение еще не найдено. В случае переполнения памяти выход заключается в том, чтобы отметить частичный путь из вершины $x \in \overline{S}$ с наименьшим значением *f(x)* к начальной вершине и затем стереть ту или иную часть построенного к этому моменту дерева. Например, можно запомнить часть этого пути, начиная с начальной вершины, и стереть все дерево поиска, кроме той его части, которая строится из последней вершины на запомненном пути. Этот вариант отсечения вершин является, в известной мере, аварийным. Более эффективной представляется процедура оценки вершин в зависимости от оценок дочерних вершин путем их частичной генерации. Идея заключается в том, чтобы подвергнуть относительно большое число вершин в $\overline{S}$ частичной генерации, используя лишь небольшое подмножество множества применимых к ним операторов. При этом часть вершин на основании такого просмотра вперед может быть сразу оценена как бесперспективная и отсечена. Эта оценка может быть приведена, например, на основе смешанной оценочной функции вида

$$f'(x) = \frac{f(x) + w\sum f(\Gamma'_i(x))}{1 + w(j-1)}, \qquad (5.24)$$

где Γ'_i, i=l, 2, . . ., j—1—выбранные для просмотра операторы, w— вес. Здесь мы сталкиваемся с необходимостью оценить относительные затраты на поиск без отсечения и дополнительные вычисления оценочных функций во всех вершинах, сопоставив их в каждом конкретном случае для принятия решения об использовании этого метода. Кроме того, при динамическом отсечении нет никакой гарантии, что мы не отсекаем именно те вершины, которые окажутся на оптимальном пути (отказ от допустимости).

Динамическое переупорядочение операторов. Идея динамического переупорядочения операторов заключается в том, чтобы по мере накопления опыта в процессе решения задачи или класса задач разработчик коммуникационных предписаний мог упорядочивать множество операторов (директив) по их полезности, испытывать затем при решении этой задачи или сходных с ней задач операторы (директивы) в установленном порядке, сокращать постепенно это множество до наиболее полезных операторов (директив) и в идеале

сопоставлять определенные группы операторов (директив) определенным категориям состояний.

Желательно, чтобы разработчик коммуникационных предписаний по мере накопления опыта синтезировал классификационную таблицу множества операторов (директив) и классов состояний, к которым эти операторы (директивы) целесообразно применять.

Применительно к коммуникационному предписанию в пространстве состояний задача может быть представлена следующим образом. Вместо оператора (директивы) Г мы вводим упорядоченное множество операторов (директив) $\Gamma' = \{ \Gamma'_i / i=1, 2, \ldots, m\}$, так что Γ'_i — i-й элемент этого множества, $\Gamma'_i = X \rightarrow X$. Г' следующим образом связан с Г: пусть $X_j = \Gamma(x_j)$ и

$$X'_j = \bigcup_{i=1}^{m} \Gamma'_i(x_j);$$

тогда, если $x_k \in X$ то $x_k \in X'_j$. Другими словами для всех j $X_j \subseteq X'_j$, т. е, мы допускаем синтез некоторых операторов (директив) и добавление их к множеству Г.

В наиболее очевидном варианте динамическое переупорядочение операторов (директив) может быть реализовано следующим образом. Первоначально порядок операторов (директив) в Г' произволен (например, задается случайным механизмом генерации индексов i, i=1, 2, . . ., m). Задается функция $\varphi: X \rightarrow \Gamma'$, отображающая $x_j \in X$, такие, что $f(x_j) = \min_{x_k \in \bar{S}} f(x_k)$ в порядковый номер оператора (директивы), т. е. для каждой вершины, выбранной для перенесения в S, операторы (директивы) испытываются в порядке, установленном к данному моменту. Далее, в процессе анализа частичных поисковых деревьев те операторы (директивы), которые были применены и соответствуют дугам на оптимальном пути в этих деревьях, «поощряются» уменьшением их индекса в множестве Г', а примененные операторы (директивы), соответствующие дугам вне оптимального пути, «наказываются» увеличением индекса. Эта схема может быть усилена, так что операторы (директивы) из некоторого подмножества $\Gamma'_r \subset \Gamma'$, r=s, s+1, …., m, в случае получения ими очередного «наказания» отбрасываются. В то же время к вершинам $x_j \in \bar{S}$, выбранным для генерации, применяется лишь ограниченное множество операторов (директив) $\Gamma'_p \subset \Gamma'$, p=1, 2, ..., s—1. В этом случае обновление Γ'_p происходит за счет перехода «наказанных» операторов (директив) из «высшей лиги» в «низшую» и замены их операторами (директивами) из низшей лиги.

Кардинальное решение проблемы переупорядочения операторов (директив) следует искать на пути разбиения поисковых пространств на обладающие различными свойствами области в духе рассмотренных нами в задаче о миссионерах и людоедах или образов Сэндуолла. Естественно предположить, что каждой такой области в результате накопления опыта или непосредственно из ее анализа можно было бы соотнести свое небольшое специализированное множество операторов (директив). Такие множества могли бы составить основу для эффективного поиска решений, однако решение этой задачи является производным от более общей проблемы глобального исследования пространства поиска и преобразования коммуникационных предписаний, которая, как отмечалось выше, еще не нашла своего решения

Доказательство допустимости и оптимальности ДПС имело своей целью построение такой стратегии, которая находила бы наилучшие в смысле минимума оценочной функции решения. Из обсуждения, касающегося работы ДПС в случае большого числа равнокачественных альтернатив, ясно, что ДПС даже в большей степени подходит для поиска всех наилучших решений, чем для поиска только одного такого решения. Это вытекает из того, что если алгоритм ДПС нашел наилучшее решение, то к этому моменту он либо раскрыл остальные лучшие решения, либо находится весьма близко к их раскрытию. Однако часто, в особенности для больших поисковых коммуникационных пространств, ставится задача поиска любого решения В этом случае применение ДПС является весьма неэффективным опять-таки в силу параллельного исследования равнокачественных альтернатив.

Таким образом, при необходимости найти любое решение целесообразен сознательный отказ от допустимости алгоритма и использование других поисковых стратегий. В частности, хорошо для этой цели могут подойти стратегии с большим ω в формуле (5. 17).

5.4. Двунаправленный поиск решения в пространстве состояний

В случае, когда целевое состояние задано явным образом, возможно осуществлять поиск решения в графе как от начальной вершины к конечной, так и от конечной вершины к начальной. Эти два направления поиска могут быть объединены в единый процесс двунаправленного поиска. Основанием для конструирования такого алгоритма является факт экспоненциального роста поисковых деревьев

и предположение, что при этом два более коротких диаметра поиска будут генерировать меньше вершин, чем один длинный (здесь под *диаметром поиска* мы понимаем глубину проникновения механизма генерации в поисковом пространстве, т. е длину максимально удаленного от начала поиска пути).

Вновь введем некоторую оценочную функцию для управления поиском. Однако, поскольку поиск производится в двух направлениях, оценочная функция должна быть определена для обоих направлений — прямого *(F)* и обратного *(B)*. Пусть она равна f_F и f_B соответственно. В следующем определении алгоритма сохраняются все обозначения из предыдущего параграфа, $a_{\min}$ равна текущему минимуму стоимости уже найденного пути, $\overline{T}$ и T — множества вершин-кандидатов и генерированных вершин в направлении B, $f_F(x)=g_F(x)+h_F(x)$, $f_B(x)=g_B(x)+h_B(x)$, где $h_F(x)$ и $h_B(x)$ — оценки минимальных стоимостей путей $\mu(x, \ldots, x_t)$ и $\mu(x_0, \ldots, x)$ соответственно.

1) Поместить x_0 в $\overline{S}$ и вычислить $f_F(x_0)$, поместить x_t в $\overline{T}$ и вычислить $f_B(x_t)$. Установить $a_{\min}=\infty$.

2) Выбрать направление F (продолжать) или B (к шагу 4).

3) Выбрать такую $x \in \overline{S}$, что $f_F(x)=\min\limits_{y\in\overline{S}} f_F(y)$. Поместить x в $\overline{S}$ (изъяв ее из $\overline{S}$) и проверить все $x_i \in \Gamma(x)$ на следующие возможности:

а) Если $x_i \notin S \bigcup \overline{S}$, то поместить ее в $\overline{S}$.

б) Если $x_i \in \Gamma(x) \cap \overline{S}$, то сопоставить x_i наименьшую из старой и вновь полученной оценку $f_F(x_i)$.

в) Если $x_i \in \Gamma(x) \cap S$, то сопоставить x_i наименьшую из старой и вновь полученной оценку $f_F(x_i)$, поместить x_i в $\overline{S}$ (изъяв ее из S).

г) В остальных случаях не делать никаких изменений в S и $\overline{S}$. К шагу 5.

4) Выбрать такую $x \in \overline{T}$, что $f_B(x)=\min\limits_{y\in\overline{T}} f_B(y)$. Поместить x в T (изъяв ее из $\overline{T}$) и проделать те же проверки, что и в шаге 3, относительно $x_i \in \Gamma^{-1}(x)$, используя T и $\overline{T}$. Продолжать.

5) Если $x \in S \cap T$, то $a_{\min}:=\min(a_{\min}, g_F(x)+g_B(x))$. Если $a_{\min} \leqslant \max\left[\min\limits_{x\in\overline{S}} f_F(x), \min\limits_{x\in\overline{T}} f_B(x)\right]$, то выход алгоритма, $a_{\min}$ дает длину наилучшего пути. Иначе к шагу 2.

Трудности реализации двунаправленного поиска связаны с двумя вопросами: выбором метода определения направления генерации и оптимизацией «точки встречи» прямого и обратного поиска.

Одним из простейших правил выбора между направлениями *F* и *B* явилось бы правило эквидистантности от начальной и конечной вершины. Однако это привело бы нас фактически к малоэффективному поиску в ширину как в прямом, так и в обратном направлениях. Поль включил в критерии выбора направления генерации размеры множеств $\overline{S}$ и $\overline{T}$, отражающие плотность дочерних вершин и сужающие поиск в случае предпочтения направления с меньшей мощностью множества. Правило выбора *«если* $|\overline{S}| \le |\overline{T}|$, *то продолжать, иначе к шагу* 4», названное Полем *принципом сравнения мощности,* может быть подставлено в шаг 2 алгоритма двунаправленного поиска. Однако для случая бесконечного дизъюнктивного ветвления этот критерий не годится. В духе определения квазиконечного упорядочения качества было бы переформулировать правило выбора следующим образом: *«Если* $\overline{S}$ *содержит меньше вершин с наилучшим качеством, продолжать, иначе к шагу* 4».

Более сложным является вопрос об управлении точкой встречи прямого и обратного поиска. Мы должны разрешить противоречие, органически присущее эвристическому поиску: эвристическая функция вводится для того, чтобы сделать поиск более направленным, и с этой точки зрения целесообразно, чтобы стратегия поиска обеспечивала исследование относительно узких областей в окрестности оптимального пути. Однако при этом возникает риск, что пути, полученные в прямом и обратном направлениях, разойдутся. Поскольку ДПС допустима, то оптимальный путь будет получен, но ожидаемые преимущества двунаправленного поиска не будут реализованы. Более того, если встреча произойдет в районе начальной (конечной) вершины, то эффективность двунаправленного поиска будет ниже, чем у поиска в одном направлении. Таким образом, встает задача привести поисковые деревья к такой точке встречи, чтобы длина путей в направлениях *F* и *B* $\mu_F=\mu_B$. Один из способов решения этой задачи — определение промежуточной вершины x_i такой, что

$$h_F(x_i) \cong h_B(x_i) \cong {}^1/_2 h_F(x_0). \qquad (5.25)$$

Другими словами, мы пытаемся направить поиск к промежуточной вершине как в направлении *F,* так и в направлении *B,* что не означает, что встреча произойдет именно в этой точке. Это решение в духе редукционного подхода, напоминающее идею ключевых состояний и операторов (директив).

Другой возможностью является непрерывное обновление эвристической функции с целью направить поиск в одном из направлений к фронту поиска в противоположном направлении. Это может быть сделано изменением целевой вершины при каждой перемене направления поиска: в качестве целевой вершины выбирается последняя вершина, исследованная в противоположном направлении (т. е. последняя вершина, помещенная в список S или T).

Мы рассмотрим одну из возможностей управления эвристической функцией h_X с помощью оценочной функции f_Y, где X и Y— противоположные направления. Пусть произошел переход от генерации вершин в направлении X к генерации в направлении Y (X и Y могут быть как F, так и B). Мы оцениваем ситуацию в поисковом пространстве с позиций оценочной функции $f_X(x)$, где x—последняя генерированная вершина в направлении X, т. е. $x \in S$, если $X=F$, и $x \in T$, если $X=B$. Предположим, что в направлении Y имеется множество вершин-кандидатов $y \in \overline{S}$, если $Y=F$, и $y \in \overline{T}$, если $Y=B$. Поиску в направлении X соответствует оценочная функция

$$f_X(x)=g_X(x)+h_X(x), \qquad (5.26)$$

в направлении Y —

$$f_Y(y)=g_Y(y)+h_Y(y) \qquad (5.27)$$

(аргументы в дальнейшем опускаем).

Рассмотрим два варианта:

1. С позиций f_X суммарный путь не дешевле g^*, где g^* — действительная стоимость, т. е. $f_X \le g_X+g_Y$. Учитывая (5.26), получаем для этого варианта $h_X \le g_Y$. В этом случае мы заключаем, что поиск со стороны Y зашел слишком далеко (заметим, что g_X — истинная стоимость оптимального пути в направлении X). Следует переопределить оценочную функцию для $y \in Y$ таким образом, чтобы она выбирала для генерации те вершины, которые находятся ближе к фронту глубины g_X. Иначе говоря, следует, чтобы f_Y более реалистично оценивала суммарную длину g_X+g_Y. Для этого достаточно, чтобы $f_Y \ge f_X$, откуда, так как $h_X \le g_Y$, $h_Y \ge g_X$ для всех $y \in Y$. Если же для некоторых $y \in Y$ $h_Y < g_X$, то устанавливается $h_Y=g_X$.

2. С позиций f_X суммарный путь дешевле, чем g^*, т. е. $f_X > g_X + g_Y$ или $h_X > g_Y$. В этом случае, в силу ограниченности ДПС и того, что $x \in X$, $g_X+g_Y<f_X \le g^*$, так что f_X представляет собой нижнюю оценку наилучшего решения. Поэтому можно положить $f_Y \ge f_X$, откуда получаем $h_Y \ge f_X - g_Y$. В случае невыполнения последнего неравенства устанавливается $h_Y = f_X - g_Y$. Мы объединим эти два варианта в один, записав

$$h_Y \ge g_X + (h_X \dot{-} g_Y), \qquad (5.28)$$

где

$$a \dot{-} b = \begin{cases} 0, & a \leqslant b, \\ a-b, & a > b. \end{cases}$$

Формула (5.28) дает нам возможность обновлять эвристическую функцию для поиска в направлении *Y* после каждого перехода к этому поиску от поиска в направлении *X*.

5.5. Поиск решения в пропозициональных графах

5.5.1. Алгоритм поиска минимального решающего графа

Мы ставим проблему решения задачи путем сведения ее к подзадачам как *поиск минимального решающего графа в неявно заданном пропозициональном графе.*

Определим понятие минимального решающего графа. С этой целью связываем с каждой дугой пропозиционального графа меру сложности, или стоимость дуги. Определим *стоимость графа* как сумму стоимостей всех его дуг. Тогда решающий граф, имеющий минимальную стоимость, называется *минимальным решающим графом.* Соответственно *минимальным путевым графом из* $s_1, s_2, .. ., s_k$ в $t_1, t_2, . . ., t_m$ называется путевой граф из $\{s_i\}$ в $\{t_l\}$, имеющий минимальную стоимость.

Введем ряд дополнительных определений. Мы по-прежнему называем применение оператора (директивы) сведения задачи к подзадачам, порождающее из некоторой вершины *s* ее дочерние вершины $s_1, s_2, . . ., s_k$, *генерацией, или раскрытием вершины* s. Очевидно, что конечная вершина никогда не раскрывается. Вершина, не являющаяся конечной и которая еще не генерирована, называется *кандидатом на генерацию.* Пусть $S = S_1S_2. . S_k$— конъюнкция. Конъюнкция называется *раскрытой* тогда и только тогда, когда раскрыты все не являющиеся конечными вершины $s_1, s_2, . . ., s_k$. Пусть $s=\{s_i/i=1, 2, . . ., q\}$ — множество начальных вершин и для s задана некоторая импликанта $P= N_1N_2. . .N_r$. Мы связываем с каждой такой импликантой оценочную функцию $f(P)=g(P)+h(P)$, где $g(P)$ — оценка стоимости минимального путевого графа из вершин $s_1, s_2,. . ., s_q$ в вершины $n_1, n_2, . . ., n_r$, $h(P)$— оценка стоимости минимального решающего графа, начинающегося в вершинах $n_1, n_2, . . ., n_r$. Соответствующие истинные значения

обозначим через $g^*(P)$ и $h^*(P)$. Заметим, что в минимальном решающем графе

$$h^*(P) \leqslant \sum_{i=1}^{r} h^*(N_i) \tag{5.29}$$

(для пропозиционального дерева верно равенство).

Основываясь на оценочной функции $f(P)$, мы можем построить алгоритм поиска решения в пропозициональном графе. Стратегия поиска напоминает ДПС, но существенно отличается от нее способом образования кандидатов на генерацию. Однако благодаря этому мы сможем доказать допустимость и оптимальность этого алгоритма. План дальнейшего изложения состоит в том, чтобы формально определить алгоритм, доказать его допустимость и оптимальность, а затем рассмотреть связь выбранной стратегии с ДПС и некоторые другие стратегии.

В приведенном ниже алгоритме $W=\{W_j\}$ — множество конъюнкций, соответствующих вершинам-кандидатам на генерацию, $V(P)$ — множество импликант S, получаемых из предписаний P замещением каждой N_i, не являющейся конечной, одной из ее непосредственных импликант $(P = N_1N_2 \ldots N_r)$, S— предписание, связанное с начальными вершинами, R — множество конъюнкций, соответствующих уже генерированным вершинам.

1) Положить $W=\{S\}$, $R = \varnothing$.

2) Вычислить $f(Q)$ для каждого $Q \in W$. Выбрать такое $P \in W$, что $f(P) = \min\limits_{Q \in W} f(Q)$. В случае равенства выбор произволен с предпочтением $Q \in T$, T — множество предписаний, соответствующих конечным вершинам.

3) Пусть $P = N_1N_2 \ldots N_r$, N_i—предписания, связанные с вершинами n_i, $i = 1,2, \ldots, r$. Если все n_i — конечные вершины, выход с решающим графом. Иначе раскрыть все нераскрытые вершины n_i, не являющиеся конечными.

4) Положить $R=R \cup \{P\}$, $W=(W \cup V) \backslash R$. Если $W= \varnothing$, выход, решающего графа нет. Иначе к шагу 2.

Пример. Рассмотрим граф на рис. 5.3, *а*.

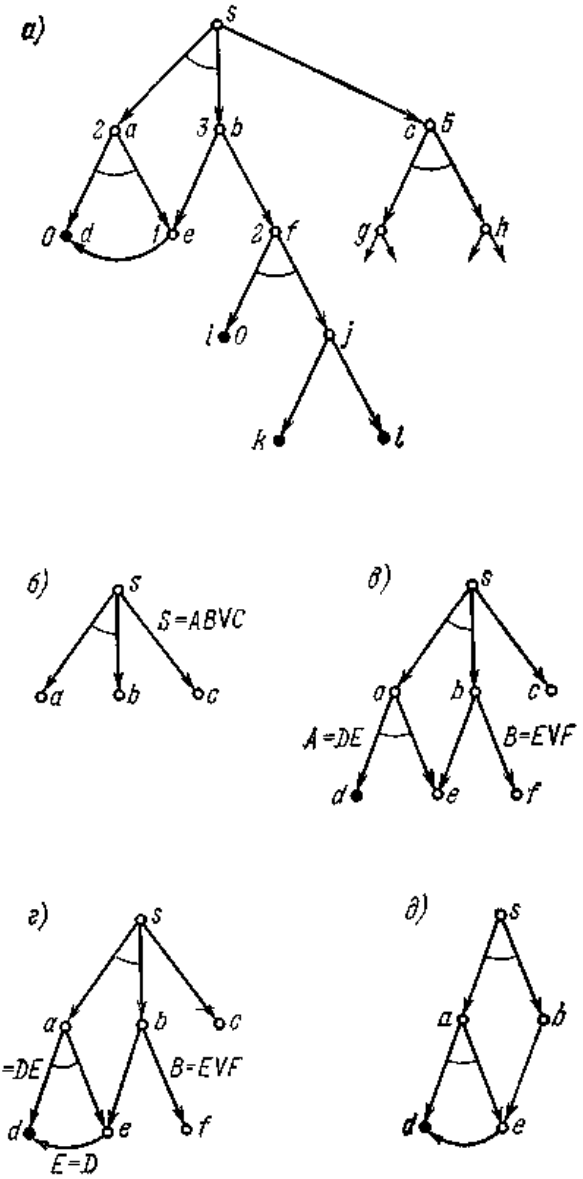


Рис. 5.3. Пример работы алгоритма: *а)* исходный граф, *б), в), г)* шаги алгоритма, *д)* решающий граф.

Положим $h(P) = r$ — 1+ min $\{h(N_1), h(N_2), ..., h(N_r)\}$. Числа рядом с вершинами n — оценки $h(N)$. Стоимости дуг положены равными единице. Последовательные шаги алгоритма представлены на рис. 5.3, *б, в, г* соответственно.

1) Раскрытие s. $W=\{AB, C\}$, $R = \{S\}$, $V=\{AB, C\}$.

$h(AB)=(2—1)+$ min $(h(A), h(B)) = 1+\min(2, 3)=3$.

$f(AB)=2+3=5$, $f(C) = 1+5=6$.

Выбираем AB для дальнейшего раскрытия.

2) Раскрытие a и b. Поскольку $AB=DE(E+F)=DE+DEF$, то

$V=\{DE, DEF\}$, $R = \{S, AB\}$, $W= \{DE, DEF, C\}$.

$h(DE)=(2—1)+\min(h(D), h(E)) = 1+\min(0, 1) = 1$.

$h(DEF)=(3-1)+$ min $(h(D), h(E), h(F))=2+\min(0,1, 2) =2$.

$f(DE)=5+1=G$, $f(DEF)=5+2=7$, $f(DE)=f(C)$.

Предположим, что для раскрытия выбрано DE.

3) Раскрытие e. Поскольку $DE=D$, то $V=\{D\}$, $R = \{S, AB, DE\}$, $W=\{D, DEF, C\}$, $h(D)=0$, $f(D)=6+0=6$, $f(D)=f(C)$.

D — конечная вершина, поэтому выбираем D. Алгоритм заканчивает работу с решающим графом, представленным на рис. 5.3, *д*.

Заметим, что в приведенном алгоритме не указаны некоторые детали, например, расстановка указателей для извлечения решения и т. д.

5.5.2. Свойства алгоритма поиска минимального решающего графа

Допустимость.

Л е м м а 5.1. *Пусть* $h(Q) \le h^*(Q)$ *для всех* Q, *являющихся импликантами* S. *Если имеется минимальный решающий граф, то алгоритм выбирает такую импликанту* P, *что* $f(P) \le f^*(S)$.

Доказательство. Пусть G_m — минимальный решающий граф, G_Q — частично раскрытый граф, $G_Q \subseteq G_m$, порожденный к моменту выбора P. Пусть $q_1, q_2, \ldots, q_r$ — конечные и нераскрытые вершины G_Q. Тогда $Q=Q_1Q_2 \ldots Q_r$ — импликанта S, $Q \in W$. В силу минимальности G_m и $G_Q \subseteq G_m$ $f^*(S)=f^*(Q)$ и $g(Q)=g^*(Q)$. Так как $h(Q) \le h^*(Q)$, то

$$f(Q) \le g^*(Q)+h^*(Q)=f^*(Q)=f^*(S),$$

Но так как $f(P) = \min\limits_{Q \in W} f(Q)$, то

$$f(P) \le f(Q \le f^*(S). \qquad (5.30)$$

Назовем пропозициональный граф G δ-*графом,* если стоимость всех дуг G не менее, чем δ.

Теорема 5.6. *Пусть* $h(Q) \le h^*(Q)$ *для всех* Q, *являющихся импликантами* S. *Для всех* δ-*графов алгоритм поиска минимального решающего графа допустим.*

Доказательство. Предположим, что δ-граф G содержит минимальный решающий граф G_m. Докажем последовательно следующие три утверждения:

1) Алгоритм закончит работу.
2) Алгоритм закончит работу с решающим графом.
3) Алгоритм закончит работу с минимальным решающим графом.

1) Рассмотрим $Q \in W$, $Q=N_1N_2 \ldots N_n$ такую, что вершины $n_1, n_2, \ldots, n_r$ находятся на расстоянии d от ближайшей из начальных вершин. Тогда $f(Q) \ge d\delta$. Следовательно, если имеемся G_m с $f^*(S)$, для всех импликант Q с вершинами $n_1, n_2, \ldots, n_r$ удаленных более чем на $f^*(S)/\delta$ от ближайшей из начальных вершин, $f(Q) > f^*(S)$. Но по лемме 5.1 алгоритм выберет такую P, что $f(P) \le f^*(S)$. Следовательно, Q не будет выбрана алгоритмом, и поэтому он закончит работу.

2) Алгоритм не может закончить на шаге 4, так как имеется минимальный решающий граф, и $W \neq \varnothing$. Тогда остановка может произойти только на шаге 3, т. е. с решением.

3) Пусть T — импликанта, выбранная алгоритмом непосредственно перед окончанием его работы. По лемме 5.1 $f(T) \le f^*(S)$. Тогда $f^*(T) \le f(T) \le f^*(S)$. Поскольку $f^*(T)$ не превосходит минимальной стоимости, то $f^*(T)$ минимальна.

Оптимальность. Поскольку для доказательства допустимости алгоритма мы наложили ограничение $h(Q) \le h^*(Q)$, то очевидно, что в пределах этого ограничения, в зависимости от конкретных значений $h(Q)$, существует множество допустимых алгоритмов. Мы хотели бы выбрать из них такой алгоритм, который обладал бы наибольшей эффективностью в том смысле, что он раскрывал бы при нахождении минимального решения минимальное количество вершин. Таким образом, накладывая некоторые дополнительные ограничения на эвристическую функцию h, мы докажем оптимальность алгоритма в указанном выше смысле. Следует подчеркнуть, что вновь, как и в задачах поиска пути минимальной стоимости, оптимальность доказывается лишь в классе допустимых алгоритмов. Вводятся следующие дополнительные ограничения на эвристическую функцию:

1) Для всех P, являющихся импликантой S и содержащих высказывания, связанные только с конечными вершинами,

$$h(P) \equiv 0. \tag{5.31}$$

2) Для любых Q, являющихся импликантой S, и P, являющихся импликантой Q,

$$h(Q) - h(P) \le k(Q, P), \tag{5.32}$$

где $k(Q, P)$— стоимость минимального путевого графа из$\{q_i\}$ в$\{p_j\}$.

Лемма 5.2. *Если алгоритм, управляемый оценочной функцией, удовлетворяющей* (5.32), *выбирает импликанту P, то* $g(P)=g^*(P)$.

Доказательство. Предположим, что $g(P)>g^*(P)$. Пусть $P=P_1P_2 \ldots P_r$. Существует минимальный путевой граф G_0 из начальных вершин в $p_1, p_2, \ldots, p_r$, частично раскрытый в силу $g(P)>g^*(P)$. Пусть $q_1, q_2, \ldots q_m$— все конечные и нераскрытые вершины в G_0. Очевидно, что $Q \in W$, $Q=Q_1Q_2 \ldots Q_m$, а P является импликантой Q. Поскольку G_0—минимальный путевой граф, $g(Q)=g^*(Q)$. По предположению леммы и из определения функции $g^*(P)$

$$g(P)>g(Q)+k(Q, P). \qquad (5.33)$$

Добавляя $h(P)$ к общим частям неравенства (5.33), получаем $g(P)+h(P)>g(Q)+h(P)+k(Q, P)$ и в силу (5.32) $f(P)>f(Q)$. Но это означало бы, что алгоритм выберет импликанту Q, что противоречит условию леммы. Поэтому

$$g(P)=g^*(P). \qquad (5.34)$$

Теорема 5.7. *Пусть* $\sum_1$ *и* $\sum_2$ *— две допустимые стратегии, управляемые эвристическими функциями* h_1 *и* h_2 *соответственно. Если*

1) *для всех P, являющихся импликантами S и содержащих по крайней мере одно предписание, связанное с неконечной вершиной,* $h_1(P)>h_2(P)$;

2) *удовлетворяются ограничения* (5.31) *и* (5.32), *то для любого* δ*-графа, имеющего минимальный решающий граф, импликанта, выбираемая* $\sum_1$ *будет также выбираться* $\sum_2$.

Доказательство. Пусть $P_1, P_2, \ldots$ — последовательность импликант, выбранных $\sum_1(P_1=S)$. Предположим, что теорема неверна. Тогда существует P_k такая, что она выбирается $\sum_1$, но не выбирается $\sum_2$. Тогда для $\sum_2$ $f_2(P_k) \geq f^*(S)$ или $h_2(P_k)>f^*(S)-g_2(P_k)$. Так как P_k— первая импликанта, выбранная $\sum_1$ и не выбранная $\sum_2$, то $\sum_2$, так же как и $\sum_1$ успела породить последовательность импликант $P_1P_2, \ldots, P_k$. Но по лемме 5.2 для $\sum_1$ $g_1(P_k) = g^*(P_k)$. Тогда по построению алгоритма для $\sum_2$ $g_2(P_k) = g^*(P_k)=g_1(P_k)$. Следовательно, $h_2(P_k) \geq f^*(S)—g^*(P_k)$. Для $\sum_1$ получаем (по лемме 5.1) $g_1(P_k)+h_1(P_k) \leq f^*(S)$, откуда, учитывая лемму 5.2, $h_1(P_k) \leq f^*(S)—g^*(P_k)$ и, наконец, $h_1(P_k) \leq h_2(P_k)$, что противоречит условию 1 теоремы.

Следствие. *При условиях теоремы* 5.7 *каждая вершина, раскрываемая* $\sum_1$ *раскрывается* $\sum_2$.

Мы уже упоминали выше, что рассмотренная стратегия поиска решающего графа напоминает ограниченную ДПС, определенную (5.6). Однако имеется одно существенное различие: алгоритм поиска решающего вывода в графе вывода на каждом шаге генерирует *только один* кандидат на вывод, т. е. по определению кандидата на вывод в направлении *B, точно один акт вывода и его посылки* присоединяются

к исходному РВ. В алгоритме раскрываются одновременно *все* еще не раскрытые вершины. В понятиях обобщенного предписания это означает, что кандидатом на вывод в направлении B является вывод D, если он содержит уже генерированный РВ $D_0 \subseteq D$, уже генерированные акты вывода, принадлежащие D_0, и для каждой посылки D_0— один акт вывода, который еще не генерирован и который имеет в качестве заключения эту посылку.

Смысл параллельного анализа всех нераскрытых вершин заключается в том, что если вывод (или, что то же самое, частично раскрытый путевой граф) зачислен в кандидаты на генерацию с определенной оценкой, то он будет иметь эту оценку все время, пока не будет раскрыт (или пока алгоритм не закончит работу). Это условие всегда удовлетворяется для поиска в пространстве состояний. Однако структура пространства поиска в пропозициональных графах существенно сложнее, и в общем случае вывод, который на данном шаге является кандидатом на генерацию, затем может перестать быть им и не будет генерирован на более поздних стадиях, поскольку данный вывод может стать частью вывода, который имеет худшее качество, чем другие кандидаты.

Оптимальность алгоритма покупается за счет параллельного исследования всех вершин, дочерних по отношению к конъюнктивной, и поэтому он весьма близок к классу алгоритмов поиска в ширину, т. е. алгоритмов, весьма слабо управляемых эвристической функцией. С этой точки зрения он является относительно неэффективным, а оптимальность его не играет большой роли, так как доказана в классе допустимых алгоритмов, также слабо управляемых функцией h.

Мы имеем две возможности сделать поиск решающего графа более направленным. Первая из них заключается в том, чтобы модифицировать алгоритм, допустив раскрытие только одной вершины. Легко сделать это, внеся изменения в шаге 3 и в определение множества V исходного алгоритма.

3) Пусть $P=N_1N_2 \ldots N_r$, N_i — предписания, связанные с вершинами n_i, $i=1,2, \ldots, r$. Если все n_i— конечные вершины, выход с решающим графом. Иначе раскрыть нераскрытую вершину n_k, $1 \le k \le r$, такую, что $f(N_k) = \min\limits_i f_k(N_i)$.

Определение множества *V*. $V(P)$ — множество импликант S, получаемых из предписаний P замещением каждой раскрытой N_i, не являющейся конечной, одной из ее непосредственных импликант $(P=N_1N_2 \ldots N_r)$.

Легко видеть, что с этими модификациями алгоритм становится частным случаем алгоритма поиска решающего вывода в графе

вывода. Как показано в п.5.2, этот алгоритм, управляемый оценочной функцией $f(P)=g(P)+h(P)$, допустим. Однако оптимальность этого алгоритма не может быть установлена по следующей причине. При доказательстве оптимальности необходимо показать, что если акт вывода генерируется стратегией $\sum_1$ с эвристической функцией h_1, то он также генерируется и $\sum_2$, управляемой такой эвристической функцией h_2, что $h_2<h_1$. Далее, для $\sum_2$ надо показать, что этот акт принадлежит кандидату на вывод лучшего качества, чем минимальное решение, и поэтому он генерируется перед окончанием алгоритма. Этот прием работает для тех случаев, когда выбранный в качестве кандидата вывод остается кандидатом неизменного качества, пока он не будет выбран для генерации (поиск пути минимальной стоимости, алгоритм). В нашем же случае качество вывода может стать хуже и тогда акт вывода, принадлежащий ранее кандидату на вывод, имевшему стоимость, меньшую стоимости минимального решения, будет теперь принадлежать только кандидатам с худшим, чем у решения, качеством. Вторая возможность повышения эффективности поиска—управление с помощью оценочной функции вида $h(P)$, т. е. без учета стоимости уже построенного частично раскрытого путевого графа.

5.5.3. Поиск решающего графа в аддитивном пропозициональном графе

Рассмотрим другое возможное обобщение алгоритма поиска минимального решающего дерева. Это обобщение, называемое *поиском решающего графа в аддитивном пропозициональном графе,* представляет интерес по нескольким причинам. Во-первых, на этом алгоритме мы продемонстрируем отличный от импликативного метод определения разрешимых вершин и решающих графов, хорошо реализуемый в списочных структурах, — так называемый **метод указателей**. Во-вторых, на примере работы этого алгоритма мы проиллюстрируем изложенный выше тезис об ухудшении качества вывода при раскрытии одной вершины, дочерней для конъюнктивной и имеющей минимальную стоимость. В-третьих, рассмотрение этого алгоритма даст нам повод к обсуждению проблемы уменьшения избыточности при представлении редукционных систем графами, а не деревьями.

Предположим, что нам дано пропозициональное дерево. Естественно, что одинаковые подзадачи, возникающие в ходе разбиения задач более высокого уровня, представляются в дереве различными вершинами. Аддитивный пропозициональный граф может быть получен из такого дерева, если

1) Одинаковые подзадачи в дереве идентифицируются как корни одних и тех же поддеревьев и представляются одной вершиной в графе без циклов.

2) Стоимость решающего подграфа приравнивается стоимости соответствующего решающего поддерева, т. е. мера сложности решения идентифицированных подзадач вычисляется один раз, но прибавляется к стоимости решающего подграфа столько раз, сколько она встречается.

На рис. 5.4, *а* представлен пример аддитивного пропозиционального графа, а на рис. 5.4, *б, в* даны его решающие графы.

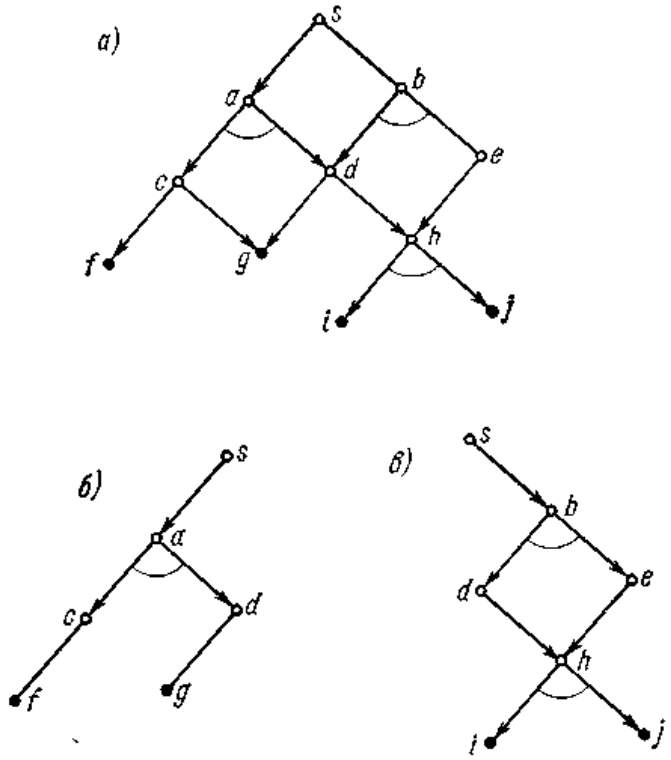


Рис. 5.4. Аддитивный пропозициональный граф *(а)* и его решающие графы *(б, в)*.

В таблице 5.3 показаны стоимости этих решающих графов, подсчитанные для общего вида пропозиционального графа и

аддитивного пропозиционального графа, в предположении единичных стоимостей дуг.

Таблица 5.3

Вид пропозиционального графа	Стоимость решающего графа	
	рис. 5.4, *б*	рис. 5.4, *в*
Общий вид (сумма стоимостей всех дуг)	5	7
Аддитивный (сумма стоимостей решающих поддеревьев)	5	9

Заметим, что неравенство (5.29) в последнем случае превратилось бы в равенство в силу указанного в пункте 2.

В общем, *стоимость решающего графа* определяется рекурсивно следующим образом:

1) С каждой конечной вершиной связывается стоимость $C = 0$.

2) Стоимость, связанная с любой другой вершиной x,

$$C(x) = \sum_{i=1}^{k} (C_i + c(x, x_i)) \qquad (5.35)$$

как для дизъюнктивных, так и для конъюнктивных вершин. Здесь x_i, i=l, 2, . . ., k,— дочерние вершины вершины x, $c(x, x_i)$— стоимость дуги, связывающей x с x_i.

3) Стоимость, связанная с начальной вершиной, есть стоимость решающего графа.

Будем считать также, что все вершины аддитивного пропозиционального графа либо конъюнктивные, либо дизъюнктивные (ранее мы допускали, чтобы одна и та же вершина могла быть по отношению к одним дочерним вершинам конъюнктивной, а по отношению к другим — дизъюнктивной). Остальные понятия являются стандартными для пропозициональных графов.

Решающий граф пропозиционального графа обладает следующими свойствами:

1) Если дизъюнктивная вершина включена в решающий граф, то *одна и только одна* ее дочерняя вершина также включена в решающий граф.

2) Если конъюнктивная вершина включена в решающий граф, то *все* ее дочерние вершины также включены в решающий граф.

Мы уже упоминали выше, что для указания частично построенного графа минимальной стоимости используется метод указателей. Это означает, что на каждом шаге цепочка указателей, начиная от начальной вершины, определяет текущего кандидата в решающий граф и следующую генерируемую вершину. Эта же цепочка указателей используется для извлечения ответа в конце работы алгоритма. Образование цепочки указателей управляется обновлением оценок вершин, предшествующих раскрываемым во время работы алгоритма. Заметим, что, в отличие от алгоритма, где раскрывались все вершины, данный алгоритм существенно зависит от работы механизма образования указателей.

За оценку стоимости вершины мы будем принимать стоимость минимального решающего графа, начинающегося в этой вершине. Эта оценка является эвристической функцией $h(x)$.

Алгоритм поиска решающего графа в аддитивном пропозициональном графе представляется следующими шагами:

1) Выбрать начальную вершину x_0.

2) Если x_0 разрешима, то выход с решающим графом, получаемым, начиная с x_0, прослеживанием направлений указателей вплоть до конечных вершин. Иначе продолжать.

3) Проследить указатели от x_0 к вершине x и раскрыть ее. Пусть вершины, дочерние по отношению к x, — x_i, i =l,2, ...,k.

4) Обновить оценки и направления указателей вершины x и всех вершин, предшествующих x, по следующим правилам:

а) если x_i уже раскрывалась ранее, она уже имеет оценку $h(x_i)$;

б) если x_i раскрыта впервые, то если x_i— конечная вершина, $h(x_i)$=0, то x_i отмечается как разрешимая; если x— дизъюнктивная вершина, $h(x) = \min_i (h(x_i) + c(x, x_i))$, то направить указатель от x к x_i и в случае разрешимости x_i отметить вершину x как разрешимую; если x— конъюнктивная вершина, $h(x) = \sum_{i=1}^{k} (h(x_i)+c(x,x_i))$, то направить указатель от x к той из не отмеченных как разрешимая вершин x_i, которая имеет наибольшую оценку, и в случае разрешимости всех x_i отметить x как разрешимую;

в) повторить процедуру обновления оценок и направления указателей для всех вершин, предшествующих *x,* вплоть до начальной вершины. К шагу 2.

Сделаем ряд замечаний к этому алгоритму.

1) Мы выбираем направления указателя от конъюнктивной вершины к той из ее дочерних вершин, которая имеет наибольшую оценку. Эвристическое основание для такого выбора состоит в том, что мы хотим как можно раньше опровергнуть гипотезу о том, что построенный к настоящему моменту частично раскрытый граф является верхней частью решающего графа. Действительно, если конъюнктивная вершина *x* на самом деле входит в решающий граф, то порядок раскрытия ее дочерних вершин безразличен, поскольку придемся раскрыть все дочерние вершины. Но если это не так, то желательно отбросить *x* и граф, начинающийся с *x,* как можно быстрее. Мы предполагаем, что выбор вершины *x* с максимальной оценкой в наибольшей степени повысит суммарную стоимость построенного графа.

2) Поскольку в аддитивном графе между двумя вершинами может быть больше одного пути, обновление оценки в данной вершине может происходить несколько раз. Тем не менее эта процедура закончится, поскольку граф не имеет циклов.

На рис. 5. 5 показан пример поиска минимального решающего графа в графе (рис. 5.5, *а*). Стоимости дуг приравнены единице. Числа, стоящие рядом с вершинами, являются оценками стоимости минимального решающего графа, начинающегося с этой вершины. Разрешимые вершины отмечены жирными точками. На рис. 5.5,*б*—5.5,*ж* показаны шаги алгоритма с обновленными оценками, указателями и последовательной отметкой разрешимых вершин. На рис. 5.5, *з* представлен решающий граф.

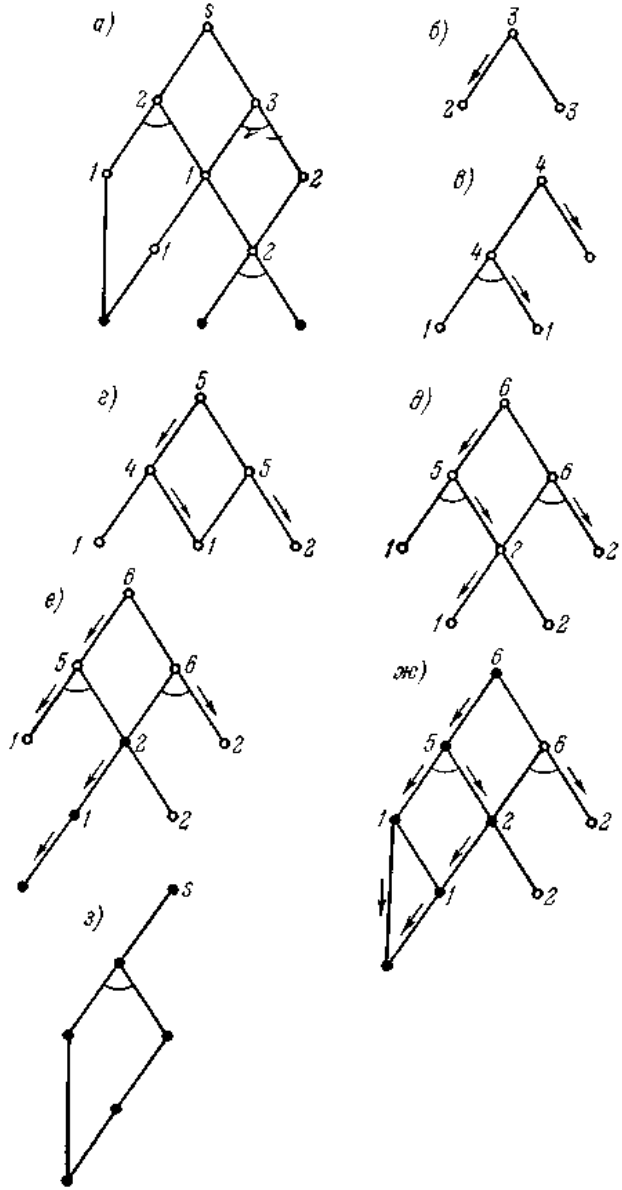


Рис. 5.5 Пример работы алгоритма а) исходный граф, *б), в), г), д), е), ж)* шаги алгоритма, *з*) решающий граф.

Использование в рассмотренных алгоритмах пропозициональных графов вместо деревьев устраняет избыточность, возникающую, когда порождаются идентичные подзадачи, а дальнейший поиск идет так, как будто бы они являются различными. Это приводит в дальнейшем к независимому, выращиванию эквивалентных поддеревьев, соответствующих сведению идентичных задач к их подзадачам. Таким образом, предписание в виде пропозициональных деревьев может в значительной степени снизить эффективность алгоритмов. В то же время использование графов требует распознавания и идентификации одинаковых подзадач, что может представить некоторые трудности чисто вычислительного характера Именно такой вид избыточности устраняется в аддитивных пропозициональных графах. Существует и другой вид избыточности. Он возникает, когда имеются различные сведения первоначальной задачи к одним и тем же подзадачам или, в терминологии обобщенного предписания, различные выводы *D* имеют одинаковые посылки. Задачу можно поставить в более общем виде устранение избыточности в случае, когда множество подзадач, полученное с помощью одного вывода, является подмножеством множества подзадач, полученного с помощью другого вывода По-видимому, эта задача не может быть решена в связи с тем, что одни и те же акты вывода могут одновременно принадлежать различным выводам. Устраняя избыточный вывод путем ликвидации соответствующего акта вывода, мы можем одновременно потерять другие выводы, которые, возможно, затем окажутся кандидатами наилучшего качества.

6. Решения коммуникационных задач формирования предписаний методами доказательства предписаний и/или директив

6.1. Структура процедур доказательства предписаний и/или директив

В п. 5.5 отмечено, что для решения многих задач может потребоваться логический анализ. Для этих целей мы ввели формальный язык — исчисление предикатов, на котором можно формулировать исходные предписания и/или директивы и делать из них логические выводы.

Благодаря общности и логической силе язык исчисления предикатов может претендовать на использование его в машинном формировании предписаниий и/или директив для широкого класса коммуникационных задач.

В этом разделе мы сначала опишем теоретическую модель, являющуюся основой для построения многих практических методов, а затем приведем наиболее эффективные методы доказательства предписаний и/или директив. Будем рассматривать процедуру доказательства предписаний и/или директив как пару (Т, ∑), где Т— аксиомы и правила вывода, а ∑ — стратегия поиска для Т.

Задача стратегии поиска состоит в том, чтобы выбрать из всего множества формул предписаниий и/или директив те, к которым на очередном шаге процедуры доказательства надо применять правило вывода. Задача правила вывода — получить непосредственное следствие из выбранного множества формул предписаниий и/или директив.

Увеличение эффективности программ, доказывающих предписаниия и/или директивы, осуществляется в основном за счет использования в процессе доказательства семантической информации, встраивания специфических свойств конкретных теорий в правила вывода и алгоритмы унификации и предоставления пользователю возможностей вмешательства в процесс доказательства.

В следующем параграфе мы рассмотрим теоретические основы, используемые при построении большинства программ доказательства предписаниий и/или директив. Изложение материала будет основываться на принципе резолюции, позволяющем на одной теоретической базе рассмотреть основные проблемы, возникающие при доказательстве предписаниий и/или директив.

6.2. Теоретические основы построения программ доказательства предписаниий и/или директив

В п.5.5 задача доказательства предписаниий и/или директив определена как в понятиях выполнимости, так и в понятиях выводимости. На практике будем использовать оба подхода. В данном разделе изложение строится на основе понятия выполнимости. При этом подходе задача доказательства предписаниий и/или директив состоит в определении невыполнимости множества дизъюнктов, полученных из исходного множества формул исчисления предикатов (п. 5.5.2).

Невыполнимость множества S дизъюнктов обозначает, что *S* ложно (т. е. конъюнкция дизъюнктов множества *S* принимает значение *F)* при

всех интерпретациях на любых областях. Данное определение не является конструктивным, так как невозможно рассмотреть все интерпретации на любых областях. Однако существует одна специальная область H такая, что S является невыполнимым тогда и только тогда, когда S является ложным при всех интерпретациях на этой области. Эта область называется *универсумом Эрбрана* для множества S и определяется следующим образом.

Пусть H_0—множество констант, появляющихся в множестве S дизъюнктов. Если в S нет констант, то в H_0 включается некоторая константа, например, $H_0=\{a\}$. Для i=0,1,2, . . .H_{i+1} есть объединение H_i и множества всех понятийных термов в форме $f^n(t_1, . . ., t_n)$ для всех n-местных функций f^n, встречающихся в S, где t_j, j=1, . . ., n, есть члены множества H_i. Тогда будем $H=H_\infty$, называть *эрбоановским универсумом* S, а H_i — его уровнем i.

Пример 6.1. Пусть

$S = \{P(x) \vee Q(a) \vee \sim P(f(x)), \sim Q(b) \vee P(g(x))\}$.

Тогда

$H_0 = \{a, b\}$,

$H_1\{a, b, f(a), f(b), g(a), g(b)\}$,

$H_2 = \{a, b, f(a), f(b), g(a), g(b), f(f(a)), f(f(b)), f(g(a)), f(g(b)), g(f(a)), g(f(b)), g(g(a)), g(g(b))\}$, ...

Пусть S — множество дизъюнктов, C — дизъюнкт множества S. *Фундаментальным примером дизъюнкта* C называется дизъюнкт, полученный замещением переменных (понятий) в C членами эрбрановского универсума S таким образом, что все вхождения одной и той же переменной (понятием) в C заменяются на один и тот же терм. *Эрбрановской базой* множества S дизъюнктов называется множество всех фундаментальных примеров атомов в S.

Пример 6.2. Пусть $S = \{R(x), P(g(y)) \vee Q(y)\}$. Определим для данного множества S эрбрановский универсум, фундаментальный пример дизъюнкта $R(x)$ и эрбрановскую базу.

Эрбрановский универсум для S равен

$$H=\{a, g(a), g(g(a)), ...\}.$$

Фундаментальным примером дизъюнкта $R(x)$ является, например, дизъюнкт

$$R(g(a)).$$

Эрбрановской базой (A) для S является множество:

$$A = \{R(a), P(a), Q(a), R(g(a)), P(g(a)), Q(g(a)), ...\}.$$

Определим над эрбрановским универсумом для S специальный вид интерпретации, называемый H-интерпретацией для S.

Пусть S —множество дизъюнктов, H—эрбрановский универсум для S и I — интерпретация S над H. Интерпретация I называется

H-интерпретацией S тогда и только тогда, когда она удовлетворяет следующим условиям:

1. *I* отображает все константы множества S сами в себя.

2. Пусть *f* есть *n*-местная функциональная буква и $h_1, \ldots, h_n$ суть элементы *H*. В интерпретации *I* функциональной букве (понятию) *f* ставится в соответствие функция, которая отображает $(h_1, \ldots, h_n)$ в $f(h_1, \ldots, h_n)$ (т. е. отображает H^n в *H*).

H-интерпретация не накладывает ограничений на соответствия, устанавливаемые для *n*-местных предикатных букв в *S*. Пусть $A = \{A_1, A_2, \ldots, A_n, \ldots\}$ является эрбрановской базой для *S*. Тогда *H* -интерпретация *I* может быть представлена множеством:

$$I=\{m_1, m_2, \ldots, m_n, \ldots\},$$

в котором m_j есть A_j или $\sim A_j$ для $j = 1,2, \ldots$ При этом, если m_j есть A_j, то A_j назначается значение *T,* иначе A_j назначается значение *F*,

Пример 6.3. Приведем для множества S из примера 6.2 некоторые *H*-интерпретации:

$I_1=\{R(a), P(a), Q(a), R(g(a)), P(g(a)), Q(g(a)), \ldots\}$,

$I_2=\{\sim R(a), \sim P(a), \sim Q(a), \sim R(g(a)), \sim P(g(a)), \sim Q(g(a)), \ldots\}$,

$I_3 = \{ R(a), \sim P(a), Q(a), R(g(a)), \sim P(g(a)), Q(g(a)), \ldots \}$.

Для компактности будем записывать I_1 в виде $I_1= \{R(x), P(x), Q(x)\}$ или $I_1=\{R, P, Q\}$. Аналогично $I_2= \{\sim R, \sim P, \sim Q\}$, а $I_3= \{ R, \sim P, Q\}$.

Справедлива следующая теорема.

Теорема 6.1. *Множество S дизъюнктов невыполнимо тогда и только тогда, когда S ложно при всех H-интерпретациях S.*

Таким образом, для установления невыполнимости множества дизъюнктов достаточно рассматривать не все его интерпретации, а только *H*-интерпретации.

Приведем без доказательства теорему Эрбрана.

Теорема 6.2. *Множество S дизъюнктов невыполнимо тогда и только тогда, когда существует конечное невыполнимое множество фундаментальных примеров дизъюнктов в S.*

Приведенная формулировка теоремы Эрбрана является основой для процедуры опровержения (установления невыполнимости множества дизъюнктов). Действительно, будем для множества *S* дизъюнктов генерировать множества $S_1, \ldots, S_n, \ldots$, где S_i есть множество всех фундаментальных примеров (предписаний) $S(H_i)$, полученных замещением переменных (понятий) в S константами из уровня *i* множества *H*, для S. Будем последовательно проверять на ложность множества S_i, i=l,2, ... Теорема Эрбрана гарантирует, что если множество невыполнимо, то данная процедура обнаружит такое *n,* что S_n является ложным.

Основным комбинаторным препятствием для достижения эффективности подобных процедур является огромная скорость роста конечных множеств S, с ростом *i*. Для каждого конечного множества *S* дизъюнктов, которое невыполнимо и для которого фактически можно построить опровержение, существует по крайней мере одно конечное подмножество *P* эрбрановского универсума для *S*, имеющее приемлемые размеры, такое, что S*(P)* невыполнимо, причем *P* минимально (в том смысле, что *S*(Q) выполнимо для любого собственного подмножества Q множества *P)*.

Ниже мы опишем основную идею принципа резолюции, являющегося теоретической базой для построения большинства методов доказательства предписаний и/или директив.

Введем необходимые определения.

6.2.1. Алфавитный порядок символов

Будем говорить, что множество всех символов *упорядочено в алфавитном порядке,* если

1) символы расположены в таком порядке: понятия, директивы, переменные, функции, предикаты, связка отрицания;

2) понятия расположены в алфавитном порядке;

3) директивы расположены в алфавитном порядке;

4) символ функции (предиката) меньшей размерности предшествует символам функций (предикатов) большей размерности, а при равных размерностях функции и предикаты упорядочены в алфавитном порядке предикатных и функциональных букв;

5) переменные расположены в алфавитном порядке.

6.2.2. Лексикографический порядок представлений и/или директив

Предписанием и/или директивой будем называть понятийные термы и функциональные литеры. Множество всех предписаний и/или директив *упорядочено в лексикографическом порядке* посредством следующего правила: короткие предписания и/или директивы предшествуют длинным, а при равенстве длин предписаний и/или директив *A* и *B* раньше ставится предписание и/или директива, у которого первый (первые) символ имеет ранний алфавитный порядок.

6.2.3. Подстановочные компоненты

Подстановочная компонента — это предписание и/или директива t/x, где x — переменная, t - терм, отличный от x, причем x называют переменной компоненты t/x, а t — термом компоненты.

6.2.4. Подстановки

Множество подстановочных компонент попарно различными переменными называется *подстановкой* и записывается в виде $\alpha=\{t_1/x_1, \ldots, t_n/x_n\}$. Применение подстановки к некоторой формуле A обозначает замену всех вхождений в A переменной x_i, $1 \le i \le n$, на вхождение терма t_i. Получившуюся формулу будем обозначать $A\alpha$ и называть α-примером A.
Например, применив к формуле $R(x, f(y))$ подстановку $\alpha=\{\varphi(a)/x, f(z)/y\}$, получим α-пример $R(\varphi(a), f(f(z)))$.

6.2.5. Композиция подстановок

Композицией $\alpha\beta$ *двух подстановок* α и β называется результат применения β к термам подстановки α с добавлением из β всех подстановочных компонент, содержащих переменные, отсутствующие в α. Например, если $\alpha=\{f(x, y)/z\}$, $\beta=\{a/x, b/y, c/w, d/z\}$, *то*

$$\alpha\beta=\{f(a, b)/z, a/x, b/y, c/w\}.$$

Можно показать, что применение к литере P последовательности подстановок α и β дает тот же результат, что и применение к P подстановки $\alpha\beta$. Можно также показать, что композиция подстановок ассоциативна: $(\alpha\beta)\gamma=\alpha(\beta\gamma)$.

6.2.6. Унификация

Множество литер $\{L_i\}$ называется *унифицируемым,* если существует такая подстановка α, что в результате применения ее к каждому элементу $\{L_i\}$ получаем, что $L_1\alpha=L_2\alpha=\ldots$ В этом случае подстановку α называют *унификатором* для $\{L_i\}$ и обозначают $\{L_i\}_\alpha$. Например, подстановка $\alpha=\{f(a)/x, b/y\}$ унифицирует множество литер $\{R(f(y), x), R(f(b), f(a))\}$ и дает в качестве ответа $\{R(f(b), f(a))\}$. *Наиболее общим* (простейшим) *унификатором* (НОУ) для $\{L_i\}$ называется такой унификатор α, что если β — какой-нибудь унификатор для $\{L_i\}$, дающий $\{L_i\}_\beta$, то найдется подстановка δ, для которой $\{L_i\}_{\alpha\beta}=\{L_i\}_\beta$. С точностью до алфавитных вариантов существует *единственный* НОУ.

6.2.7. Алгоритм унификации

Алгоритм, который находит НОУ для унифицируемого множества литер $\{L_i\}$ и сообщает о неудаче, если множество неунифицируемо, будем называть *алгоритмом унификации*. Приведем описание работы алгоритма.

1. Положить $k=0$ и $\sigma_k=\varepsilon$ (пустая подстановка). Перейти к пункту 2.
2. Если $\{L_i\}_{\sigma_k}$ не является одноэлементным множеством, то перейти к пункту 3. В противном случае положить $\sigma = \sigma_k$ и закончить работу.
3. Каждая из литер в $\{L_i\}_{\sigma_k}$ рассматривается как цепочка символов, и выделяются первые подпредписания литер, не являющихся одинаковыми у всех элементов $\{L_i\}_{\sigma_k}$. Указанные подпредписания, расположенные в лексикографическом порядке, образуют множество рассогласования B_k для $\{L_i\}_{\sigma_k}$. Пусть V_k—самый первый (левый), а U_k — следующий за ним элемент B_h. Тогда, если V_k является переменной, не входящей в U_k, то $\sigma_{k+1}=a_k\{U_k/V_k\}$, $k=k+1$ и перейти к пункту 2; в противном случае окончить работу.

Можно показать, что описанный процесс всегда завершается.

Поясним работу алгоритма на примере. Пусть $\{L_i\} = \{P(x, z, v), P(x, k(y), y), P(x, z, b)\}$. На первом шаге работы алгоритма будет получено множество рассогласования $B_0=\{z, k(y)\}$ и подстановка $\sigma_1= \{k(y)/z\}$. На втором шаге $\{L_i\}_{\sigma_1}$ не является одноэлементным множеством. $\{L_i\}_{\sigma_1} = \{P(x, k(y), v), P(x, k(y), y), P(x, k(y), b)\}$, множество $B_1=\{ v, y, b\}$, $\sigma_2=\{k(y)/z, y/v\}$. На третьем шаге $\{L_i\}_{\sigma_2} =\{P(x, k(y), y), P(x, k(y), y), P(x, k(y), b)\}=\{P(x, k(y), y), P(x, k(y), b)\}$, $B_2= \{y, b\}$, $\sigma_3=\{k(y)/z, y/v, b/y\}$. На четвертом шаге $\{L_i\}_{\sigma_3} =\{P(x, k(b), b)\}$ является одноэлементным множеством, а наиболее общим унификатором является подстановка

$$\{k(y)/z, y/v, b/y\}.$$

Дизъюнкт можно рассматривать как множество литер $\{L_i\}$. Если подмножество литер некоторого дизъюнкта $\{L_i\}$ унифицируемо с помощью НОУ α, то $\{L_i\}_\alpha$ называется *фактором* $\{L_i\}$. У одного дизъюнкта может быть несколько факторов, но число их конечно. Например, факторами дизъюнкта

$$R(g(x)) \vee R(x) \vee Q(a, g(u)) \vee Q(x, g(b)) \vee Q(z, w)$$

являются дизъюнкты

$$R(g(z)) \vee R(z) \vee Q(a, g(u)) \vee Q(z, g(b)), R(g(a)) \vee R(a) \vee Q(a, g(b)).$$

Первый фактор получен унификацией двух последних вхождений литеры Q в исходный дизъюнкт, а второй — трех. В приведенном предложении два вхождения литеры R нельзя унифицировать, так как в множестве рассогласования $\{x, g(x)\}$ переменная x входит в терм $g(x)$, что противоречит требованию, содержащемуся в пункте 3 алгоритма унификации.

6.2.8. Резольвента.

Пусть $\{L_i\}$ и $\{M_i\}$ — два дизъюнкта, не имеющие общих переменных (это можно всегда получить переименованием переменных). Пусть $\{l_i\}$ и $\{m_i\}$—такие два подмножества $\{L_i\}$ и $\{M_i\}$, что

1) $\{l_i\} \subseteq \{L_i\}$ и $\{m_i\} \subseteq \{M_i\}$;

2) для $\{l_i\}\ U\{\sim m_i\}$ существует наиболее общий унификатор α, т. е. $\{m_i\}_\alpha$ и $\{l_i\}_\alpha$ являются дополнительными.

Тогда говорят, что исходные дизъюнкты *разрешаются,* и из них выводим новый дизъюнкт, называемый *резольвентой:*

$$[\{L_i\}-\{l_i\}]_\alpha \cup [\{M_i\}-\{m_i\}]_\alpha.$$

Два дизъюнкта могут иметь более одной резольвенты, так как способ выбора $\{l_i\}$ и $\{m_i\}$ может оказаться не единственным. Если условия 1 и 2 не соблюдаются, то дизъюнкты не имеют резольвент.

Поясним процесс образования резольвент на примере. Пусть даны два дизъюнкта:

$$\{L_i\} = R\{y, g(a)) \vee R(y, g(x)) \vee P(x),$$
$$\{M_i\} = \sim R(z, g(a)) \vee \sim P(z).$$

Выбирая в качестве $\{l_i\}$ и $\{m_i\}$ соответственно $\{R(y, g(a))\}$ и $\{\sim R(z, g(a))\}$, мы получаем резольвенту $R(z,g(x)) \vee P(x) \vee \sim P(z)$. Если в качестве $\{l_i\}$ и $\{m_i\}$ выбрать соответственно $\{R(y, g(a)), R(y, g\{x))\}$ и $\{\sim R(z, g(a))\}$, то резольвентой будет $P(a) \vee \sim P(z)$. Всего для этих двух дизъюнктов можно образовать четыре резольвенты.

6.2.9. Резолюция (директива)

Пусть S — множество дизъюнктов. Будем называть *резолюцией (директивой)* правило вывода, генерирующее резольвенты из множества S дизъюнктов. Объединение S с множеством всех резольвент, которые могут быть образованы из дизъюнктов, входящих в S, будем обозначать $R(S)$. Обозначим $R_0(S)=S$ и определим $R_{n+1}(S) = R(R_n(S))$ для $n \geq 0$. Справедлива следующая теорема, устанавливающая полноту коммуникационной системы.

Теорема 6.3. *Если S — произвольное конечное множество дизъюнктов, то S невыполнимо тогда и только тогда, когда $R_n(S)$ содержит для некоторого $n \geq 0$ пустой дизъюнкт.*

Итак, мы определили формальную коммуникационную систему, использующую одно правило вывода (правило резолюции (директивы)) и не требующую логических аксиом.

На основе приведенной выше теоремы можно построить процедуру опровержения. Эта процедура состоит в вычислении для конечного множества *S* дизъюнктов последовательности множеств *S*, $R_1(S)$, $R_2(S)$, ... и может закончиться одним из трех исходов.

1. $R_n(S)$ содержит пустой дизъюнкт и, следовательно, множество S невыполнимо.
2. $R_n(S)$ совпадает с $R_{n+1}(S)$ и, следовательно, S выполнимо.
3. Процедура для определенных множеств S продолжает работу бесконечно в связи с неразрешимостью исчисления предикатов.

В некоторых случаях бесконечный процесс можно распознавать и прерывать работу.

Процесс опровержения, использующий принцип резолюции (директивы), удобно представлять в виде графа. Вершинам графа соответствуют дизъюнкты. Дизъюнкты исходного множества изображаются в виде *концевых вершин,* т. е. вершин, не имеющих предшественников. Если два дизъюнкта образуют резольвенту, то из этих дизъюнктов проводятся ребра к вершине—дизъюнкту, соответствующему выведенной резольвенте. Вершина графа, у которой нет следующих за ней вершин, называется *корневой вершиной.* Она соответствует дизъюнкту, выведенному этим графом. Пустой дизъюнкт будем обозначать символом *NIL.* Принято корень графа изображать внизу, а исходные дизъюнкты вверху. На рис. 6.1 приведен пример представления процесса опровержения в виде графа для невыполнимого множества дизъюнкюв $\{R(x) \vee \sim Q(x) \vee \sim P(x), R(x) \vee P(x), R(x) \vee Q(x), \sim R(x)\}$.

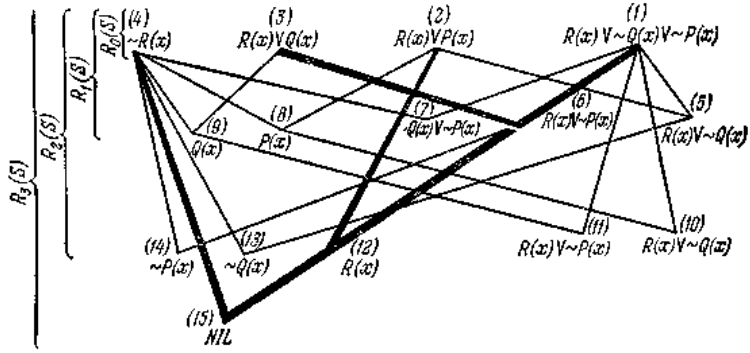


Рис. 6.1. Поиск опровержения с использованием бинарной резолюции (директивы).

На графе изображены дизъюнкты, получаемые в процессе применения бинарной резолюции (директивы) к исходному множеству.
Процедура опровержения, построенная непосредственно на основе теоремы о резолюции (директиве), является не вполне неэффективной, так как множества $R_1(S)$, $R_2(S)$,... очень быстро разрастаются. В следующих параграфах мы опишем практические процедуры доказательства, сокращающие перебор как за счет введения правил, ограничивающих принцип резолюции (директивы) (6.3 и 6.4), так и за счет применения более эффективных стратегий поиска (6.5).

6.3. Системы вывода в исчислении предикатов без равенства

Как было указано в 6.1, процедура доказательства может быть представлена парой (Т, ∑), где Т — аксиомы и правила вывода, а ∑ — стратегия поиска. В данном параграфе и в 6.4 будут рассмотрены правила вывода, ограничивающие принцип резолюции (директивы). Для лучшего понимания правил вывода мы будем приводить примеры их использования в контексте процедур доказательства. В этих примерах будет использоваться простейшая стратегия поиска — стратегия поиска в ширину (см. 5.3.2).

6.3.1. Семантическая резолюция (директива)

На рис. 6.1 мы привели пример процесса опровержения, полученного бинарной резолюцией (директивой). Из всех генерированных дизъюнктов для вывода *NIL* достаточны (6) и (12). Остальные дизъюнкты являются неуместными или избыточными. Рассмотрим на данном примере, как можно было бы избежать генерации лишних дизъюнктов. Один из способов состоит в разделении множества S исходных дизъюнктов на два подмножества, при этом для образования резольвенты используют дизъюнкты из разных подмножеств. На практике для разбиения множества на два класса используют интерпретацию.
Пусть *I*— некоторая интерпретация множества S дизъюнктов. Тогда в одно подмножество (S_1) включаются дизъюнкты, которым выбранная интерпретация не удовлетворяет, а в другое (S_2) включаются дизъюнкты, которым интерпретация удовлетворяет. Так как рассматриваемое множество дизъюнктов невыполнимо, то любая интерпретация разделит его на два непустых подмножества.

В рассматриваемом нами примере интерпретация $I= \{\sim R, \sim Q, \sim P\}$ разделит исходное множество дизъюнктов на S_1={(2), (3)} и S_2={(1), (4)}. Это позволит не генерировать дизъюнкт (7), так как он образуется из элементов одного подмножества.

Для ограничения количества генерируемых дизъюнктов используется понятие *упорядочения предикатных букв*. Упорядочим в исходном множестве (рис. 6.1) предикатные буквы следующим образом: $P>Q>R$. В двух дизъюнктах (один из S_1, а другой из S_2) будем резольвировать только такую литеру дизъюнкта из S_1, которая является *наибольшей (старшей) предикатной буквой* в данном дизъюнкте. Это ограничение позволит нам не резольвировать дизъюнкты (8) и (9), так как у них резольвируемая предикатная буква не является старшей $(R<P, R<Q)$.

Для примера на рис. 6.1 при $I=\{\sim R, \sim Q, \sim P\}$, $P>Q>R$ и введенных нами ограничениях на уровне $R_1(S)$ возможно сгенерировать только дизъюнкты (5) и (6). Им обоим удовлетворяет интерпретация I, следовательно, мы их включаем и подмножеаво S_2. После этого можно образовать два одинаковых дизъюнкта: R (из (3) и (5)) и R (из (2) и (6)). Дизъюнкту R выбранная интерпретация не удовлетворяет, и он включается в подмножество S_1. Но в S_2 есть дизъюнкт $\sim R$. Резольвируя R и $\sim R$, получим пустой дизъюнкт. Итак, существует два способа, которыми может быть получен дизъюнкт R:(((1) (2)) (3)) и (((1) (3)) (2)). Способы отличаются только порядком использования дизъюнктов. Так как для доказательства достаточно одного способа получения дизъюнкта, то необходимо избежать второго способа получения дизъюнкта R. Для этого вводится понятие *клаша*. Идея клаша состоит в том, чтобы генерировать R непосредственно из (1), (2) и (3) без образования «промежуточных» дизъюнктов (5) и (6).

Семантическая резолюция (директива) для ограничения резолюции (директивы) использует объединение указанных выше понятий: **интерпретации, упорядочения предикатных букв и клаша**.

Приведем теперь формальное определение *семантической резолюции (директивы)*.

Пусть I есть интерпретация. Пусть P — упорядочение предикатных букв. Конечное множество дизъюнктов $\{E_1, \ldots, E_q, N\}$, $q\geq 1$, называется *семантическим клашем* по отношению к P и I (PI-клашем) тогда и только тогда, когда $E_1, \ldots, E_q$ (называемые *сателлитами)* и N (называемое *ядром)* удовлетворяют следующим условиям:

1. Интерпретация I не удовлетворяет $E_1, \ldots, E_q$ (т. е. $E_1, \ldots, E_q$ ложны в I).
2. Пусть $R_1=N$. Для каждого $i=1, \ldots, q$ существует резольвента R_{i+1}, образованная из R_i и E_i.

3. Резольвируемая литера в E_i является наибольшей предикатной буквой в E_i, $i=1, \ldots, q$.

4. Интерпретация I не удовлетворяет R_{q+1} (т. е. R_{q+1} ложно в I). R_{q+1} называется *PI*-резольвентой (из *PI*-клаша $\{E_1, \ldots, E_q, N\}$).

Будем называть *семантической резолюцией (директивой) (PI-резолюцией)* правило вывода, генерирующее *PI*-резольвенты из множества дизъюнктов.

Пример 6.4. Пусть

$$E_1 = T(x) \vee Q(x), \quad E_2 = Q(x) \vee R(x),$$
$$N = \sim T(x) \vee \sim Q(x) \vee R(x).$$

Пусть $I=\{\sim T, \sim Q, \sim R\}$ и P есть упорядочение, в котором $T>Q>R$. Тогда $\{E_1, E_2, N\}$ есть *PI*-клаш. *PI*-резольвентой этого клаша является $R(x)$.

В этом примере ни $\{E_1, N\}$ ни $\{E_2, N\}$ не являются *PI*-клашем, так как интерпретация I удовлетворяет образованным резольвентам (нарушение условия 4).

Отметим, что при изменении упорядочения P на такое, что $R>Q>T$, множество дизъюнктов $\{E_1, E_2, N\}$ не является *PI*-клашем (нарушено условие 3).

В *PI*-клаше $\{ E_1, E_2, \ldots, E_q, N\}$ E_1 рассматривается как первый сателлит, E_2— второй, а E_q — последний. Фактически порядок сателлитов не является существенным. Мы можем пометить любой сателлит как первый, среди оставшихся любой как второй и т. д. Независимо от выбранного порядка мы получим одну и ту же *PI*-резольвенту. Следовательно, мы можем избежать генерации одной резольвенты многими способами.

Пусть I — интерпретация для множества S дизъюнктов и P — упорядочение предикатных букв, появляющихся в S. Вывод из S называется *PI-выводом* тогда и только тогда, когда каждый дизъюнкт в выводе является или дизъюнктом из S, или *PI*-резольвентой.

Пример 6.5. Пусть

$S = \{Q(a) \vee R(x), \sim Q(x) \vee R(x), \sim R(a) \vee \sim T(a), T(x)\}$

Пусть $I=\{\sim Q, R, \sim T)$ и P— упорядочение предикатных букв $R>Q>T$.

На рис. 6.2, *а* показан *PI*-вывод пустого дизъюнкта из S.

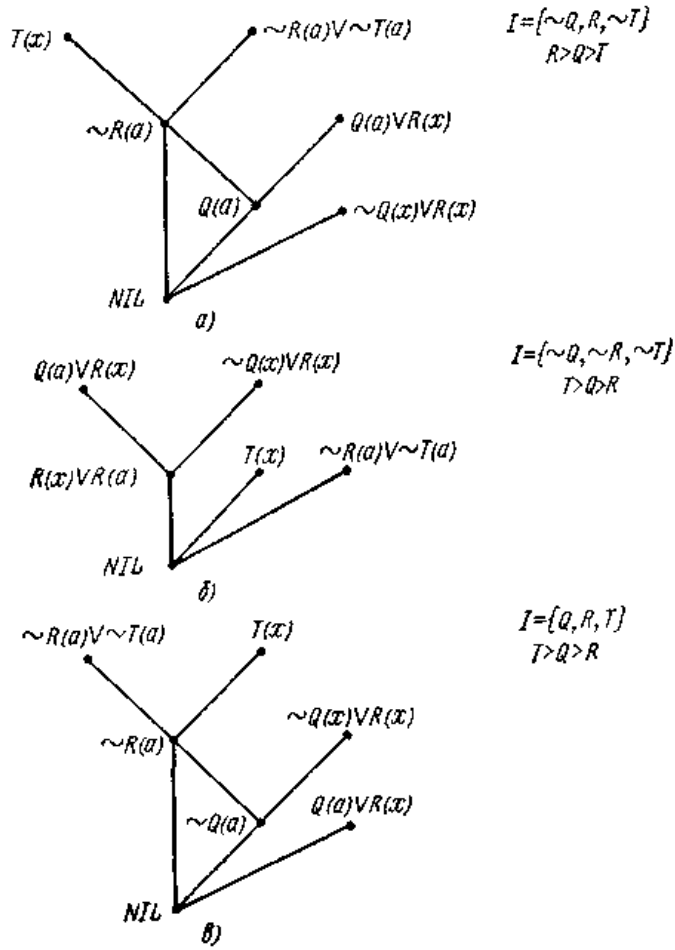


Рис. 6.2. Граф опровержения, полученный *PI*–резолюцией (директивой), положительной гиперрезолюцией (гипердирективой) и отрицательной гиперрезолюцией (гипердирективой).

Особенностью семантической резолюции (директивы) является тот факт, что можно использовать любую интерпретацию и любое упорядочение предикатных букв. Например, если в примере 6.5 выбрать *I={~Q, ~R, ~T}* и *T>Q>R*, то мы получим *PI*-вывод пустого дизъюнкта, изображенный на рис. 6.2, б.

Можно показать, что *PI* -резолюция (директива) полна, т. е. *если P есть упорядочение предикатных букв в конечном невыполнимом*

множестве S дизъюнктов и I есть интерпретация множества S, то существует PI-вывод пустого дизъюнкта из S.

6.3.2. Специализация семантической резолюции (директивы)

В этом разделе мы опишем два правила вывода, которые могут быть получены из семантической резолюции (директивы) выбором специальных интерпретаций. Одна интерпретация ведет к *гиперрезолюции* (*гипердирективе*), а другая — к *стратегии множества поддержки*.

Будем называть дизъюнкт *положительным,* если он не содержит знаков отрицания. Дизъюнкт называется *отрицательным,* если каждая его литера содержит знак отрицания. Дизъюнкт называется *смешанным,* если он не является ни положительным, ни отрицательным.

Выберем интерпретацию *I*, в которой каждая литера является отрицанием атома. В этой частной интерпретации *PI*-резолюция (директива) называется *положительной гиперрезолюцией* (*гипердирективой*). В положительной гиперрезолюции (гипердирективе) все сателлиты и *PI*-резольвенты являются положительными дизъюнктами.

Отрицательной гиперрезолюцией (*гипердирективой*) называется специальный случай *PI*-резолюции (директивы), в которой интерпретация *I* не содержит литер со знаками отрицания. В отрицательной гиперрезолюции (гипердирективе) все сателлиты и *PI*-резольвенты (гиперрезольвенты) являются отрицательными дизъюнктами.

Из полноты *PI*-резолюции (директивы) следует, что положительные и отрицательные гиперрезолюции (гипердирективы) полны.

Для примера 6.5 и упорядочения $T>Q>R$ вид положительной и отрицательной гиперрезолюции (гипердирективы) изображен на рис. 6.2, *б* и рис. 6.2, *в* соответственно.

Рассмотрим другую специализацию семантической резолюции (директивы).

Пусть *S* — конечное невыполнимое множество дизъюнктов и *T* есть подмножество S такое, что *S*—*T* выполнимо. Так как *S*— *T* выполнимо, то существует интерпретация *I,* которая удовлетворяет дизъюнктам множества *S* — *T.* Выберем данную интерпретацию. Пусть *P* — любое упорядочение предикатных букв в S. На основании полноты *PI*-резолюции (директивы) можно утверждать, что при выбранных *P* и *I* существует вывод *D* пустого дизъюнкта из S. Рассмотрим в *D* каждый

PI -клаш *{E_1, E_2, . . ., E_q, N}*. По определению *PI* -клаша сателлиты ложны в *I*, поэтому ни один дизъюнкт из S — *T* не является сателлитом.

PI-резольвента этого клаша может быть получена как результат последовательности бинарных резолюций (директив) E_1 и *N*, затем E_2 и резольвенты, полученной на предыдущем шаге (от E_1 и *N)*, и т. д. Таким образом, в каждой бинарной резолюции (директиве) одновременно оба резольвируемых дизъюнкта не принадлежат множеству *S—T*.

Резолюция (директива), удовлетворяющая указанному требованию, называется *резолюцией (директивой) множества поддержки (опорного множества),* а множество *T — множеством поддержки*. Разложив *PI*-клаши в *D* в последовательность бинарных резолюций (директив), мы получим *вывод множества поддержки,* т. е. вывод, в котором каждая резолюция (директива) есть резолюция (директива) множества поддержки.

Из приведенного рассуждения и полноты *PI*-резолюции (директивы) следует, что стратегия множества поддержки полна.

6.3.3. Семантическая резолюция (директива), использующая упорядоченные дизъюнкты

Упорядочение предикатных букв, используемое в *PI*- выводе, в общем случае не дает возможности выбрать в сателлите единственную литеру, которая должна резольвироваться.

Пример 6.6. Пусть

$$S = \{R(a) \vee R(b) \vee R(c) \vee R(d), \sim R(x)\},$$
$$I = \{\sim R(x)\}$$

и *P* - любое упорядочение. Тогда в *PI*-клаше

$\{R(a) \vee R(b) \vee R(c) \vee R(d), \sim R(x)\}$ каждая из четырех литер в сателлите имеет одинаковое старшинство и, следовательно, является кандидатом на резольвирование с ядром.

Стремление иметь единственного кандидата на резольвирование в каждом сателлите приводит к идее упорядочения дизъюнктов.

Упорядоченным дизъюнктом называется определенная последовательность литер. Упорядоченный дизъюнкт, так же как и дизъюнкт, является дизъюнкцией входящих в него литер. Различие состоит в том, что дизъюнкт рассматривается как множество литер, а упорядоченный дизъюнкт как некоторая последовательность литер.

Будем говорить, что *литера L_2 старше литеры L_1* в упорядоченном дизъюнкте (или L_1 *младше,* чем L_2) тогда и только тогда, когда L_2 следует за L_1 в последовательности, определенной упорядоченным

дизъюнктом. Отметим, что *старшая (наибольшая) литера* дизъюнкта является последней литерой дизъюнкта, а *младшая (наименьшая) литера* — первой.

Рассмотрим $R(a) \vee R$ (Ь) $\vee R(c)$ как упорядоченный дизъюнкт. В нем $R(c)$—старшая (наибольшая) литера, а $R(a)$— младшая литера.

Мы будем определять семантическую резолюцию (директиву), использующую упорядоченные дизъюнкты. Предварительно введем необходимые понятия.

Если две или больше литер (с одинаковыми знаками) упорядоченного дизъюнкта C имеют наиболее общий унификатор σ, то упорядоченный дизъюнкт, полученный из последовательности $C\sigma$ вычеркиванием любой литеры, идентичной младшей литере, называется *упорядоченным фактором дизъюнкта C.*

Пример 6.7. Для дизъюнкта $C=R(x) \vee Q(x) \vee R(a)$ первая и третья литеры имеют НОУ $\sigma = \{a/x\}$. Следовательно, $C\sigma = R(a) \vee Q(a) \vee R(a)$. В последовательности $C\sigma$ существуют две идентичные литеры (первая и третья литеры). В соответствии с определением младшей литерой считается литера, расположенная левее. Для получения упорядоченного фактора надо из $C\sigma$ удалить литеру, идентичную младшей литере. В рассматриваемом дизъюнкте это последняя литера. Таким образом, упорядоченным фактором является последовательность $R(a) \vee Q(b)$.

Пусть C_1 и C_2 — упорядоченные дизъюнкты. *Упорядоченная бинарная резольвента* дизъюнкта C_1 и дизъюнкта C_2 (не имеющих общих переменных) определяется следующим образом. Пусть L_1 и $\sim L_2$ — две литеры в C_1 и C_2 соответственно. Если L_1 и $\sim L_2$ имеют НОУ σ и C есть упорядоченный дизъюнкт, полученный из конкатенации последовательностей $C_1\sigma$ и $C_2\sigma$ путем устранения $L_1\sigma$ и $L_2\sigma$ и вычеркивания из оставшейся последовательности любой литеры, которая идентична младшей литере последовательности, то C называется *упорядоченной бинарной резольвентой.*

Заметим, что упорядоченная бинарная резольвента C_1 и C_2 не тоже самое, что упорядоченная бинарная резольвента C_2 и C_1.

Пример 6.8. Рассмотрим упорядоченные дизъюнкты $C_1=R(x) \vee Q(x) \vee T(x)$ и $C_2=\sim R(a) \vee Q(a)$. Вычислим упорядоченную бинарную резольвенту C_1 и C_2. $L_1=R(x)$, $L_2=\sim R(a)$, $\sigma=\{a/x\}$. Конкатенация $C_1\sigma$ и $C_2\sigma$ дает последовательность $R(a) \vee Q(a) \vee T(a) \vee \sim R(a) \vee Q(a)$. Устранив $L_1\sigma$ и $L_2\sigma$, получим последовательность $Q(a) \vee T(a) \vee Q(a)$. В оставшейся последовательности первая литера является младшей, а третья литера идентична ей. Устранив третью литеру, получим последовательность $Q(a) \vee T(a)$, являющуюся упорядоченной бинарной резольвентой C_1 и C_2.

Упорядоченной резольвентой упорядоченных дизъюнктов C_1 и C_2 называется одна из следующих упорядоченных бинарных резольвент:
1) C_1 и C_2;
2) C_1 и упорядоченного фактора C_2;
3) упорядоченного фактора C_1 и C_2;
4) упорядоченного фактора C_1 и упорядоченного фактора C_2.

Упорядоченной резолюцией (директивой) называется правило вывода, которое генерирует упорядоченные резольвенты из множества упорядоченных дизъюнктов. Можно показать, что упорядоченная резолюция (директива) полна.

Рассмотрим теперь *семантическую резолюцию (директиву) для упорядоченных дизъюнктов.*

Пусть I — интерпретация. Конечная последовательность упорядоченных дизъюнктов $\{E_1, E_2, \ldots, E_q, N\}$ называется *упорядоченным семантическим клашем* по отношению к I (для краткости *OI*-клашем) тогда и только тогда, когда $E_1, E_2, \ldots, E_q$ (называемые *упорядоченными сателлитами)* и N (называемое *упорядоченным ядром)* удовлетворяют перечисленным ниже условиям.

1. Интерпретация I не удовлетворяет $E_1, E_2, \ldots, E_q$.
2. Пусть $R_q=N$. Для каждого $i=q, q$—1, ..., 1 существует упорядоченная резольвента R_{i-1} сателлита E_i и R_i.
3. Резольвируемая литера в E_i является последней литерой в E_i, $i = 1, \ldots, q$, резольвируемая литера в R_i является наибольшей литерой среди литер, которым удовлетворяет I.
4. Интерпретация I не удовлетворяет R_o, R_o называется *OI*-резольвентой *OI* -клаша $\{E_1, E_2, \ldots, E_q, N\}$.

Пример 6.9. Рассмотрим упорядоченные дизъюнкты:
(1) $Q(a) \vee R(x)$,
(2) $S(x)$,
(3) $\sim R(x) \vee \sim S(a) \vee T(b)$.

Пусть I есть интерпретация, в которую все литеры входят со знаком отрицания. Интерпретация I не удовлетворяет дизъюнктам (1) и (2), поэтому они могут быть использованы как сателлиты. Так как (3) имеет литеры, которым удовлетворяет I, то (3) может быть использован как упорядоченное ядро. В (3) есть две литеры, которым удовлетворяет $I(\sim R(x)$ и $\sim S(a))$, следовательно, необходимы два сателлита. Пусть $R_2=N=\sim R(x) \vee \sim S(a) \vee T(b)$. В R_2 наибольшей из литер, которым удовлетворяет I, является $\sim S(a)$. Так как $\sim S(a)$ и последняя литера (2) могут резольвироваться, то $E_2=S(x)$. Упорядоченная резольвента R_1 сателлита E_2 и R_2 есть $R_1=\sim R(x) \vee T(b)$. В R_1 наибольшей литерой, которой удовлетворяет I, является $\sim R(x)$.

Так как ~ $R(x)$ и последняя литера (1) могут резольвироваться, то $E_1=Q(a) \vee R(x)$. Упорядоченная резольвента R_0 сателлита E_1 и R_1 есть $R_0=Q(a) \vee T(b)$ Так как интерпретация I не удовлетворяет R_0, мы заключаем, что (E_1, E_2, N) или ((1), (2), (3)) есть *OI*-клаш и $Q(a) \vee T(b)$ есть *OI*-резольвента этого клаша.

Из приведенного примера видно, что порядок упорядоченных сателлитов определяется порядком литер в упорядоченном ядре. Поэтому для рассмотренного примера ((2), (1), (3)) не является *OI* -клашем.

Если рассматривать дизъюнкты из примера 6.6 как упорядоченные дизъюнкты и $I=\{\sim R\}$, то для них существует только одна *OI*-резольвента $(R(a) \vee R(b) \vee R(c))$, в отличие от четырех *PI*-резольвент.

Пусть S — множество упорядоченных дизъюнктов и I— интерпретация для S. Вывод из S называется *OI-выводом* тогда и только тогда, когда каждый упорядоченный дизъюнкт в выводе есть или упорядоченный дизъюнкт из S, или *OI* -резольвента.

Для множества S дизъюнктов из примера 6.6 и $I=\{\sim R\}$ *OI*-вывод пустого дизъюнкта из S (рис. 6.3) требует всего четыре *OI* -резольвенты.

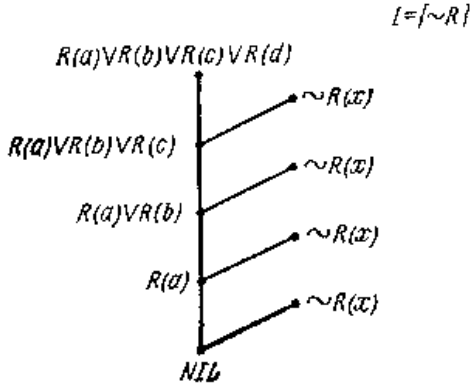


Рис. 6.3. Пример *OI*-вывода пустого дизъюнкта.

При использовании *PI*-резолюции (директивы) генерируется 40 *PI*-резольвент до того момента, пока метод насыщения уровня (поиск в ширину) не найдет пустой дизъюнкт.

6.3.4. Выполнение семантической резолюции (директивы)

Хотя *OI*-резолюция (директива) неполна, можно использовать понятие упорядоченных дизъюнктов для выполнения *PI*–резолюции (директивы).

Упорядоченным положительным дизъюнктом называется упорядоченный дизъюнкт, не содержащий литер со знаками отрицания. *Упорядоченным отрицательным дизъюнктом* называется упорядоченный дизъюнкт, в котором каждая литера содержит знак отрицания. *Упорядоченным неположительным (неотрицательным) дизъюнктом* называется упорядоченный дизъюнкт, не являющийся положительным (отрицательным). Ниже мы рассмотрим положительную гиперрезолюцию (гипердирективу). Отрицательная гиперрезолюция (гипердиректива) может быть представлена подобным образом. Будем рассматривать только положительные упорядоченные дизъюнкты как кандидаты в сателлиты и неположительные как ядра. Примем соглашение, что в любом упорядоченном неположительном дизъюнкте отрицательные литеры располагаются после положительных.

Пусть S—множество упорядоченных дизъюнктов и P — упорядочение предикатных букв в S. Приводимый ниже алгоритм вычисляет положительные гиперрезольвенты.

1. Пусть M и N есть множество всех положительных и неположительных упорядоченных дизъюнктов в S.
2. j=0.
3. $A_0 = \varnothing$, $B_0 = N$.
4. i=0.
5. Если A_i содержит *NIL*, то конец (опровержение найдено), иначе переход к следующему шагу.
6. Если B_i пусто, то переход к шагу 9, иначе переход к следующему шагу.
7. Вычислить множество W_{i+1},

W_{i+1} = {упорядоченные резольвенты C_1 и C_2, где C_1— упорядоченный дизъюнкт или упорядоченный фактор упорядоченного дизъюнкта в M, а C_2 — упорядоченный дизъюнкт в B_i. Резольвированная литера C_1 содержит наибольшую предикатную букву в C_1, а резольвированная литера в C_2 является «последней» литерой C_2}.

Пусть A_{i+1} и B_{i+1} будут множествами всех положительных и отрицательных упорядоченных дизъюнктов в W_{i+1} соответственно.

8. $i=i+1$, переход к шагу 5.

9. $T=A_0 \bigcup \ldots \bigcup A_i$ и $M = T \bigcup M$.

10. $j=j+1$, $i=1$.

11. Вычислить множество R,

R = {упорядоченные резольвенты C_1 и C_2, где C_1 есть упорядоченный дизъюнкт или упорядоченный фактор упорядоченного дизъюнкта в T, а C_2 есть упорядоченный дизъюнкт в N. Резольвированная литера в C_1 содержит наибольшую предикатную букву в C_1}. (Отметим, что резольвированная литера в C_2 может быть любой литерой в C_2.) Пусть A_1 и B_1 будут множествами всех положительных и неположительных упорядоченных дизъюнктов в R соответственно.

12. Переход к шагу 5.

В приведенном алгоритме j является номером уровня. Множество гиперрезольвент, сгенерированных на уровне j, размещается в множестве A_i,. При этом i указывает на количество сателлитов, участвующих в образовании данной гиперрезольвенты. В каждом уровне j B_i будет убывать до пустого множества, так как максимальное число отрицательных литер в любом упорядоченном дизъюнкте в B_i уменьшается на единицу при увеличении i на единицу.

Шаг 11 введен для того, чтобы на уровне (j+1) не генерировать резольвенты, полученные на уровне j. Можно показать, что если S невыполнимо, то пустой дизъюнкт генерируется приведенным алгоритмом.

С рассмотренным алгоритмом без потери свойства полноты может быть связана стратегия вычеркивания. То есть для T и M, полученных на 9-м шаге алгоритма, любой дизъюнкт в T или M, поглощаемый другими дизъюнктами в T или M, может быть вычеркнут. (Говорят, что дизъюнкт C_1 поглощает дизъюнкт C_2, если существует такая подстановка, что $C_1\sigma \subseteq C_2$.)

Пример 6.10. Пусть S есть множество упорядоченных дизъюнктов, $S=\{W(x) \vee S(x), \vee V(x) \vee S(x), S(x) \vee P(x), \sim P(x) \vee \sim Q(x), \; R(x) \vee \sim W(x), \; \sim S(x), \; Q(x) \vee \sim V(x) \vee \sim R(x)\}$.

Пусть предикатные буквы упорядочены следующим образом: $W>V>S>R>Q>P$.

Приведенный ранее алгоритм, примененный к множеству S породит следующие шесть положительных гиперрезольвент: $S(x) \vee R\{x)$, $P(x)$, $R(x)$, $S(x) \vee Q(x), Q(x)$, NIL.

Для получения гипервывода пустого дизъюнкта из S надо проследить последовательность порождения алгоритмом резольвент. Для рассматриваемого примера последовательность представлена на рис. 6.4, *а*.

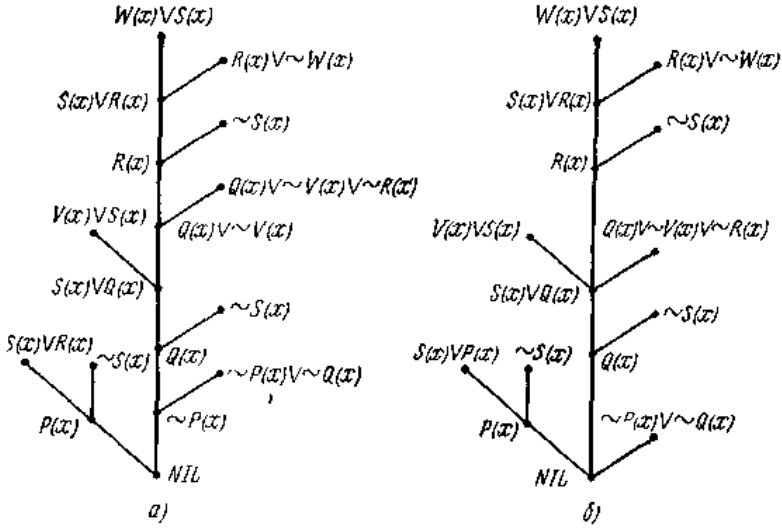


Рис. 6.4. Гипервывод пустого дизъюнкта.

Указанная последовательность естественным образом трансформируется в гипервывод *NIL* из 5 (рис. 6.4, *б)*.

6.3.5. Линейная резолюция (директива), использующая упорядоченные литеры и информацию о резольвированных литерах

Предварительно определим простейшее и эффективное правило вывода, называемое входной резолюцией (директивой). Пусть *S* — исходное множество дизъюнктов. Дизъюнкты множества S будем называть *входными дизъюнктами*. *Входной резолюцией* (*директивой*) называется бинарная резолюция (директива), у которой хотя бы один из дизъюнктов является входным. Эта стратегия хорошо ограничивает перебор, но не является полной.

Небольшое усложнение входной резолюции (директивы) приводит к линейному выводу.

Линейным выводом D из множества *S* дизъюнктов называется последовательность дизъюнктов (C_1, . . .,C_n), в которой $C_1 \in S$, а каждый член C_{i+1}, i=1, . . ., n—1, является резольвентой дизъюнкта C_i (называемого *центральным дизъюнктом)* и дизъюнкта *B* (называемого

боковым дизъюнктом), который удовлетворяет одному из двух условий:

1) $B \in S$ (в этом случае B называют *входным дизъюнктом* дизъюнкта C_{i+1});

2) B является некоторым дизъюнктом C_j, предшествующим в выводе дизъюнкту C_i, т. е. $j<i$ (в этом случае B называется *предшествующим дизъюнктом* дизъюнкта C_{i+1}).

Если C_{i+1} получен на основании первого из условий, то говорят, что имела место *входная резолюция* (*директива*), в противном случае — *резолюция* (*директива*) *предшествования.*

Пример 6.11. Рассмотрим невыполнимое множество W дизъюнктов, $W=\{\sim P\{x), P(x) \vee Q(x), \sim Q(x) \vee R(x), \sim R(x) \vee S(x), \sim R(x) \vee \sim S(x) \vee T(x), P(x) \vee \sim T(x)\}$.

На рис. 6.5, *а* приведен вид графа опровержения, удовлетворяющего условиям линейного вывода, для невыполнимого множества W дизъюнктов.

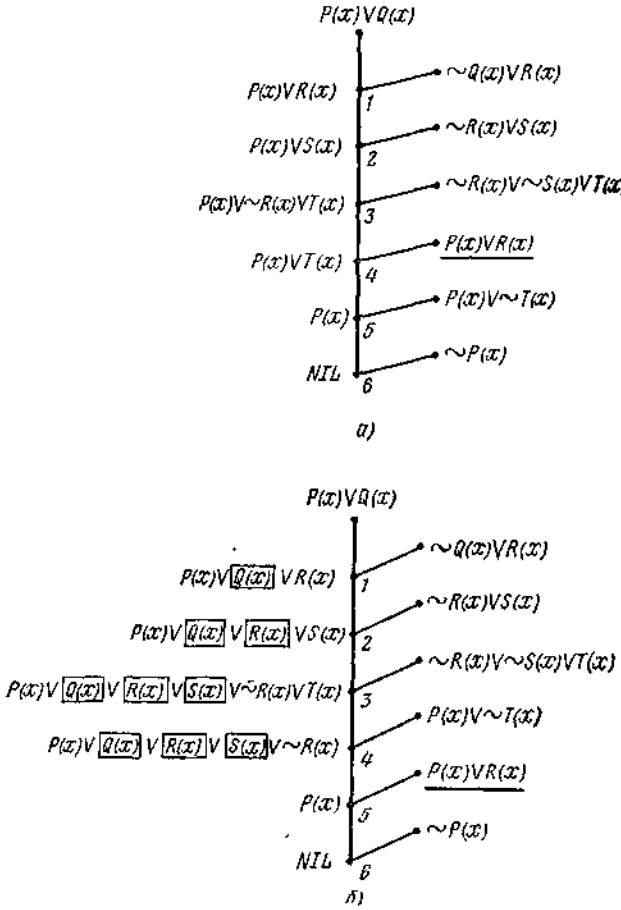


Рис. 6.5. Пример линейного опровержения.

Здесь слева изображены центральные дизъюнкты, а справа — боковые. Резольвенты 1, 2, 3, 5 и 6 получены входной резолюцией (директивой), а резольвента 4 получена резолюцией (директивой) предшествования.

Концевую вершину *A* в графе, удовлетворяющем условиям линейного вывода, будем называть *начальной,* если любая другая вершина графа либо является концевой, либо следует за *A*.

Сформулируем без доказательства теорему, утверждающую, что для любого невыполнимого множества дизъюнктов всегда существует граф опровержения, удовлетворяющий условию линейного вывода.

Теорема 6.4. *Пусть G — граф опровержения для невыполнимого множества S дизъюнктов и A — некоторый дизъюнкт из S, появляющийся в G. Тогда для S существует граф опровержения G', удовлетворяющий условиям линейного вывода, для которого A служит начальной вершиной.*

Приведенная теорема устанавливает полноту линейного вывода.

Отметим, что при выборе начальной вершины в графе опровержения, удовлетворяющем условиям линейного вывода, допускается некоторый произвол. Начальная вершина должна появляться в каком-нибудь графе опровержения *G*, т. е. она выбирается из некоторого подмножества $K \subseteq S$, содержащего дизъюнкты, появляющиеся в каком-нибудь опровержении. Например, в качестве *K* могут выбираться дизъюнкты, возникающие при отрицании доказываемого определения.

Линейная резолюция является довольно примитивным правилом вывода. Однако это правило можно усилить введением упорядоченных дизъюнктов и использованием информации о резольвированных литерах. Связывание понятия упорядоченных дизъюнктов (см. п. 6.3.3) с линейной резолюцией (директивой) не нарушает ее полноты, но существенно увеличивает эффективность метода.

Обычно при выполнении резолюции (директивы) образование резольвенты происходит путем устранения резольвированных литер. Однако оказывается, что эти литеры несут очень полезную информацию, которая может быть использована для усиления линейной резолюции (директивы). До введения механизма, использующего информацию о резольвированных литерах, рассмотрим простой пример. Пусть дано множество *S* дизъюнктов, линейное опровержение для которого изображено на рис. 6.5, *а*. Заметим, что в этом графе все резольвенты, кроме одной $(P(x) \vee R(x))$, удовлетворяют входной резолюции (директиве). Было бы полезно найти необходимое и достаточное условие, когда надо применять резолюцию (директиву) предшествования и с каким дизъюнктом. Это позволило бы в процессе линейного вывода, до тех пор пока не выполнено указанное условие, применять более эффективную входную резолюцию (директиву). Покажем, что это условие может быть определено, если используется понятие упорядоченных дизъюнктов и соответствующим образом записывается информация о резольвированных литерах. Вывод, использующий оба эти понятия, называется *линейным упорядоченным выводом (OL-выводом)*. До формального определения этого вывода рассмотрим его применение к множеству упорядоченных дизъюнктов из примера 6.11 (рис. 6.5, б).

Начальной вершиной выберем упорядоченный дизъюнкт $P(x) \vee Q(x)$, последняя литера которого резольвирует с дизъюнктом $\sim Q(x) \vee R(x)$.

Будем образовывать резольвенту по правилам, подобным образованию резольвенты для упорядоченных дизъюнктов. Отличие будет состоять в том, что вместо устранения обеих резольвированных литер будем оставлять в резольвенте первую из них, но помечать ее особым образом. Будем записывать резольвированные литеры в рамке и называть их *A*-литерами. Остальные литеры будем называть *B*-литерами. Если за *A*-литерой не следуют *B*-литеры, то *A*-литера (литеры) вычеркиваются.

В приведенном примере в качестве первой резольвенты получим дизъюнкт $P(x) \vee \boxed{Q(x)} \vee R(x)$. Аналогичным образом получаются вторая, третья и четвертая резольвенты (рис. 6.5, б).

Все четыре первые резольвенты образованы входной резолюцией. Четвертая резольвента имеет вид $P(x) \vee \boxed{Q(x)} \vee$

$\vee \boxed{R(x)} \vee \boxed{S(x)} \vee \sim R(x) \vee \boxed{T(x)}$. Так как за *A*-литерой $\boxed{T(x)}$ не следуют *B*-литеры, то эта *A*-литера вычеркивается. Отличительной особенностью полученной резольвенты

$$\left(P(x) \vee \boxed{Q(x)} \vee \boxed{R(x)} \vee \boxed{S(x)} \vee \sim R(x)\right)$$

является тот факт, что последняя *B*-литера *(~R(x))* является дополнительной к *A*-литере $\left(\boxed{R(x)}\right)$. Данное обстоятельство и является указанием на необходимость использования при образовании очередной резольвенты не входной резолюции (директивы), а резолюции (директивы) предшествования.

Так как за *A*-литерами в пятой резольвенте $\left(P(x) \vee \boxed{Q(x)} \vee \boxed{R(x)} \vee \boxed{S(x)} \vee \boxed{\sim R(x)}\right)$ не следуют *B*-литеры, то *A*-литеры удаляются и резольвента принимает вид *P(x)*. Шестая резольвента равна пустому дизъюнкту.

Введем теперь формальное определение *OL*-вывода.

Упорядоченный дизъюнкт *C* называется *уменьшающимся упорядоченным дизъюнктом* тогда и только тогда, когда последняя литера *C* унифицируется с отрицанием некоторой *A*-литеры в *C*.

При получении уменьшающегося упорядоченного дизъюнкта нет необходимости искать, с каким из полученных ранее дизъюнктов он образует резолюцию (директиву) предшествования. Вместо этого можно просто вычеркивать последнюю литеру в этом упорядоченном дизъюнкте. Мы будем называть это вычеркивание *операцией уменьшения.* Операция уменьшения позволяет не запоминать в *OL*-выводе промежуточных дизъюнктов. Этот аспект *OL*-вывода делает его очень удобным для выполнения на вычислительной машине. Операцию устранения *A*-литер, за которыми не следуют *B*-литеры, будем называть *операцией сокращения.*

Пусть C — уменьшающийся дизъюнкт и L — его последняя литера, унифицируемая с некоторой *A*-литерой наиболее общим унификатором σ. Тогда упорядоченный дизъюнкт, полученный из $C\sigma$ применением операций уменьшения и сокращения, будем называть *уменьшенным упорядоченным дизъюнктом* дизъюнкта C.

Итак, эффективность *OL*-вывода вызвана двумя факторами:

1. Обнаружением уменьшающегося дизъюнкта.
2. Введением операции уменьшения.

Упорядоченный фактор дизъюнкта C определим так же, как в п. 6.3.3. Условимся при образовании упорядоченного фактора применять операцию сокращения.

Упорядоченная бинарная резольвента дизъюнкта C_1 и дизъюнкта C_2 (не имеющих общих переменных) определяется аналогично определению, данному в п.6.3.3. Пусть L_1 и L_2 — две литеры в упорядоченных дизъюнктах C_1 и C_2 соответственно. Если L_1 и $\sim L_2$ имеют НОУ σ и C есть упорядоченный дизъюнкт, полученный из конкатенации последовательностей $C_1\sigma$ и $C_2\sigma$ путем

1) заключения в рамку $L_1\sigma$;
2) устранения $L_2\sigma$;
3) вычеркивания из оставшейся последовательности любой *B*-литеры, которая идентична младшей *B*-литере последовательности;
4) применения операции сокращения,

то дизъюнкт C называется *упорядоченной бинарной резольвентой* C_1 и C_2.

Упорядоченную резольвенту определим так же, как в разделе 6.3.3.

Пусть дано множество S упорядоченных дизъюнктов и упорядоченный дизъюнкт C_1 из S. Линейный вывод дизъюнкта C_n из S с начальным дизъюнктом C_1 называется *OL-выводом,* если выполнены следующие условия:

1. Для i=l, . . ., n—1, C_{i+1} является упорядоченной резольвентой дизъюнкта C_i и B_i (называемых соответственно *центральным* и

боковым), при этом резольвированная литера C_i (или упорядоченного фактора C_i) является последней литерой C_i.

2. B_i является или некоторым дизъюнктом C_j, $j<i$ (если C_i есть уменьшающийся дизъюнкт), или дизъюнктом из S (во всех остальных случаях). Если B_i есть некоторый дизъюнкт C_j, $j<i$, то C_{i+1} является уменьшенным дизъюнктом.

3. В выводе нет тавтологий.

Определение упорядоченного дизъюнкта может быть использовано для доказательства следующего утверждения.

В *OL*-выводе, если C_i есть уменьшающийся упорядоченный дизъюнкт, то существует центральный упорядоченный дизъюнкт C_j, $j<i$, такой, что уменьшенный дизъюнкт C_{i+1}, полученный из C_i, является упорядоченной резольвентой C_i и C_j. На рис. 6.5,*б* этот дизъюнкт подчеркнут.

Следующая теорема устанавливает полноту *OL*-резолюции (директивы).

Теорема 6.5. *Если C есть упорядоченный дизъюнкт n невыполнимом множестве S упорядоченных дизъюнктов и если S — {C} выполнимо, то существует OL-опровержение из S с упорядоченным дизъюнктом C как начальным дизъюнктом.*

Легко показать, что *OL*-резолюция (директива) включает в себя резолюцию (директиву) множества поддержки, т. е. полнота *OL*-резолюции (директивы) гарантирует полноту резолюции (директивы) множества поддержки.

6.3.6. Линейный вывод

Рассмотрим вопросы эффективного выполнения линейного вывода. Предположим, что для невыполнимого множества *S* дизъюнктов в качестве начального выбран дизъюнкт C_1. После резольвирования C_1 со всеми возможными боковыми дизъюнктами мы получим резольвенты $R_1, \ldots, R_m$. Каждый R_i, $1 \le i \le m$, является возможным центральным дизъюнктом. Если некоторый R_i является пустым дизъюнктом, то доказательство завершено. Иначе, для каждого i мы находим все возможные боковые дизъюнкты, которые могут резольвировать с R_i. Указанный процесс продолжается до получения пустого дизъюнкта. Для удобства представления указанного процесса будем изображать центральные дизъюнкты в виде вершин графа, а боковые в виде помеченных дуг, связывающих соответствующий центральный дизъюнкт и образующуюся резольвенту.

Проиллюстрируем сказанное на примере.

Пример 6.12. Пусть множество $S=\{R(x) \vee Q(x),\ \sim R(x) \vee Q(x),\ R(x) \vee \sim Q(x),\ \sim R(x) \vee \sim Q(x)\}$. В качестве начального дизъюнкта выберем дизъюнкт $R(x) \vee Q(x)$. На рис. 6.6 изображено дерево поиска пустого дизъюнкта с *OL*-резолюцией. (Для иллюстративных целей тавтологии в примере не устраняются.) Номер у вершины указывает порядок получения резольвенты.

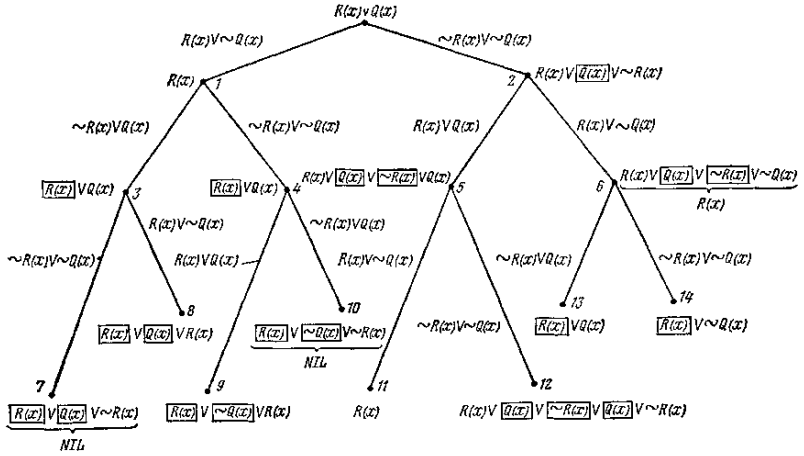


Рис. 6.6. Представление *OL*-опровержения в виде дерева поиска.

Нетрудно видеть, что задача нахождения *OL*-опровержения в указанной постановке подобна задаче эвристического поиска. В понятиях эвристического поиска задача нахождения опровержения, представленная на рис. 6.6, решена методом *поиска в ширину*. Здесь состояния представлены центральными дизъюнктами, а операторами являются боковые дизъюнкты. Начальным состоянием является дизъюнкт $d=\{R(x) \vee Q(x)\}$, а конечным *NIL*.

Следует отметить, что *OL*-вывод эффективно конкурирует практически со всеми методами за счет простоты организации поиска. Эта простота объясняется как тем, что в *OL*-выводе один из резольвируемых дизъюнктов определен, так и тем, что здесь не требуется запоминать промежуточные дизъюнкты.

6.4. Правила вывода в исчислении предикатов с равенством

Равенство является важным и широко используемым в теории коммуникаций отношением. Многие определения могут быть легко

записаны с использованием равенства. Если отношение равенства используется для выражения некоторого определения и доказательство ведется с использованием единственного правила резолюции (директивы), то, кроме аксиом, необходимо добавить набор аксиом, описывающих свойства равенства.

Аксиомы равенства для множества S дизъюнктов имеют вид:

1. $\{x=x\}$ (рефлексивность).
2. $\{x\neq y \vee y=x\}$ (симметричность).
3. $\{x\neq y \vee y\neq z \vee x=z\}$ (транзитивность).
4. $\{ x_j\neq x_0 \vee \sim P(x_1,\ .\ .\ .,\ x_j,\ .\ .\ .,\ x_n) \vee P\{x_1, .\ .\ ., x_0, .\ .\ ., x_n)\}$, где $j = 1, .\ .\ ., n$, для каждой n-местной предикатной буквы P, встречающейся в S.
5. $\{x\neq x_o \vee f(x_1,\ .\ .\ .,\ x_j,\ .\ .\ .,\ x_n)=f(x_1,\ .\ .\ .,\ x_0,\ .\ .\ .,\ x_n)\}$, где $j = 1, .\ .\ ., n$, для каждой n-местной функциональной буквы f, встречающейся в S.

Аксиомы 4 и 5 определяют свойство подстановочности равенства.

В приведенных аксиомах для обозначения предикатного символа равенства мы используем = и пишем $a= b$ и $a\neq b$ вместо $=(a, b)$ и $\sim(=(a, b))$ соответственно.

Введение равенства с помощью аксиом и универсального правила вывода (резолюции) имеет определенные недостатки. Покажем это на примере.

Пусть дано: 1) $Q(a)$; 2) $a=b$. Надо доказать 3) $Q(b)$.

Если пользоваться аксиомами равенства и принципом резолюции (директивы), то сначала получим 4) $a\neq x_2 \vee Q(x_2)$, резольвируя выражение 1) и аксиому равенства $x_1\neq x_2 \vee \sim Q(x_1) \vee Q(x_2)$. Затем, резольвируя 4) и 2), получим $Q(b)$. Естественно было бы не получать вспомогательное выражение 4), т. е. с помощью специальных правил вывода получить доказательство, имеющее вывод меньшей длины. На практике более существенным оказывается даже не увеличение длины вывода, а тот факт, что появившиеся бесполезные дизъюнкты (выражение 4)) приводят в свою очередь к появлению новых бесполезных дизъюнктов, «загрязняют» пространство поиска, а это приводит к снижению эффективности метода.

Следует указать, что сам факт присутствия аксиом равенства, увеличивающих общее количество формул, также является недостатком.

Существуют другие способы введения отношения равенства. Например, Дарлингтон использовал аксиомы второго порядка, Робинсон ввел обобщенный принцип резолюции, который обеспечил встраивание равенства, Моррис предложил E-резолюцию. Мы будем в этом параграфе рассматривать правило парамодуляции, являющееся наиболее естественным из предложенных правил.

Введем необходимые определения. Мы определили ранее, что множество *S* дизъюнктов является невыполнимым тогда и только тогда, когда *S* ложно на всех интерпретациях. Однако существует много множеств дизъюнктов, которые истинны в одних интерпретациях, но ложны в других. Пусть *W* есть множество всех интерпретаций *S*. Пусть *Q* является непустым подмножеством *W*. Будем называть множество S дизъюнктов *Q-невыполнимым* тогда и только тогда, когда S ложно для каждого элемента *Q*.

Определим теперь *R*-невыполнимые множества, где *R* обозначает равенство.

R-интерпретация I множества *S* дизъюнктов есть интерпретация, удовлетворяющая следующим условиям:

а) $(\alpha=\alpha) \in I$;

б) если $(\alpha=\beta) \in I$, то $(\beta=\alpha) \in I$;

в) если $(\alpha=\beta) \in I$ и $(\beta=\gamma) \in I$, то $(\alpha=\gamma) \in I$;

г) если L' есть результат замещения некоторого одного появления α в L на β и $(\alpha=\beta) \in I$, то $L' \in I$.

Здесь α, β и γ— любые термы в эрбрановском универсуме для *S*, а *L* — любая литера в *I*.

Фактически *R*-интерпретация удовлетворяет условиям рефлексивности, симметричности, транзитивности и подстановочности, т. е. является моделью теории равенства. Многие авторы рассматривали специальные классы моделей различных теорий (частичного упорядочения, теории множеств и т. п.). Довольно общий подход к решению этой задачи предложен Слэйглом и продемонстрирован им на примерах теории равенства, частичного упорядочения и теории множеств, а позднее на теориях с коммутативностью и ассоциативностью. Цель этого подхода заключается в том, чтобы заменить некоторые аксиомы рассматриваемой теории полными (по опровержению) и эффективными (по времени) правилами вывода. Новые правила позволяют исключить огромное число лишних следствий, возникающих при использовании правила резолюции и аксиом теории. На основании этих идей частичное и общее упорядочение были встроены в правила вывода и получены экспериментальные результаты.

В этой работе будут рассматриваться только модели теории равенства, так как подход к теории равенства и другим теориям является подобным.

Множество *S* дизъюнктов называется *R-удовлетворимым,* если существует *R*-интерпретация, удовлетворяющая всем дизъюнктам в S. Иначе S называется *R-неддовлетворимым* (*R*-невыполнимым).

Пусть S — множество дизъюнктов. Определим *множество F функционально рефлексивных аксиом* для S как $F=\{f(x_1, ...,x_n)=f(x_1,... x_n)$ для всех n-местных функциональных букв f встречающихся в $S\}$. S будем называть *функционально-рефлексивной системой,* если S содержит F и $\{x=x\}$, и просто *рефлексивной системой,* если S содержит $\{x=x\}$, но не содержит F.

Теорема 6.6. *Пусть S — множество дизъюнктов и P — множество аксиом равенства для S. S является R-невыполнимым тогда и только тогда, когда $S \cup P$ невыполнимо.*

Для R-невыполнимых множеств справедливо расширение теоремы Эрбрана.

Теорема 6.7. *Конечное множество S дизъюнктов R-невыполнимо тогда и только тогда, когда существует конечное множество S' фундаментальных примеров дизъюнктов в S такое, что S' является R-невыполнимым.*

6.4.1. Парамодуляция

Определим правило вывода для равенства, называемое *парамодуляцией*. Используя резолюцию и парамодуляцию, мы всегда можем вывести пустой дизъюнкт из R-невыполнимого множества дизъюнктов.

По правилу парамодуляции из дизъюнкта C_1, равного $\{L(t)\vee C'_1\}$, и дизъюнкта C_2, равного $\{r=s \vee C'_2\}$, не имеющих общих переменных (где C'_1 и C'_2 — дизъюнкты, r и s — термы и $L(t)$ — литера, содержащая терм t) и таких, что t и r имеют НОУ σ, выводится дизъюнкт, называемый *бинарным парамодулянтом* C_1 и C_2:

$$L\sigma[s\sigma] \cup C'_1\sigma \cup C'_2\sigma,$$

где $L\sigma[s\sigma]$ обозначает результат замены одного появления $t\sigma$ в $L\sigma$ на $s\sigma$. Литеры L и $r=s$ называют *парамодулированными литерами*.

Пример 6.13. Рассмотрим дизъюнкты:

C_1: $Q(g(f(x))) \vee R(x)$;

C_2: $f(g(b))=a \vee T(g(c))$.

Образуем бинарный парамодулянт C_1 и C_2. Здесь L есть $Q(g(f(x)))$, C'_1 есть $R(x)$, r есть $f(g(b))$, s есть a и C'_2 есть $T(g(c))$. L содержит терм t, равный $f(x)$, который унифицируется с r. НОУ t и r есть $\sigma=\{g(b)/x\}$. Следовательно, $L\sigma[t\sigma]$ равно $Q(g(f(g(b))))$, а $L\sigma[s\sigma]$ есть $Q(g(a))$. Так как $C'_1\sigma$ равно $R(g(b))$ и C'_2 равно $T(g(c))$ то бинарный парамодулянт C_1 и C_2 равен

$$Q(g(a)) \vee R(g(b)) \vee T(g(c)).$$

Парамодулированными литерами являются $Q(g(f(x)))$ и $f(g(b))=a$.

Парамодулянт дизъюнктов C_1 и C_2 есть один из следующих бинарных парамодуляптов:

1) бинарный парамодулянт C_1 и C_2;
2) бинарный парамодулянт C_1 и фактора C_2;
3) бинарный парамодулянт фактора C_1 и C_2;
4) бинарный парамодулянт фактора C_1 и фактора C_2.

Приведем пример совместного использования правила парамодуляции и резолюции (директивы) для получения опровержения.

Пример 6.14. Если $x^2=e$ для всех x в группе, то группа коммутативна.

1. $f(e, x) = x$.
2. $f(x, e)=x$.
3. $f(x, f(y, z))=f(f(x, У), z)$.
4. $f(x_t x)=e$.
5. $f(a, b)=c$.
6. $c\neq f(b,a)$.
7. $f(x, e)=f(f(x, y), y)$, парамодулянт 3 и 4, $t=f(y,z)$.
8. $x=f(f(x, y), y)$, парамодулянт 7 и 2, $t=f(x, e)$
9. $a=f(c, b)$, парамодулянт 8 и 5, $t=f(x, y)$.
10. $f(y, f(y, z))=f(e, z)$, парамодулянт 3 и 4, $t=f(x,y)$.
11. $f(y, f(y, z))=z$, парамодулянт 10 и 1, $t=f(e, z)$.
12. $f(c, a)=b$, парамодулянт 11 и 9, $t=f(y, z)$.
13. $c=f(b, a)$, парамодулянт 8 и 12, $t=f(x, y)$.
14. *NIL*, резольвента 13 и 6.

Приведем алгоритм получения опровержения, используя правила парамодуляции и резолюции. Пусть S_0 — множество всех факторов дизъюнктов из S. Для нечетного $i>0$ пусть S_i формируется из S_{i-1} добавлением всех дизъюнктов, которые могут быть получены парамодуляцией двух дизъюнктов в S_{i-1}. Для четного $i>0$ пусть S_i формируется из S_{i-1} добавлением всех факторов дизъюнктов, которые могут быть получены резольвированием двух дизъюнктов в S_{i-1}. Если на некотором шаге получен пустой дизъюнкт, то это обозначает *R*-невыполнимость множества S.

Робинсон и Вос показали, что если S является конечным функционально-рефлексивным множеством дизъюнктов, то приведенный выше алгоритм является полуразрешающей процедурой для *R*-невыполнимости.

Непосредственное использование резолюции (директивы) и парамодуляции неэффективно, поэтому используются различные стратегии. Ограничивающие стратегии для парамодуляции более слабы, чем для резолюции (директивы). Неполны аналоги *PI*-резолюции и *OL*-резолюции (директивы) в парамодуляции. Гиперрезолюция

(гипердиректива) и линейная резолюция могут быть распространены на парамодуляцию (гипермодуляция и линейная парамодуляция).

6.4.2. Гиперпарамодуляция

Пусть P — упорядочение предикатных букв, которое включает буквы в дизъюнктах C_1 и C_2. Парамодулянт C_1 и C_2 называется *P-гиперпарамодулянтом* тогда и только тогда, когда выполняются следующие условия:

1. C_1 и C_2—положительные дизъюнкты.
2. Парамодулированные литеры в C_1 и C_2 содержат наибольшую предикатную букву в C_1 и C_2 соответственно.

Напомним, что P-гиперрезольвента есть PI-резольвента, когда каждая литера в интерпретации I содержит знак отрицания. Заметим, что в этом случае сателлиты и P-гиперрезольвенты являются положительными дизъюнктами.

Пусть P — упорядочение предикатных букв в множестве S дизъюнктов. Тогда *P-гипервывод с резолюцией (директивой) и парамодуляцией* есть вывод, в котором каждый дизъюнкт есть дизъюнкт из S или P-гиперрезольвента, или P-гиперпарамодулянт. *P-гиперопровержение с резолюцией (директивой) и парамодуляцией* есть P-гипервывод с резолюцией (директивой) и парамодуляцией пустого дизъюнкта.

Можно показать, что P-гиперпарамодуляция полна, т. е. справедлива следующая теорема.

Теорема 6.8. *Если P — упорядочение предикатных букв в конечном R-невыполнимом множестве S дизъюнктов, то существует P-гиперопровержение с резолюцией (директивой) и парамодуляцией из $(S \cup \{x=x\} \cup F)$, где F—множество функционально-рефлексивных аксиом для S.*

Полнота гиперпарамодуляции говорит, что можно рассматривать для резолюции (директивы) только клаши с положительными сателлитами, а для парамодуляции — только положительные дизъюнкты. Эта процедура вывода особенно хорошо работает при малом количестве положительных дизъюнктов. Отметим, что упорядочение P предикатных букв может играть важную роль в дедуктивном коммуникационном предписании. Действительно, введя для предикатной буквы равенства наименьший (наибольший) порядок, мы будем получать больше (меньше) резолюций (директив), чем парамодуляции. Если предикатной букве равенства присвоить

наименьший порядок, то мы получим следующее следствие теоремы 6.8.

Следствие 6.1. *Если P есть упорядочение предикатных букв в конечном R-невыполнимом множестве S дизъюнктов таких, что букве равенства R (*или *=) присвоен наименьший порядок в* P, *и если F есть множество функционально-рефлексивных аксиом для S, то пустой дизъюнкт выводится из* $(S \cup \{x=x\} \cup F)$ *резолюцией* (*директивой*) *и парамодуляцией, в которых*

а) *все резольвенты суть P-гиперрезольвенты;*

б) *парамодуляция выполняется только между дизъюнктами, состоящими из положительных литер с единственной предикатной буквой (буквой равенства), и положительными дизъюнктами.*

Заметим, что если мы используем *P*-гиперпарамодуляцию и *P*-гиперрезолюцию (гипердирективу), то множество *F* функционально-рефлексивных аксиом требуется для получения доказательства. Например, рассмотрим $S=\{a=b,\ f(a)\neq f(b)\}$. *S R*-невыполнимо, но не существует *P*-гиперопровержения с резолюцией (директивой) и парамодуляцией из $S \cup \{x=x\}$. Однако сущестует *P*-гиперопровержение с резолюцией (директивой) и парамодуляцией из $\{S \cup \{x=x\} \cup F\}$.

Пример 6.15. Произведем на примере поиска опровержения невыполнимого множества *S* дизъюнктов сравнение парамодуляции и гиперпарамодуляции. Для обоих методов будем использовать поиск в ширину.

Для данного множества *S* дизъюнктов мы будем вырабатывать последовательности дизъюнктов $S_0, S_1, \ldots, S_n, \ldots$ до тех пор, пока не получим пустой дизъюнкт. Пусть $S_0=S$.

$$S_n = \begin{cases} \text{резольвенты из } S_0 \cup \ldots \cup S_{n-1} \text{ для нечетных } n, \\ \text{парамодулянты из } S_0 \cup \ldots \cup S_{n-1} \text{ для четных } n. \end{cases}$$

Кроме того, после генерации нового дизъюнкта будем пользоваться стратегией вычеркивания поглощаемых дизъюнктов. Вычеркнутые дизъюнкты будем помечать звездочкой.

Пусть $S=\{\sim Q(a) \vee \sim S(a) \vee \sim T(a) \vee a=b, Q(a), S(a), T(a), f(a)\neq(b), f(x)=f(x)\}$.

Пусть *P* будет следующее упорядочение предикатных букв: $Q>S>T>=$.

А) Рассмотрим *P*-гиперпарамодуляцию с ограничениями следствия 6.1.

S_0: *1. $\sim Q(a) \vee \sim S(a) \vee \sim T(a) \vee a=b$.
2. $Q(a)$.
3. $S(a)$.
4. $T(a)$.

5. $f(a) \neq f(b)$.
6. $f(x)=f(x)$

S_1: 7. $a=b$, Р-гиперрезольвента 1, 2, 3 и 4.

S_2: 8. $Q(b)$, Р-гиперпарамодулянт 2 и 7.
9. $S(b)$, Р-гиперпарамодулянт 3 и 7.
10. $T(b)$, Р-гиперпарамодулянт 4 и 7.
11. $f(a)=f(b)$, } Р-гиперпарамодулянты 6 и 7.
12. $f(b)=f(a)$, }
13. *NIL*, Р-гиперрезольвента 5 и 11.

Б) Рассмотрим теперь парамодуляцию без ограничений.

S_0: *1. $\sim Q(a) \vee \sim S(a) \vee \sim T(a) \vee a=b$.
2. $Q(a)$.
3. $S(a)$.
4. $T(a)$.
5. $f(a) \neq f(b)$.
6. $f(x)=f(x)$.

S_1. *7. $\sim S(a) \vee \sim T(a) \vee a=b$, резольвента 1 и 2.

S_2: *8. $\sim S(a) \vee \sim T(a) \vee Q(b)$, парамодулянт 2 и 7.
*9. $\sim S(a) \vee \sim T(a) \vee S(b)$, парамодулянт 3 и 7.
*10. $\sim S(a) \vee \sim T(a) \vee T(b)$, парамодулянт 4 и 7.
*11. $\sim S(a) \vee \sim T(a) \vee f(b) \neq f(b)$, парамодулянт 5 и 7.
*12. $\sim S(a) \vee \sim T(a) \vee f(a)=f(b)$, } парамодулянты 6 и 7.
*13. $\sim S(a) \vee \sim T(a) \vee f(b)=f(a)$, }

S_3: *14. $\sim T(a) \vee a=b$, резольвента 3 и 7.
*15. $\sim T(a) \vee Q(b)$, резольвента 3 и 8.
*16. $\sim T(a) \vee S(b)$, резольвента 3 и 9.
*17. $\sim T(a) \vee T(b)$, резольвента 3 и 10.
*18. $\sim T(a) \vee f(b) \neq f(b)$, резольвента 3 и 11.
*19. $\sim T(a) \vee f(a)=f(b)$, резольвента 3 и 12.
*20. $\sim T(a) \vee f(b)=f(a)$, резольвента 3 и 13.

S_4 *21. $\sim T(a) \vee \sim f(f(a)=f(f(b))$, } парамодулянты 6 и 19
*22. $\sim T(a) \vee \sim f(f(b)=f(f(a))$, }

S_5: 23. $a=b$, резольвента 4 и 14.
24. $Q(b)$, резольвента 4 и 15.
25. $S(b)$, резольвента 4 и 16.
26. $T(b)$, резольвента 4 и 17.
27. $f(b) \neq f(b)$, резольвента 4 и 18.
28. $f(a)=f(b)$, резольвента 4 и 19.
29. $f(b)=t(a)$, резольвента 4 и 20.

30. $f(f(a))=f(f(b))$, резольвента 4 и 21.
31. $f(f(b))=f(f(a))$, резольвента 4 и 22.

S_6: 32. $f(f(f(a)))=f(f(f(b)))$, 33. $f(f(f(b)))=f(f(f(a)))$, } парамодулянты 6

S_7: 34. *NIL*, резольвента 5 и 28.

6.4.3. Линейная парамодуляция

Для данного множества S дизъюнктов и дизъюнкта C_1 из S *линейным выводом* C_n *с резолюцией и парамодуляцией* с начальным дизъюнктом C_1 называется последовательность дизъюнктов $C_1, . ., C_n$, в которой
1) для i=l, .. ., n—1 C_{i+1} является резольвентой или парамодулянтом дизъюнктов C_i, и B_i;
2) каждый B_i является или некоторым дизъюнктом из S, или некоторым C_j, $j<i$.
Линейным опровержением с резолюцией (*директивой*) *и парамодуляцией* называется линейный вывод с резолюцией (директивой) и парамодуляцией пустого дизъюнкта.

Пример 6.16. Пусть S есть $\{\sim R(c) \vee c=d,\ \sim R(c) \vee g(c)\neq g(d),\ R(c) \vee a=b,\ R(c) \vee g(a)\neq g(b),\ g(x)=g(x)\}$. Тогда линейное опровержение из S с начальным дизъюнктом $\sim R(c) \vee c=d$ представлено на рис. 6.7.

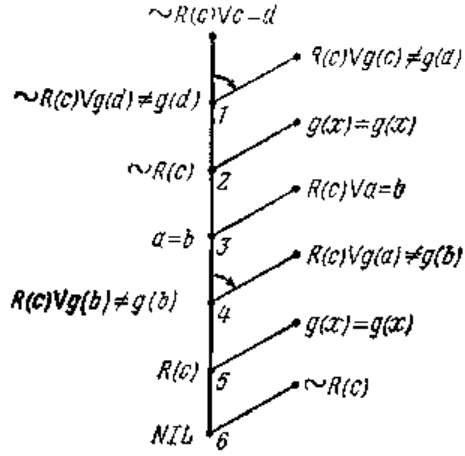


Рис. 6.7. Линейное опровержение с резолюцией (директивой) и парамодуляцией.

Здесь шесть боковых дизъюнктов. Пять из них из множества S, а один $(\sim R(c))$ является предшествующим.

Можно показать, что линейная парамодуляция полна, т. е. справедлива следующая теорема.

Теорема 6.9. *Если C есть дизъюнкт в R-невыполнимом множестве S дизъюнктов, включающем x=x и функционально-рефлексивные аксиомы, и если S —{C} является R-выполнимым, то S имеет линейное опровержение с резолюцией* (*директивой*) *и парамодуляцией с начальным дизъюнктом C.*

6.5. Стратегии поиска

В 6.1 указано на то, что **процедура доказательства предписания (директивы) может быть разделена на правило вывода и стратегию поиска.**

В предыдущих параграфах описаны правила вывода, используемые для поиска опровержения. В данном параграфе будут рассмотрены стратегии поиска. Задача стратегии поиска — выбрать те дизъюнкты-кандидаты, к которым на очередном шаге процедуры доказательства **предписания (директивы)** следует применять правило вывода.

В 2.5 было показано, что пространство поиска при доказательстве предписания (директивы) может быть представлено в виде абстрактного графа доказательства предписания (директивы) (G, s), в котором различным выводам одной и той же формулы соответствуют различные вершины. Если граф доказательства предписания (директивы), изображенный на рис. 2.9, интерпретировать как граф, получаемый применением принципа резолюции (директивы), используя метящую функцию $c: G \rightarrow B^*$, то два дизъюнкта $c(n_7)$ и $c(n_8)$ должны представлять собой все резольвенты пары $c(n_3)$ и $c(n_4)$. Дизъюнкт $c(n_8)$ не должен резольвировать ни с одним из дизъюнктов $c(n_i)$, где $1 \le i \le 14$. Дизъюнкты $c(n_{10})$ и $c(n_{11})$ являются или факторами дизъюнкта $c(n_7)$, или получены из $c(n_7)$ резольвированием $c(n_7)$ с самим собой. Если, например, $C=c(n_6)=c(n_7)=c(n_{14})$, то C имеет три вывода: два первого уровня и один третьего уровня. Отметим, что выводы нет необходимости представлять в виде дерева. В приведенном примере дизъюнкт $c(n_2)$ используется дважды в выводе $c(n_{13})$, но представлен в графе в виде одной вершины.

Как было указано в 2.5, абстрактная задача доказательства предписания (директивы) может быть представлена в виде четверки $P=(G, s, F, g)$, где $F \subseteq G$ является множеством терминальных вершин для P (или решающих вершин) и $g: G \rightarrow R$ — оценочная функция,

выражающая меру сложности вывода, (*R* — множество вещественных чисел).

Решение задачи *P* состоит в разработке стратегии поиска ∑, которая генерирует из G_0 вершину $n \in F$.

Стратегия поиска ∑ для *P* есть функция ∑: $2^G \rightarrow 2^G$, которая вырабатывает подмножество *G* из другого подмножества *G*. Задача стратегии поиска в данной постановке состоит в выборе наиболее перспективных дизъюнктов для применения к ним правил вывода. Другими словами, **задача состоит в упорядочении дизъюнктов на основании некоторой меры, называемой обычно оценочной функцией.** При этом дизъюнкты, имеющие наименьшее значение оценочной функции, считаются предпочтительными и именно к ним и первую очередь применяются правила вывода.

Наибольшее распространение на практике получил специальный вид оценочной функции $f(n)=g(n)+h(n)$ (для всех $n \in G$), выраженной в терминах меры сложности *K* и дополнительной эвристической функции *h* (см. 5.2). Эвристическая функция *h* для *P* есть функция $h:G \rightarrow R$ такая, что $h(n) \geq 0$, для всех $n \in G$. Интерпретация эвристической функции *h* состоит в том, что $f(n)=g(n)+h(n)$ оценивает стоимость *g* (*n**), где *n**—терминальная вершина $n^* \in F$ такая, что $n \leq n^*$, т. е. *h(n)* оценивает $g(n^*)—g(n)$.

В терминах этой оценочной функции можно описать ранние поисковые стратегии. Обычно в них *g(n)* характеризовало уровень в графе доказательства, а *h(n)* — длину дизъюнктов. Выбор в качестве *h(n)* длины дизъюнкта обосновывается тем фактом, что резольвирование дизъюнктов, содержащих одну литеру, сразу приводит к опровержению. Стратегии *насыщения уровня* могут быть охарактеризованы как стратегии, использующие упорядочение на основании оценочной функции $f(n)=g(n)$, т. е. в первую очередь получаются все резольвенты исходного множества дизъюнктов (нулевого уровня), затем первого уровня, второго и т. д. Эта стратегия эквивалентна стратегии поиска в ширину.

Стратегия *предпочтения единичным дизъюнктам* (одночленам) может быть представлена оценочной функцией $f(n)=h(n)$ при условии, что $g(n)<g_{max}$. Данная стратегия широко используется на практике. Однако эта стратегия является неэффективной при наличии в исходном множестве аксиом большого количества единичных дизъюнктов (что имеет место в вопросно-ответных системах) .

Стратегия *предпочтения наименьшим литерам,* предложенная Слэйглом, также выражается функцией $f(n)=h(n)$. Отличие ее от стратегии предпочтения единичным дизъюнктам состоит только в способе вычисления *h(n)*. Здесь *h(n)* вычисляется не для дизъюнкта-

кандидата, а для пары дизъюнктов-кандидатов. Предпочтительной считается такая пара дизъюнктов (*A, B),* которая образует резольвенту *(R)* наименьшей длины. Длина резольвенты может быть вычислена до образования резольвенты по следующей формуле:

$$h(R) \leq h(A) + h(B) - 2.$$

На практике, так как важно не абсолютное значение длины, а относительное, $h(R)$ считают по формуле $h(R) = h(A) + h(B)$.

Мы привели выше простейшие стратегии поиска, используемые в программах доказательства предписаний (директив). Ковальский обратил внимание на необходимость разработки более сложных стратегий и определил класс диагональных поисковых стратегий (ДПС) (см. 5.2) и восходящих ДПС (см. 5.3).

Рассмотрим предложенный Ковальским частный алгоритм *восходящей диагональной стратегии поиска,* названной им ∑*-алгоритмом.

Пусть m будет вершина в пространстве поиска, а $c(n)$— соответствующий ей дизъюнкт. В ∑*-алгоритме оценочная функция определяется парой (i, j), где i — длина дизъюнкта $c(n)$, а j — уровень дизъюнкта $c(n)$. Если даны дизъюнкты $c(n_1)$ с оценкой (i_1, j_1) и дизъюнкт $c(n_2)$ с (i_2, j_2), то $c(n_1) < c(n_2)$ (т. е. $c(n_1)$ *лучше* $c(n_2)$), если

а) $i_1 + j_1 < i_2 + j_2$ или б) $i_1 + j_1 = i_2 + j_2$ и $i_1 < i_2$.

Если $i_1 + j_1 = i_2 + j_2$ и $i_1 = i_2$, то $n_1 =_{d^u} n_2$ (т. е. n_1 и n_2 равны в смысле введенного порядка).

Таким образом, при равенстве суммы длины и уровня двух дизъюнктов предпочтение отдается более короткому дизъюнкту, исходя из того, что цель стратегии поиска— получить дизъюнкт нулевой длины.

На основе введенного упорядочения Ковальский разделил пространство поиска на множества $A(i, j)$. Каждое множество $A(i, j)$ состоит из дизъюнктов, имеющих одинаковую меру упорядочения $<_{d^u}$. Алгоритм ∑* пытается генерировать дизъюнкты в соответствии с упорядочением восходящей ДПС. Сначала ∑* пытается найти среди исходных дизъюнктов дизъюнкт для множества A (0, 0), что возможно, только если в исходном множестве S есть пустой дизъюнкт. Затем делается попытка последовательно генерировать (путем выбора из исходного множества или применением правил вывода) дизъюнкты в множества A (0, 1), A (1,0), A (0, 2), A (1, 1), A (2,0), A (0,3), . . ., $A(i, j)$, где i - длина дизъюнкта, а j — уровень. ∑* генерирует дизъюнкты для множества $A(i, j)$ одним из следующих способов:

1. При $j = 0$ дизъюнкт исходного множества длиной i заносятся в $A(i, j)$.
2. При $j > 0$ фактор (факторы) дизъюнктов из $A(i+1, j-1)$ заносятся в $A(i, j)$.

3. При $j>0$ в множество A (i, j) заносится резольвента дизъюнкта $c(n_1)$ из A (i_1, j_1) и дизъюнкта $c(n_2)$ из $A(i_2, j_2)$, где $i=i_1+i_2-2$ и $j=\max(j_1, j_2)+1$. В начале доказательства все множества A пусты. Они заполняются программой НАПОЛНИТЬ (i, j), которая генерирует первым или третьим из указанных способов дизъюнкт с оценкой (i, j). Алгоритм $\sum^*$ начинает работу с вызова НАПОЛНИТЬ (0, 0). Дизъюнкты $c(n_1)$ и $c(n_2)$, образующие резольвенту третьим способом, являются дизъюнктами из множества A (i_1, j_1) с лучшей мерой, чем (i, j). Когда программа НАПОЛНИТЬ (i, j) образовала все возможные дизъюнкты, то вызывается программа НАПОЛНИТЬ (i',j'), где (i',j') есть следующая мера упорядочения в смысле $>_{d^u}$. Всякий раз, когда НАПОЛНИТЬ (i, j) генерирует новый дизъюнкт $c(n)$, вызывается программа РЕКУРСИЯ $(c(n))$. Ее задача — проверить, взаимодействует ли $c(n)$ с ранее генерированными дизъюнктами, и в этом случае выработать способом 2 и 3 все дизъюнкты с оценкой $(i', j')<_{d^u}(i, j)$, которые являются потомками $c(n)$. К полученным дизъюнктам также применяется программа РЕКУРСИЯ. Этот рекурсивный процесс продолжается до тех пор, пока на некотором уровне не удается образовать дизъюнкты с мерой, лучшей, чем (i, j). Тогда управление возвращается предыдущему уровню программы РЕКУРСИЯ. В конце концов осуществляется повторный вход в программу НАПОЛНИТЬ, и указанный процесс продолжается до нахождения пустого дизъюнкта или до исчерпания времени (или памяти), отведенного на доказательство.

Ковальский определил восходящую ДПС для оценочной функции $f(n)=g(n)+h(n)$. Можно усложнить данную функцию и рассматривать ее как линейную комбинацию не двух, а большего количества функций. Например,

$$f(n)=W_o g(n)+\omega_1 h_1(n)+\omega_2 h_2(n),$$

где ω_i есть весовой коэффициент $\left(\sum_i \omega_i = 1\right)$, g — уровень дизъюнкта, h_1 — длина дизъюнкта, h_2 — мера функциональной сложности дизъюнкта (например, наибольший уровень вложенности функций, входящих в дизъюнкт).

В изложенных выше стратегиях поиск начинается со всего множества исходных аксиом G_o (формул нулевого уровня). При решении многих практических задач множество G_o оказывается очень большим, что приводит к невозможности найти опровержение в приемлемое время. Применительно к рассматриваемой нами проблеме формирования и доказательства предписаний (директив) G_0 включает множество всех

исходных знаний. Будем называть все факты, входящие в G_0, *базовыми данными*. Аксиомы базовых данных, не содержащие переменных, будем называть *константными аксиомами*. Остальные аксиомы будем называть *общими аксиомами*. Необходимо отметить, что для решения любой конкретной задачи формирования и доказательства предписаний (директив) требуются не все базовые данные, а только небольшая их часть. Поэтому необходимо при поиске доказательства выбирать из базовых данных только те дизъюнкты, которые уместны для решаемой задачи. Выбор уместных аксиом может быть осуществлен в основном за счет использования семантической информации о дизъюнктах, а не синтаксической, использованной в методах упорядочения (длина, уровень или мера сложности дизъюнктов).

Продемонстрируем методы внесения в процедуру поиска не только синтаксической, но и семантической информации на примере *Q** алгоритма. Данный алгоритм состоит из двух подалгоритмов:

1. *Алгоритма выбора базового дизъюнкта,* осуществляющего выбор из множества всех дизъюнктов тех, которые относятся к решаемой задаче.
2. *Алгоритма дедукции,* определяющего последовательность, в которой к дизъюнктам применяются правила вывода .

Оба алгоритма удобно описать в контексте поиска *в системе редукций.* Система редукций состоит из трех множеств: начальных задач, множества операторов и конечных (разрешимых) задач.

В контексте доказательства предписаний (директив) методом резолюции будем дизъюнкт рассматривать как задачу, а литеру как подзадачу. Дизъюнкт при этом соответствует конъюнкции подзадач, так как для решения задачи все литеры должны быть разрешены путем применения операторов (других дизъюнктов).

Состав множества начальных задач зависит от используемой системы вывода. Например, если используется стратегия множества поддержки или стратегия, включающая множество поддержки (например, *OL*-вывод), то множество начальных задач состоит из дизъюнктов, входящих в отрицание доказываемого предписания (директивы).

Множеством операторов является множество дизъюнктов, применимых к задаче посредством правил вывода. Однако множество применимых операторов зависит от используемых правил вывода. Резолюция (директива) использует два дизъюнкта, и для многих систем вывода любой из этих дизъюнктов можно рассматривать как задачу, а другой как оператор. Однако для ряда систем вывода кажется естественным делать различие между дизъюнктами. Например, в случае линейной или *OL*-резолюции (директивы) боковые дизъюнкты

естественно рассматривать как операторы. Конечными задачами являются дизъюнкты определенного вида. Типичным примером является пустой дизъюнкт, сообщающий, что решена вся задача. Более сложным является определение окончания подзадачи.

Подзадача *A*, соответствующая литере в дизъюнкте, может быть решена одним из двух способов:

1) разрешена за один шаг применением опровергающего ее оператора (например, путем резольвирования с одночленным дизъюнктом, дополнительным к *A*);

2) оператор, примененный к *A,* порождает дополнительное множество подзадач; при этом *A* считается разрешимым тогда и только тогда, когда будут разрешены все подзадачи.

Определить на практике, когда данная подзадача разрешена, весьма трудно, так как в зависимости от системы вывода подзадачи рассматриваются в произвольном порядке. Решение указанной задачи в общем случае требует запоминания громоздкой информации. Однако в случае *OL*-резолюции (директивы) проблема упрощается, так как на данном уровне испытывается только одна подзадача, а информация, требуемая для запоминания, хранится в представлении дизъюнкта. Если решение некоторой подзадачи приводит к другим подзадачам, то в *OL*-резолюции (директиве) создается *A*-литера, а вновь образованным подзадачам соответствуют *B*-литеры, расположенные справа от данной *A*-литеры. Когда все *B*-литеры будут разрешены (*OL*-резолюция (директива) выполняется над самой правой *B*-литерой) и *A*-литера оказывается справа, то *A*-литера устраняется из дизъюнкта операцией сокращения (см. п. 6.3.4), и только в этот момент выбирается для решения новая подзадача.

Процесс дедукции, используемый в алгоритме *Q**, подобен процессу поиска, описанному для алгоритма $\sum^*$, и состоит в вычислении оценочной функции. Отличие состоит в том, что с целью ускорения взаимодействия уместных базовых дизъюнктов с вновь полученными дизъюнктами можно вводить базовые дизъюнкты не только в программе НАПОЛНИТЬ, но и в программе РЕКУРСИЯ. В алгоритме широко используются различные *правила вычеркивания,* направленные на устранение избыточных или семантически бессмысленных дизъюнктов. Примерами таких дизъюнктов являются *тавтологии, алфавитные варианты* уже существующих дизъюнктов или *поглощаемые дизъюнкты*.

Алфавитным вариантом некоторого дизъюнкта *C* называется дизъюнкт *C'*, получаемый из *C* подстановкой, заменяющей только названия переменных.

Дизъюнкт C_1 *поглощает* дизъюнкт C_2, если существует такая подстановка, что $C_1\sigma \subseteq C_2$. Например, дизъюнкт $P(x)$ поглощает дизъюнкт $P(y) \vee Q(z)$, так как при $\sigma=\{y/x\}$

$$\{P(x)\}_\sigma \subseteq \{P(y) \vee Q(z)\}.$$

Тавтологии, алфавитные варианты и поглощаемые дизъюнкты являются неуместными и излишними, и они должны вычеркиваться, когда для этого предоставляется возможность. Вычеркивание алфавитных вариантов и тавтологий не нарушает полноты правил вывода. Вычеркивание поглощаемых дизъюнктов нарушает полноту некоторых правил вывода. Так, например, для бинарной резолюции (директивы) (или семантической резолюции (директивы)) и стратегии насыщения уровня вычеркивание поглощаемых дизъюнктов не нарушает полноты правил вывода, только если оно осуществляется после насыщения уровня.

Для устранения избыточных дизъюнктов применяется способ *означивания предикатов,* осуществляемый путем ссылок на семантическую информацию конкретной проблемной коммуникационной области. Так, например, пусть мы имеем дизъюнкт $C \vee \sim F(Мэри, x)$, где C — дизъюнкт. Если нам известно на основании семантической информации, что для истинности предиката F первый его аргумент должен быть объемом мужского пола, то мы можем определить, что $\sim F(Мэри, x)$ в данной интерпретации принимает значение «истино» и, следовательно, весь дизъюнкт можно отбросить, не нарушая свойств невыполнимости оставшегося множества. Если же некоторая литера в результате означивания приобретает значение «ложно», то она может быть устранена из того дизъюнкта, в который она входит.

Хотя означивание предикатов является полезной стратегией, но она не защищает от генерации неуместных дизъюнктов. Указанная задача может быть с успехом решена путем тщательного выбора уместных базовых дизъюнктов.

Для того чтобы выбрать аксиомы (операторы, директивы), уместные для данной задачи, алгоритм Q^* первоначально генерирует дизъюнкты из отрицания предписания. В зависимости от системы вывода алгоритм рассматривает одну или все литеры итерированных дизъюнктов (дизъюнктов «хозяев») как спецификацию, на основании которой осуществляется выбор аксиом. Каждую из этих литер будем называть *выделенной литерой*. Выделенная литера используется для выбора аксиом, которые *резольвируют* по данной литере с генерированным дизъюнктом. Таким образом, выделенные литеры действуют как образцы. Этот прием подобен идее Грина, на основании которой, только определенные дизъюнкты (активные) принимают участие в

процессе поиска доказательства. Количество аксиом, отобранных выделенными литерами, может быть уменьшено отфильтровыванием на основании семантики (например, несоответствие типов аргументов). Оставшиеся после фильтрации операторы (директивы) заносятся в список СПЕЦ и упорядочиваются таким образом, чтобы более уместные испытывались в первую очередь.

В результате работы алгоритма выбора базового дизъюнкта дедуктивный алгоритм получит не все базовые дизъюнкты, имеющие данную меру упорядочения. Как следствие этого, выбранные базовые дизъюнкты не обязательно будут генерироваться в соответствии с мерой упорядочения. Следовательно, в отличие от алгоритма ∑*, алгоритм *Q** не обладает допустимостью (см. р. 5). Однако потеря допустимости имеет только теоретическое значение. На практике кажется более важным найти быстро какое-нибудь решение, а не обязательно простейшее решение, требующее больших затрат и влекущее за собой риск не решить задачу в отведенные время и объем памяти.

Опишем теперь более подробно, каким образом происходит **выделение аксиом, их фильтрация и упорядочение.**

Как уже указывалось ранее, литеры генерированных дизъюнктов используются для выделения аксиом, которые становятся кандидатами для генерации. Система вывода, используемая алгоритмом дедукции, будет определять, какие дизъюнкты будут использоваться для этих целей и какие критерии будут применяться для выбора аксиом. Если система вывода использует стратегию опорного множества или если система связана с опорным множеством, как в *OL*-резолюции (директиве), то используются только литеры дизъюнктов, обладающих поддержкой. С другой стороны, если система вывода не включает стратегию опорного множества, то все сгенерированные дизъюнкты и общие аксиомы будут использоваться для выделения аксиом. При этом критерий резолюции (директивы) должен быть ослаблен таким образом, что аксиома будет становиться кандидатом, если она содержит литеры, которые *унифицируют* с литерами сгенерированного дизъюнкта. Ослабление критерия резолюции (директивы) необходимо для того, чтобы обеспечить опровергающую полноту алгоритма, например, в случае, когда для опровержения теоремы необходимо доказательство леммы, а предписание не используется до тех пор, пока лемма не доказана.

Система вывода указывает также, какие литеры должны использоваться при выделении аксиом. Так, например, такая система вывода, как *OL*-резолюция (директивы), указывает одну литеру

каждого генерированного дизъюнкта, которая используется. Другие системы вывода используют для этих целей все литеры.

Рассмотрим теперь проблему семантической фильтрации найденных аксиом-кандидатов. Фильтрация может делаться на основе типов переменных и констант, входящих и выделенную литеру. Например, предположим, что литера генерированного дизъюнкта имеет вид *~ РОДИТЕЛЬ (x, Петр)* и из контекста известно, что переменная *x* имеет тип *МУЖЧИНА*. Предположим, что потенциальными кандидатами, найденными в базовых данных, являются дизъюнкты:

1) *~ОТЕЦ(u, v) ∨ РОДИТЕЛЬ (u, v);*

2) МАТЬ (u, v) ∨ РОДИТЕЛЬ (u, v).

Применяя фильтр, получим, что аксиома 2) является несовместимой с выделенной литерой, так как переменная *u* в литере *РОДИТЕЛЬ* имеет из контекста тип *ЖЕНЩИНА* (так как является первым аргументом предиката *МАТЬ,* требующего типа *ЖЕНЩИНА)*. Таким образом, кандидатом становится только дизъюнкт 1).

Если некоторая подзадача может быть полностью решена с использованием константных аксиом, запомненных в явном виде в базовых данных, то нет необходимости генерировать для решения подзадачи общие аксиомы. Например, пусть *МАТЬ (x, Эмиль)* является литерой сгенерированного дизъюнкта. Пусть системе известен факт, что у каждого человека есть точно одна мать. Если аксиома *МАТЬ (Роза, Эмиль)* найдена в базовых данных системы, то она полностью решает подзадачу и мы выбираем только одного кандидата. Никакие другие аксиомы, которые могут резольвировать с предикатом *МАТЬ (x, Эмиль),* не следует вводить в пространство поиска для решения этой подзадачи.

Описанный процесс фильтрации уменьшает число аксиом, которые принимают участие в процессе поиска. Так как меньшее количество аксиом будет участвовать в логическом взаимодействии, то уменьшится число генерируемых дизъюнктов.

Следует отметить, что, хотя фильтрация делает систему неполной в смысле исчерпания всех вариантов решений, она оставляет ее полной в смысле опровержения.

После того как аксиомы-кандидаты выделены и подвергнуты семантической фильтрации, их необходимо упорядочить так, чтобы «более перспективные» обрабатывались в первую очередь.

Упорядочение выполняется в два этапа:

1) упорядочение кандидатов, связанных с выделенной литерой (это соответствует упорядочению операторов, применяемых к подзадаче);

2) упорядочение списков кандидатов, соответствующих различным выделенным литерам (это соответствует упорядочению подзадач).

Упорядочение кандидатов выделенной литеры осуществляется двумя способами. Либо упорядочение совершается на основании рекомендаций пользователя аналогично списку рекомендаций, нпример, в языке PLANNER, либо в соответствии с оценочной функцией, используемой в дедуктивном алгоритме. В последнем случае константные аксиомы предшествуют общим аксиомам

Для упорядочения подзадач (литер) используется эвристика, состоящая в предпочтении литерам, имеющим одну пли несколько констант. Смысл этой эвристики состоит в том, что

1) константные литеры будут резольвировать (унифицировать) с меньшим числом базовых дизъюнктов, чем общие литеры,

2) если предписпние касается частного утверждения, то предлагаемый механизм будет более уместен, чем слепой перебор на основе общих литер.

На рис. 6.8 приведен пример доказательства с предпочтением литерам, содержащим константы.

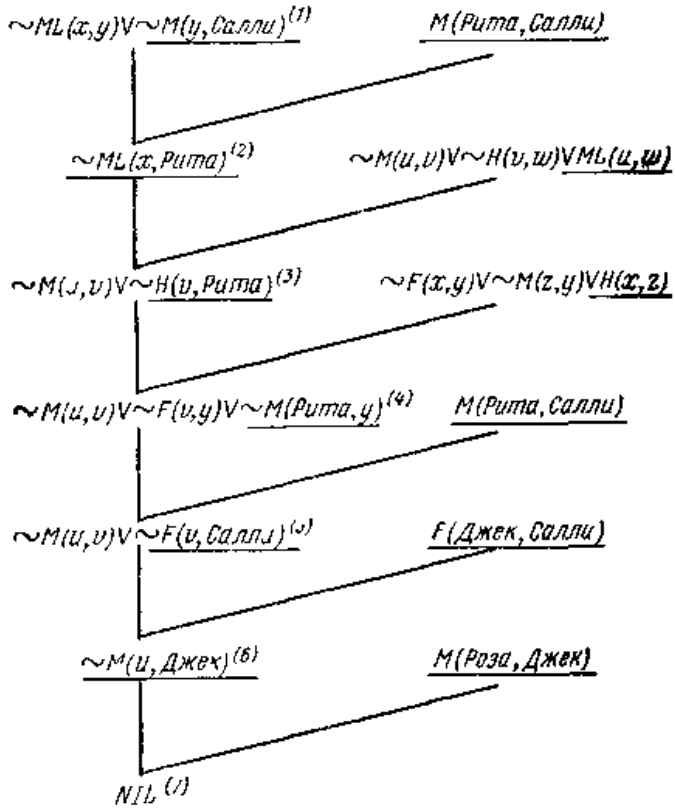


Рис. 6.8 Использование констант для ведения поиска.

Дизъюнкт 1) имеет две выделенные литеры. Мы отдаем предпочтение литере *~M(y, Салли),* так как она содержит константу. Аксиомы, которые могут взаимодействовать с литерой *~M (y, Салли),* принимают участие в поиске до других аксиом, применяемых к дизъюнкту 1). По аналогичным причинам мы отдаем предпочтение литере *~H (v, Puma)* в дизъюнкте 3).

Если существует несколько подзадач, содержащих константы, то упорядочение осуществляется на основании предсказанного значения оценочной функции резольвенты, полученной из дизъюнкта «хозяина» выделенной литеры и первой аксиомы в списке кандидатов этой литеры.

Следует отметить, что в некоторых базовых данных определенные константы встречаются очень часто. Примером может служить константа, отражающая пол (мужчина, женщина) в картотеке личного состава какого-либо учреждения. Так как такие константы часто встречаются в базе данных и не именуют конкретного индивидуума, то они мало будут помогать в ведении поиска. Поэтому эти типы констант при упорядочении целесообразно рассматривать как переменные. Определенные события, происходящие в процессе поиска, могут быть использованы для сокращения пространства поиска и устранения определенных кандидатов из списка литер. Мы упоминали при описании фильтрации кандидатов-аксиом, что некоторые подзадачи могут быть полностью разрешены с помощью константных базовых аксиом. В более общей форме можно сказать, что некоторые подзадачи имеют точное число решений, в то время как другие — неопределенное число. Если число решений для данной подзадачи известно, то, когда все решения найдены, часть графа поиска (сформированная для решения данной подзадачи) может быть отброшена. В дополнение к этому аксиомы-кандидаты, выделенные при решении этой подзадачи, могут быть устранены. Результатом такого сокращения является уменьшение числа неуместных дизъюнктов в пространстве поиска и, что даже более важно, устранение их преемников.

Выполнение этого семантического действия весьма сложно для произвольной системы вывода, так как для данного дизъюнкта (задачи) попытки решения всех подзадач выполняются одновременно. Однако для *OL*-резолюции (директивы) этот подход применим, так как здесь в текущий момент рассматривается только одна подзадача и информация, требуемая для обнаружения решения подзадачи, встроена в представление *OL*-резолюции (директивы).

Итак, мы описали приемы по использованию семантики в процессе поиска доказательства предписаний (директив). Некоторые из этих приемов являются довольно общими и могут применяться для многих коммуникационных областей.

Заканчивая описание методов доказательства предписаний (директив), отметим, что основными способами устранения причин «экспоненциального взрыва» имеющего место при доказательстве предписаний (директив) практической степени сложности, является использование семантики и встраивание в правила вывода и алгоритмы унификации специфики конкретной области применения.

6.6. О машинном доказательстве предписаний (директив)

Современные компьютеры могут выполнять множество сложнейших операций, доступных ранее только человеку. Кроме таких привычных действий, как распознавание образов, перевод текстов с одного языка на другой, они все больше внедряются в самые сложные сферы искусственного интеллекта – научного направления, моделирующего процесс человеческого мышления. Одно из таких направлений – автоматическое доказательство предписаний (директив) – мы собираемся рассмотреть ниже.

Практически каждый из нас имеет представление о том, что такое аксиомы, определения и как проводятся математические доказательства. Подобную работу могут выполнять компьютеры (вернее, компьютерные программы), что, кажется, должно свидетельствовать о том, что они способны мыслить – в нашем, человеческом, понимании. На самом деле это, конечно же, не так. Они лишь слепо используют определенный набор правил и приемов, которому их нужно предварительно научить. Однако недостаток творческого начала в некоторых случаях вполне может компенсироваться быстродействием, и хорошее тому подтверждение – успехи шахматных программ. А исходной точкой для «машинного интеллекта» всегда будет некоторая формализация наших знаний, каковой является, в частности, математическая логика – дисциплина, посвященная изучению доказательств. Впрочем, от читателей не требуется глубоких познаний в этой области, но тем, кто совсем не знаком с понятиями «предикат», «логическое следствие», «формальный вывод», возможно, не лишне будет предварительно прочесть соответствующую врезку – чтобы лучше понять, каким же образом действует компьютер, выполняя доказательство.

Автоматическое доказательство и логическое программирование

Установление отношений логического следствия, или, что то же самое, доказательство предписаний (директив) является одной из главных задач логики. Она представляет не только теоретический, но и практический интерес для многих научных и технических областей. В качестве примера назовем две достаточно широкие научно-прикладные сферы, проблемы из которых могут быть выражены в терминах доказательства предписаний (директив):

- анализ программ. Если процесс выполнения программы описывается некоторой логической формулой A, а условие завершения – формулой B, то проверку того, что программа закончит работу, можно сформулировать как доказательство следования B из A;
- преобразование состояний. Если заданы набор состояний и множество операторов над ними, то проблема достижения некоторого заданного состояния из исходного опять же сводится к доказательству логического следствия.

В логике было предложено немало методов установления логического следствия. Большинство из них основаны на доказательстве того, что некоторая логическая формула, связанная с предписание (директивой), истинна или ложна. Однако данные методы требуют большой рутинной работы (переписывания, перебора вариантов) для своей реализации, т. е. слишком трудоемки и громоздки для «ручного» применения. В то же время они легко программируются на ЭВМ, и это делает задачу установления логического следствия гораздо более доступной. Приемы и алгоритмы, используемые для решения таких задач, будем называть автоматическими (машинными) методами доказательства предписаний (директив).

Хотя многие алгоритмы были предложены давно и постоянно совершенствуются, их реализация на традиционных языках программирования вызывает существенные трудности. Потребность в адекватных инструментальных средствах привела к созданию целого направления в программировании, называемого логическим, которое включает подходы и методы, приспособленные для решения задач логики, и в рамках которого созданы специализированные языки.

При решении задач на этих логических языках, как правило, описывают предметную область, например, коммуникационные процессы, (называемую иногда универсумом рассмотрения), и на ней определяют функции и предикаты, выражающие взаимосвязи между исследуемыми сущностями. Логическое программирование при этом обладает особыми свойствами, отличающими его от других подходов. В функциональном программировании процедура действует как математическая функция, возвращая значение от аргументов. Логическая же процедура соответствует математическому понятию отношения и обладает большей гибкостью. При определении отношений ничего не говорит о том, какие аргументы исходные и

каковы возможные результаты – в разных запросах одни и те же объекты могут играть различные роли.

Само исчисление, реализованное в логическом программировании, состоит из **формализованного языка и механизма вывода.** Последний, в свою очередь, включает некоторое множество исходных предложений, называемых аксиомами, и правил вывода, позволяющих из одних предложений получать другие. Последовательность предложений, каждое из которых либо является аксиомой, либо следствием предыдущих, называется формальным выводом или доказательством, а предложения, получаемые в этом процессе, – предписаниями.

Наличие формальных правил вывода позволяет проверять правильность доказательств предписаний, порождая с их помощью из аксиом все новые и новые следствия – до получения интересующего нас предписания. Не менее часто применяется и обратный метод: берут интересующее предписание и выясняют, из каких посылок оно может быть получено, затем анализируют эти посылки и т. д., пока не придут к аксиомам. К сожалению, в обоих случаях крайне сложно оценить необходимый объем вычислений.

Программные пакеты, выполняющие автоматические доказательства или некоторые их элементы, традиционно разделяют на *интерактивные* и *автоматические пруверы* (системы доказательств, interactive/automated theorem provers) и *верификаторы моделей* (model checkers). Данная классификация основана на некоторых параметрах решаемых задач и степени участия пользователя, хотя в целом такое разделение в известной степени условно.

Надо отметить, что с развитием машинных методов стал актуальным вопрос о том, что же считать доказательством в математике. Традиционно под ним понималась обозримая последовательность выводов, которую можно проверить «вручную», условно говоря, с помощью карандаша и листа бумаги. Однако участие в процессе компьютера потребовало пересмотра такого «устаревшего» представления.

Для иллюстрации сказанного приведем пример фундаментальной математической проблемы, решенной (в основном) машинными методами. Это так называемая задача о четырех красках, не

поддававшаяся аналитическому решению более века. Она была сформулирована в 1852 г., когда один английский географ заметил, что для раскраски карты Британии (при которой соседние графства имели бы разный цвет) достаточно четырех красок. Математики, заинтересовавшиеся этим любопытным фактом, задались вопросом: а достаточно ли четырех красок для раскрашивания произвольной карты? Оказывается, да.

Упуская занимательную предысторию, скажем, что первое правильное доказательство этой теоремы было получено только в 1976 г. с использованием машинных методов. Между тем оно до сих пор вызывает сомнение у ряда специалистов. Дело в том, что в нем одновременно с громоздкой аналитической частью доказательства, состоящего из десятков страниц сложнейших выкладок и тысяч (!) дополнительных диаграмм, имеется машинная часть, которую удалось проверить только на компьютере (использовался суперкомпьютер Cray-1A). Авторы доказательства логически свели задачу к исследованию 1482 базовых карт, сформированных специальным образом – их раскраска на Cray потребовала 50 суток компьютерного времени. Это-то и поставило под сомнение надежность данного метода – вероятность ошибки (как программной, так и аппаратной) при таком объеме не является пренебрежимо малой. Впрочем, доказательство 1976 г. было подтверждено позже. В 1990-х годах задача о четырех красках была сведена к раскрашиванию «всего» 633 базовых карт, что может быть реализовано на современном ПК за вполне приемлемое время – всего за несколько часов. Тем не менее поводы для скепсиса не исчезли – ведь задача опять решена машинными методами, и о проверке доказательства «вручную» можно даже не мечтать.

Coq и Gallina

Важной особенностью большинства приложений, реализующих методы автоматического доказательства, является их некоммерческий, экспериментальный характер. С одной стороны, это обеспечивает бесплатный и неограниченный доступ к данному ПО всех желающих, но с другой – подобные системы, как и вообще многие некоммерческие IT-проекты, зачастую не имеют развитой пользовательской поддержки, нередко отличаются нестабильным функционированием и т. д.

Выше мы кратко упомянули об основных классах логических программ. К сожалению, даже тезисный обзор лучших представителей

каждого из них, который смог бы сориентировать читателя, скажем, в выборе подходящего ему пакета, является совершенно необозримой задачей. Поэтому в качестве примера мы детальнее остановимся лишь на одном из продуктов, обладающим, несмотря на общедоступность, весьма развитыми инструментами для решения сложных задач.

Система автоматизированного построения доказательств Coq, созданная в исследовательском институте INRIA (Франция), является достаточно сбалансированным приложением, сочетающим поддержку широкого спектра логических процедур, высокую стабильность работы и относительную простоту освоения для широких кругов пользователей. Система и сопровождающая ее документация могут быть свободно загружены с узла продукта. Разработчики определяют Coq как Proof Assistant, что в приведенной классификации соответствует интерактивным пруверам. Кроме того, Coq поддерживает процедуры полностью машинного вывода, характерные для автоматических пруверов, т. е. система обладает качествами сразу двух выделенных нами классов программ. Coq реализован на нескольких платформах (Linux, Mac OS, Windows, Solaris) и имеет набор интерфейсов, облегчающих взаимодействие с пользователем. В ОС Windows основным средством работы с системой является стандартная командная консоль, хотя в поставку входит и графический интерфейс CoqIde.

Интерфейс интегрированной среды разработки системы автоматизированного построения доказательств Coq

Базируется Coq на собственном языке спецификаций Gallina, состоящем из **объявлений, определений и команд**. Объявление ставит в соответствие некоторому имени его спецификацию, что примерно отвечает созданию переменной определенного типа в алгоритмических языках. **Спецификации делятся на логические высказывания, математические коллекции и абстрактные типы**. Например, спецификация nat является аксиоматическим понятием, принадлежащим математической коллекции, и означает натуральное число. Команда Variable n:nat. объявляет объект n как переменную натурального типа.

С помощью определений создаются и именуются новые объекты (в то время как объявления приписывают объектам только типы). Например, в арифметическом модуле Coq определен объект plus, ставящий в соответствие двум натуральным числам их сумму (очевидно, что он является прямым аналогом функционального символа «сумма» из исчисления предикатов).

Как же проводятся доказательства в Coq? Кратко поясним только схему, не вдаваясь в детали. **Определение (в терминах Gallina называемое целью), которое должно быть доказано, записывается с помощью соответствующих операторов**. Заранее, если необходимо, вводятся используемые в нем объекты – переменные, функциональные символы и др. Затем к цели применяются специальные команды, называемые **тактиками**, являющиеся примитивами построения доказательства. Тактика действует на текущую цель, пытаясь построить доказательство исходного определения, возможно, на основе некоторых гипотетических суждений, добавляющихся затем к текущему списку определений. Процесс проведения доказательства с помощью тактик предусматривает активное участие пользователя и осуществляется, как правило, за несколько шагов, число которых может быть весьма значительным. При этом от него требуются довольно глубокие знания языка спецификаций. Таким образом, Coq функционирует как интерактивный прувер. Однако в Gallina также имеются тактики, пытающиеся построить доказательство без участия человека. Для их применения достаточно введения единственной команды, а в случае успеха выдается соответствующее сообщение. В

этом случае Coq действует подобно полностью автоматическим системам.

В завершение нашего знакомства с Coq укажем на такое важное качество системы, как модульность и расширяемость. Применяемые в системе понятия и тактики оформлены в виде модулей-библиотек, которые можно подключать по мере необходимости. При обычном старте в систему загружается минимальный набор библиотек, содержащий лишь логические связки и ряд арифметических понятий. Для работы в каких-то специфических предметных областях потребуются соответствующие дополнительные инструменты.

Таким образом, Coq является открытой расширяемой системой, способной выражать самые разные системы понятий и решать в них логические задачи. Так, в 2004 г. была завершена полная формализация в Coq упомянутой теоремы о четырех красках, что продемонстрировало ее высокие возможности.

Завершая этот краткий обзор, хотелось бы затронуть вопрос о сравнении интеллектуальных способностей человека и возможностей современных пакетов автоматических доказательств при проведении логических выкладок. Иными словами, способны ли сегодняшние системы доказательств конкурировать, скажем, с математиками, делающими это без применения компьютеров? При всей неоднозначности такой постановки вопроса считается, что машинные системы доказательств пока заметно отстают от профессиональных ученых, тогда как уровень среднего школьника и даже студента ими давно пройден. Для сравнения: компьютерные программы, играющие в шахматы, сегодня практически победили человека; имеется всего лишь несколько шахматистов, способных играть на равных с Fritz или Shredder, особенно с их двухпроцессорными версиями (которые обычно снабжаются приставкой Deep). При этом программы неустанно совершенствуются, так, настоящий фурор произвел шахматный движок Rybka (созданный всего одним человеком, правда, профессиональным шахматистом), возглавляющим сегодня все рейтинги. Вместе с тем имеются области, где пруверы вне конкуренции – например, в Software Engineering для проверки корректности программ или непротиворечивости требований к ПО. Поэтому хотя невозможно спрогнозировать, превзойдут ли машинные системы доказательств человека, в настоящее время они уже нашли

свои ниши, и в дальнейшем сфера их применения будет только расширяться.

Формирование и моделирование речевой коммунизации будем осуществлять средствами логики высказываний и логики предикатов.

7. Планирование решения коммуникационных задач и выпонение коммуникационных действий

7.1. Анализ систем решения коммуникационных задач

7.1.1. Среды функционирования коммуникаций и планы

Рассмотренные в предыдущих разделах методы представления и решения коммуникационных задач в коммуникационных системах создают необходимую концептуальную и алгоритмическую основу для перехода к анализу систем решения коммуникационных задач в коммуникационных системах (КС). Мы предполагаем, что КС существует и работает в среде обеспечивающей функционирование коммуникаций, которая обновляется как в результате действий самой КС, так и в результате некоторых независимых от КС действий в этой среде.

Будем предполагать, что КС существует и функционирует в среде, которая обновляется как в результате действий самой КС, так и в результате некоторых независимых от КС действий в этой среде. Автоматизированная коммуникационная система (АКС) обладает способностью воспринимать информацию о среде функционирования, рещать коммуникационные задачи и выполнять заданные действия (предписания, директивы) в этой среде. С алгоритмической точки зрения это означает, что АКС обладает описанием среды функционирования в виде ее (среды) модели, списком моделей элементарных действий (директив), или операторов, в общем случае изменяемых разработчиком АКС, либо автоматически в результате обобщения опыта функционирования АКС, а также общим механизмом создания целенаправленных стратегий в виде последовательности элементарных действий (директив) или

операторов и воплощения стратегий в рациональные программы действий АКС в окружающей среде.

Будем различать два больших класса окружающих сред — статические и динамические среды.

Статической средой мы назовем среду, в которой источником активных действий является коммуникационня система или коммуникационная сеть, управляемая централизованно, а факты о среде, представляемые в КС, не зависят в явной форме от времени. Соответственно мы можем указать следующие факторы, определяющие *динамичность среды:*

а) наличие в среде независимых от КС источников действий, изменяющих среду; к числу таких источников могут относиться другие КС или «слепые силы природы»;

б) наличие в среде фактов и определение в ней действий, явно зависящих от времени (например, факт OPEN (D1, 17, 19), т. е. «дверь D1 открыта с 17 до 19 часов», или оператор TURN (R1, 20), т. е. «вернуться в комнату R1 к 20 часам».

Следствием статичности среды является возможность упорядочения множества фактов, описывающих среду, по степени их неизменности в процессе решения коммуникационных задачи. Такое упорядочение является весьма относительным и может меняться от задачи к задаче. Так, если поставщик перемещает ящики, то факты об их цвете являются более неизменными, чем об их положении. Если же поставщик красит ящики, то ситуация оказывается противоположной Однако, по-видимому, положение ящика является всегда более неизменным фактом, чем положение поставщика.

Возможность ранжирования фактов по степени их неизменности является для нас важной (см. п. 7.4.1). Отметим, чго в сложных динамических средах можно и не найти такое ранжирование.

Целенаправленная стратегия поиска решения коммуникационной задачи называется *планом.*

Определим *пространство моделей M* как множество всех возможных состояний моделей среды, описываемых алгоритмической системой.

Моделью элементарного действия КС в среде, или *оператором,* называется некоторое отношение F_j, определенное в пространстве моделей, т. е. $F_j: M \rightarrow 2^M$. Множество всех операторов обозначим через F. В общем случае отношение, определяющее оператор, является частичным отношением, определенным на собственном подмножестве M_k множества M.

Теперь мы можем дать формальное определение плана

Планом называется помеченный направленный граф, удовлетворяющий следующим условиям:

I) каждая дуга графа помечена оператором $F_j \in F$, в общем случае оператором-схемой, т. е. семейством операторов, определяемым некоторым параметром;

2) с каждой вершиной графа n_k связана некоторая формула R_k, определяющая в свою очередь подмножество $M_k \subseteq M$,

M — множество моделей;

3) из одной вершины исходят одинаково помеченные дуги;

4) множество M_k, соответствующее n_k, содержится в области определения оператора F_j, помечающего дуги, исходящие из этой вершины.

Это определение является важным по нескольким причинам. Во-первых, оно определяет *обобщенный план,* т. е. множество планов, которое можно получить из исходного конкретной подстановкой значений параметров. Во-вторых, множество дуг, исходящих из одной вершины графа, определяет множество *возможных результатов применения* оператора, т. е. для каждого $m_i \in M_k$ $F_j(m_i)=\{m_1, m_2, \ldots, m_n\}$. Наконец, в-третьих, из определения вытекает применимость оператора F_j к любому состоянию модели $m_i \in M_k$.

План называется *простым,* если для всех вершин n_k оператор является однозначной функцией модели, т. е. $F_j(m_i)=m$, $m_i \in M_k$, $m \in M$ (из каждой вершины исходит ровно одна дуга).

План называется *сложным,* если каждому результату применения оператора m_p, $p=1, 2, \ldots, n$, приписывается некоторая оценка C_p правдоподобия его появления или оценка его полезности с точки зрения достижения цели (т. е. оценки приписываются дугам, исходящим из вершины n_k). В случае, если альтернативным результатам применения оператора не приписаны оценки (или, что то же самое, приписаны одинаковые оценки), *план* называется *составным.* План не обязательно может приводить к достижению поставленной цели. Мы вводим в связи с этим определение *полного* и *неполного плана*. План P называется *полным планом из начальной модели m_0 к целевой m,* если существует $P' \subset P$ такой, что 1) P' есть план; 2) все вершины P' достижимы из такой вершины n_r, что $m_k \in M_r$; 3) существует по меньшей мере одна такая вершина $n_t \in P'$, что R_t влечет за собой m. *План,* не удовлетворяющий хотя бы одному из этих условий, называется *неполным.*

Алгоритмическая система КС, осуществляющая построение планов решения коммуникационной задачи, называется *планирующей системой* (ПС).

7.1.2. Планы и действия

Проблема планирования для КС занимает важное место в проблемах решения коммуникационных задач. Поскольку КС должна функционировать в реальной среде, т. е. среде, неполностью соответствующей той модели среды, которая заложена в КС, взаимосвязь проблемы планирования и выполнения действий является принципиально важным фактором, определяющим возможность выполнения КС поставленных перед ней задач.

В отличие от оператора, *элементарное действие* КС в среде функционирования есть отношение, определенное на декартовом произведении пространств среды W и моделей M и отображающее его в самого себя,

$$Q_j : W \times M \rightarrow W \times M. \qquad (7.1)$$

Таким образом, мы можем выделить в действии две компоненты: компоненту нового состояния среды

$$w = Q_{j,w}(w_i, m_i) \qquad (7.2)$$

и компоненту нового состояния модели этой среды в КС

$$m = Q_{j,w}(w_i, m_i). \qquad (7.3)$$

Идесь Q_j представляет из себя *действие-схему,* т. е. множество действий, определяемое некоторым параметром.

В силу неточности и неполноты моделей для достаточно сложных сред, а также погрешностей восприятия окрежающей среды КС и ее подсистем мы не можем ожидать функционального соответствия между средой и ее моделью в КС, т. е. отношение моделирования $R: W \rightarrow M$ не является функцией n в строгом смысле слова, что не позволяет в общем случае надечться на полную формализацию процесса построения адекватных моделей среды функционирования КС.

При создании формальной теории планирования предписаний действий в КС естественным решением является организация обратной связи, позволяющей корректировать модель среды в соответствии с действительными изменениями, происходящими в ней.

Выполнение планов и корректировка модели среды в соответствии с действительными результатами их выполнения осуществляются специальной алгоритмической системой КС, называемой *исполняющей системой* (ИС). Таким образом, ИС выполняет план, обращаясь к определенным действиям, которые, как предполагается, соответствуют операторам в плане, получает планы (в том числе неполные), обращаясь к ПС, и выдает команды ПС о корректировке плана в случае выявления несоответствий действительного состояния среды и ее

модели в КС. В каждом состоянии ИС должна осуществить выбор между планированием и действием.

Определения полных и неполных, простых, составных и сложных планов создают основу для классификации ИС. Эта классификация приведена в таблице 7.1.

Таблица 7.1

		Наличие только полных планов	Наличие полных или неполных планов
Отсутствие обратной связи (только простые планы)		A	B
Наличие обратной связи	Простые планы	C	F
	Составные планы	D	G
	Сложные планы	E	H

Примечание. Буквы идентифицируют класс ИС для таблицы 7.2 и дальнейшего изложения.

Здесь под обратной связью понимается проверка модели с помощью ИС после каждого шага выполнения плана. В таблице 7.2 сведены основные функциональные особенности ИС различного класса.

Таблица 7.2

Класс ИС	Характер взаимодействия ПС и ИС	Действия ПС при несоответствии мира и текущей модели	Характер работы ИС
A	Полное соответствие последовательности действий плану (даже если он неудачен)	Несоответствие не учитывается, так что управление к ПС в процессе выполнения плана не передается	Простое прохождение списка действий
B	Полное соответствие на уровне неполного плана	При несоответствии неполного плана результатам его выполнения возможен пересмотр плана при хранении дерева планирования с оценками полезности	Простое прохождение списка действий, соответствующего неполному плану
C	Полное соответствие при успешном выполнении. Обмен информацией для проверки соответствия модели миру после каждого шага выполнения	Полное перепланирование, однако при хранении дерева планирования возможно использование промежуточных результатов	Выполнение действия — проверка соответствия — сигнал о выполнении (соответствие) или о перепланировании (несоответствие)
D E	То же, что и C, но производится проверка и модели, и плана	Поиск дочерней вершины, соответствующей новому состоянию модели. Если такой вершины нет, аналогично C	Выполнение действия — проверка соответствия — поиск возможных продолжений, при несоответствии — перепланирование в случае неудачи
F G H	То же, что в C / То же, что в D и E — На уровне неполного плана. Выбор планирования или выполнения в соответствии с оценками	То же, что в C, D и E соответственно для неполного плана. Перепланирование по сигналу ИС	Решение задачи выбора планирования или выполнения на каждом шаге

Наиболее интересным является выбор направления деятельности ИС в случае работы с неполными планами. На каждом этапе решения задачи ИС стоит перед дилеммой: выполнять действия в соответствии с намеченным неполным планом или продолжать построение неполного плана. Каждая из альтернатив может привести к нежелательным результатам. Действительно, при выполнении неполного плана робот может не достигнуть цели, в то время как дальнейшее планирование могло бы это показать. С другой стороны, продолжение планирования может оказаться нецелесообразным, если выполнение уже построенного плана могло бы показать его бесперспективность. Таким образом, планирование и выполнение для ИС класса F, G и H являются конкурирующими действиями. С каждым из этих действий могут быть связаны некоторые оценки полезности в виде, например, функции стоимости уже построенного плана и оценки стоимости остатка плана, приводящего к цели. Другим возможным подходом являлась бы формулировка задачи планирования как игры с природой. Во всяком случае, именно здесь открываются возможности использования изложенных в предыдущих двух разделах алгоритмов. В свою очередь ИС по результатам своих действий и оценивая текущее состояние планирования, могла бы определить в каждом конкретном случае, планировать или выполнять.

Постановка задачи о совместной оптимизации процессов планирования и выполнения связана, по-видимому, с некоторым обобщением процессов в единый процесс, аналогичный процессу решения задачи. Два таких возможных обобщения мы рассмотрим в п. 7.5.2.

В заключение этого параграфа мы отметим еще одну задачу, встающую в связи с проблемой планирования и выполнения действий,— задачу обобщения моделей и уже найденных планов и использования их как в последующих процессах построения планов, так и при выполнении действий. Один шаг на пути к такому обобщению уже сделан введением операторов-схем. Дальнейшее развитие идей обобщения — это постепенное наращивание множества операторов путем их укрупнения, а также соответствующее обобщение условий их применения, с тем чтобы создать иерархию планов от укрупненных к более детализированным.

Заучивание и обобщение получаемых в процессе решения задач успешных частичных решений является предметом исследований в области теории коммуникаций. Мы рассмотрим идеи решения коммуникационных задач в преломлении к проблемам планирования и выполнения действий в КС.

7.2. Планирующая система «Решатель коммуникационных задач»

Рассмотрим вопросы, связанные с планированием и выполнением действий в КС, кратким описанием планирующей системы «Решатель коммуникационных задач» (ПС РКЗ) по следующим причинам:

1) система представляет весьма широкий класс универсальных решателей задач для КС, работающих в достаточно сложной среде;

2) ПС РКЗ является, видимо, пока единственной системой, которая работает в реальной среде, взаимодействуя с ИС «Исполнитель планов» (ИП);

3) ПС РКЗ является иллюстрацией проблем планирования и выполнения действий и возможных путей их решения;

4) формализм ПС РКЗ отражает все преимущества и недостатки декларативных предписаний, создавая в то же время приемлемую основу для ряда обобщений.

Тривиальная (примитивная) АКС ПС РКЗ функцинрует в среде, состоящей из комнат с дверьми и предметами (ящики, призмы), и способна в автоматическом режиме осуществлять с этими предметами относительно простые коммуникации.

Среда АКС представляется в ПС в виде модели, состоящей из набора правильно построенных формул (ппф) в исчислении предикатов первого порядка, описывающих состояние среды в данный момент.

Действия в АКС моделируются множеством операторов, определяемых *наименованием, списком параметров,* а также *условиями применимости (предусловиями)* и *результатами действия* в виде *схем ппф,* т. е. ппф, зависящих от параметров.

Результаты действия оператора описываются *списком вычеркивания* тех схем ппф, которые перестают быть истинными после применения оператора, и *списком добавлений* схем ппф, которые становятся истинными после применения оператора.

Важно различать параметры, входящие в схему ппф, и обычные связанные переменные, т. е. переменные под знаком кванторов общности или существования. В связи с этим системы доказательства предписаний, описанные в р. 6, требуют некоторой модификации.

Пусть имеется некоторая целевая схема ппф $G(p)$, p — множество параметров схемы, которую нужно доказать на множестве M дизъюнктов, т. е. найти опровержение множества дизъюнктов $M \cup \{\sim G(p)\}$. Для вычисления частного случая p' множества p, при котором множество $M \cup \{\sim G(p)\}$ невыполнимо, можно использовать стандартный алгоритм унификации (п. 6.2). С помощью этого алгорит-

ма можно найти наиболее общие частные случаи параметров, при которых обеспечивается унификация. Однако необходимо определить, какие подстановки допустимы в случае параметров. Определим следующие типы термов, которые могут быть подставлены вместо переменной: переменные, константы, параметры и функциональные термы, не содержащие переменных. Вместо параметра могут быть подставлены следующие типы термов: константы, параметры и функциональные термы, не содержащие функций Сколема, переменных или параметров.

Поскольку один и тот же параметр может иметь несколько вхождений в множестве дизъюнктов, он должен замещаться при резолюции (директиве) термом во всех дизъюнктах, являющихся производными от резольвенты.

В качестве примера приведем оператор «переместить объект *k* из места *m* в место *n*». Наименование:

PUSHTO *(k, m, n) (k, m, n* — параметры).

Предусловия:

At (Отправитель, *m)* $\wedge$ At *(k, m)* (и отправитель, и объект *k* должны быть в месте *m)*.

Список вычеркиваний:

At (Отправитель, *m*);

At *(k, m)* (отправитель и объект *k* больше не находятся в месте *m)*.

Списокдобавлений: At (Отправитель, *n);*

At(*k, n)* (отправитель и объект *k* находятся в месте *n)*.

Конечная модель или цель также описывается в виде ппф.

Трудности использования формальных методов доказательства предписаний в качестве стратегий поиска плана, главным образом связанные с проблемой границ, оказались непреодолимыми. В связи с этим в ПС процесс поиска плана полностью отделен от метода доказательства предписаний. Последний используется только внутри модели среды для ответа на вопросы, связанные с анализом применимости операторов и проверкой выполнимости условий достижения цели.

Для поиска решения в пространстве моделей ПС РКЗ использует описанный ранее механизм редукции GPS. Преимущество такого комбинированного подхода заключается в возможности рассмотрения сложных моделей, используя описательную мощь исчисления предикатов первого порядка, и использования эффективных эвристик разбиения цели на подцели, свойственных механизму редукции GPS.

В процессе поиска механизм редукции порождает иерархию цели, подцелей и моделей, которую можно представить в виде дерева

поиска. Каждая вершина дерева имеет вид (модель, (список целей)) и соответствует задаче достижения по порядку подцелей из списка целей указанной модели среды. Схема работы ПС представлена на рис. 7.1.

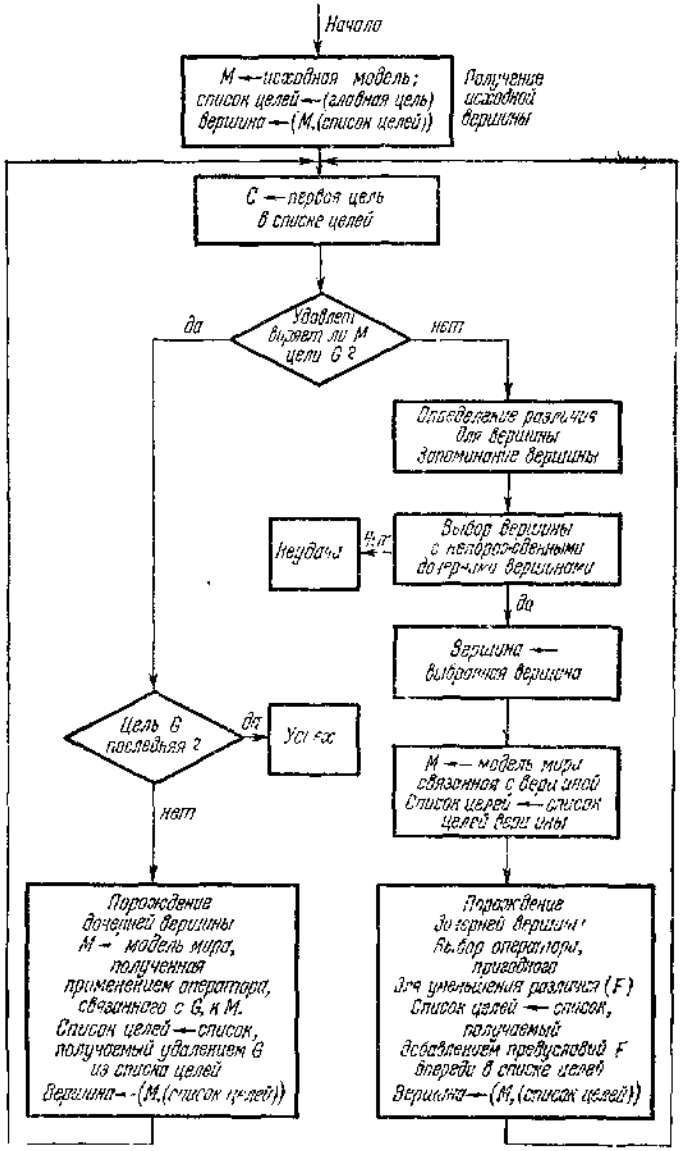


Рис. 7.1. Схема работы планирующей системы РКЗ.

Мы рассмотрим принципы работы системы на примере и дадим для этого примера дерево поиска. Мир отправвителя изображен на рис. 7 2.

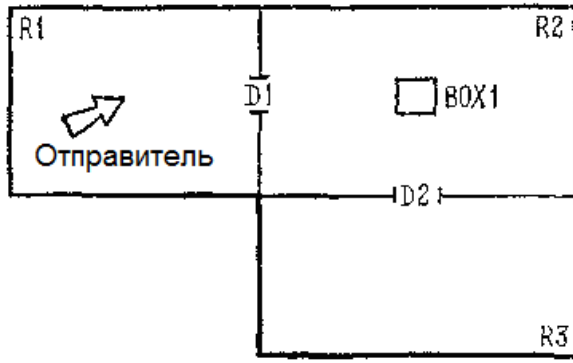


Рис. 7.2. Мир отправителя для примера задачи, решаемой ПС РКЗ.

Пусть перед отправителем ставится задача передвинуть ящик из комнаты R2 в комнату R1.
Начальная модель M_o.
M_o: INROOM (Отправитель, R1) — отправитель находится в комнате R1,
CONNECTS (Dl, Rl, R2) — комнаты R1 и R2 соединены дверью D1.
CONNECTS (D2, R2, R3) — комнаты R2 и R3 соединены дверью D2.
BOX (BOX1) — ящик есть BOX1.
INROOM (BOX1, R2)— ящик BOX1 находится в комнате R2.
$(\forall x \forall y \forall z)$ [CONNECTS (x, y, z)→CONNECTS (x, y, z)]— порядок комнат в предикате CONNECTS безразличен.
Целевая ппф G_o.
G_o: $(\exists x)$[BOX(x) $\wedge$ INROOM (x, Rl)].
Нам задан список из двух операторов:
1) GOTHRU(d, $r1$, $r2$) — отправитель идет через дверь d из комнаты $r1$ в комнату $r2$ (d, $r1$, $r2$— параметры).
Предусловия:
INROOM (Отправитель, rl) $\wedge$ CONNECTS (d, rl, r2).
Список вычеркиваний:
INROOM (Отправитель, $), где $ может принимать любые значения.
Список добавлений.
INROOM (Отправитель, $r2$).

2) PUSHTHRU *(b, d, r*1, *r2)* — отправитель толкает объект *b* через дверь *d* из комнаты *r*1 в комнату *r2 (b, d, r*1, *r2* — параметры).
Предусловия:
INROOM *(b,* rl) ∧ INROOM (Отправитель, *r*1) ∧ CONNECTS (*d*, rl, *r*2).
Список вычеркиваний:
INROOM (Отправитель, $);
INROOM *(b*, $).
Список добавлений
INROOM (Отправитель, r2);
INROOM (*b*, r2).
Поиск решения начинается с попытки доказать, что G_0 следует из M_o Эта попытка терпит неудачу, однако часть целевой ппф BOX *(x)* удовлетворяется при *x*=BOX1. Поэтому определяется различие между G_0 и M_0 в виде INROOM (BOX1, Rl). Система определяет, что частичная подстановка PUSHTHRU (BOX1, *d*, rl, Rl) может обеспечить это предписание, так как в его списке добавлений имеется INROOM *(b, r2)*. Тогда в качестве новой цели G1 устапавливаются предусловия этого оператора с соответствующими подстановками параметров, т. е.
Gl: INROOM(BOX1, *r*1) Л INROOM (Отправитель, *r*1) ∧
∧ CONNECTS(*d*, *r*l, Rl)
Gl не может быть доказано из M_o и определяется различие между ними в виде INROOM (Отправитель, R2), так как при *r*1= R2 и *d=D*l остальные литералы G1 имеются в M_o. Система устанавливает, что для устранения этого различия уместен оператор GOTHRU при r2=R2 Устанавливается очередная подцель
G2 INROOM (Отправитель, rl) ∧ CONNECTS(*d*, *r*l, R2), которая может быть выведена из M_o при rl=Rl и *d=D*l Поэтому к M_0 применяется GOTHRU (D1, Rl, R2), преобразуя ее в
Ml: INROOM (Отправитель, R2);
CONNECTS (Dl, Rl, R2);
CONNECTS (D2, R2, R3);
BOX (BOX 1);
INROOM (BOX 1, R2);
(∀ *x* ∀ *y* ∀ *z)*[CONNECTS *(x, y,* z)→CONNECTS (*x, z,y*)].
Теперь делается попытка доказать Gl из новой модели Ml. Эта попытка приводит к успеху при rl=R2, *d=D*1. Таким образом, к Ml применяется оператор
PUSHTHRU (BOX1, Dl, R2, Rl),
так как остальные подстановки были сделаны ранее. Модель Ml преобразуется в
M2: INROOM (Отправитель, Rl);
CONNECTS (Dl, Rl, R2);

CONNECTS (D2, R2, R3);
BOX (BOX 1);
INROOM (BOX 1, Rl),

$(\forall x \forall y \forall z)$ [CONNECTS $(x, y, z) \rightarrow$ CONNECTS (x, z, y)].

Теперь делается попытка доказать G_0 из М2. Эта попытка оказывается успешной, так что решением является план

GOTHRU (Dl, Rl, R2); PUSHTIIRU(BOX1, Dl, R2, Rl). (7.4)

Дерево поиска для этой задачи представлено на рис. 7 3.

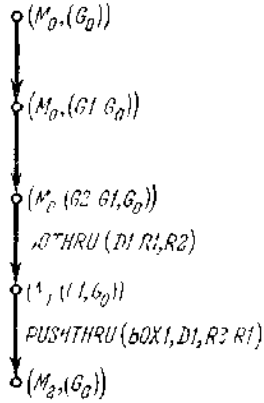


Рис. 7.3. Дерево поиска для примера задачи

В данном случае наше дерево выродилось в путь, поскольку в примере не возникло альтернативных возможностей (выбор оператора GOTHRU, а не PUSHTHRU, на втором шаге, объясняется точным совпадением различия со списком добавлений GOTHRU). Естественно, что в более сложных задачах и при большем списке операторов деревья поиска могут быть существенно «ветвистее».

Следует отметить два обстоятельства, не затронутые нашим простым примером.

Во-первых, порядок образования дуг дерева поиска, соответствующих применению операторов, вовсе не предопределяет вхождение операторов в этом порядке в окончательный план. Это лишний раз подчеркивает гибкость стратегии поиска механизма редукции GPS, сочетающего в себе свойства как прямого, так и обратного поиска.

Во-вторых, ПС РКЗ снабжена эвристическим механизмом выбора узлов дерева поиска. В системе используется оценочная функция, учитывающая такие факторы, как число оставшихся целей в списке целей, число и тип предикатов в оставшихся выражениях цели, а также сложность различий, связанных с данным узлом

Как отмечалось ранее, главной трудностью использования механизма редукции GPS является выбор операторов, пригодных для устранения или уменьшения различий. В рассматриваемой системе этот вопрос решается следующим образом Предположим, что в дереве поиска образован узел *(M,* (G_i, G_{i-1},…, G_0)*)*, причем система доказательства предписаний пытается доказать невыполнимость множества $M \bigcup (\sim G_i\}$. Если доказательство успешно, то к модели *M* применяется оператор с предусловиями G_i. Если же в течение определенного времени опровержение не будет найдено, то незавершенное доказательство или, в случае, если оно велико, его часть, выбираемая из эвристических соображений, и берется в качестве различия между *M* и G_i и связывается с данным узлом

Процесс выбора пригодного оператора происходит в два шага. На первом шаге создается упорядоченный список операторов-кандидатов. Выбор кандидатов основан на простом сравнении предикатов в различии с предикатами списков добавлений операторов. На втором шаге программа доказательства предписаний определяет, могут ли директивы из списка добавлений оператора резольвировать с директивами в различиях. Если новые резольвенты являются производными от директив в списке добавлений, то соответствующий оператор объявляется пригодным. Заметим, что из одного оператора-схемы может быть образовано несколько пригодных частных случаев.

Ранее мы отмечали трудности решения проблемы границ для декларативных предписаний. Представление результатов применения операторов в виде списков добавлений и списков вычеркиваний решает часть этой проблемы в духе метода контекстов и контекстных графов. Однако, как отмечалось, проблема выведенных фактов этим методом не решается, так что в общем случае требуется вновь выводить их в каждом новом состоянии, т. е. в каждой новой модели. Эта проблема разрешается в ПС РКЗ путем задания множества простейших предикатов, имеющих наинизший ранг в смысле, указанном в п. 7.1.1. Остальные предикаты связываются с этим простейшим множеством. Другими словами, любой производный дизъюнкт связывается с теми из простейших предикатов, от которых зависит его истинность. Таким образом, предикат On(B2, B1) должен быть связан с предикатом At (B1, *A)*. Тогда истинность всех производных дизъюнктов может быть определена непосредственно из списков вычеркивания соответствующих операторов.

7.3. Обобщение планов и планирование с помощью макрооператоров

7.3.1. Представление планов

Задача обобщения планов состоит в том, чтобы после построения успешного плана преобразовать этот конкретный план в план, который мог бы быть затем использован для множества подобных задач. Другими словами, мы хотим получить план-схему, т. е. **параметризованное семейство планов**. При этом необходимо, чтобы такой обобщенный план мог быть использован как в последующих процессах планирования в качестве дополнительного к имеющемуся списку операторов, так и при выполнении планов, составленных из таких обобщенных планов-операторов, с помощью ИС.

Прежде всего необходимо определить формализм представления планов, в понятиях которого можно было бы описать и решить поставленную задачу.

Таким формализмом является *треугольная таблица* (ТТ), строкам и столбцам которой соответствуют операторы плана. Пример ТТ представлен на рис. 7.4.

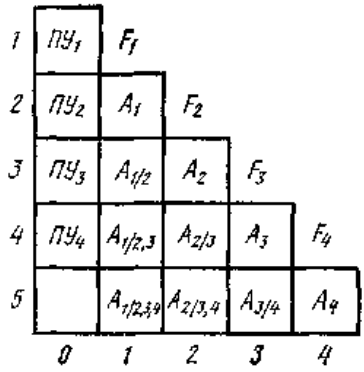


Рис. 7.4. Треугольная таблица.

Столбцы ТТ, кроме нулевого, помечены операторами F_1, F_2, F_3, F_4 составляющими план. Обозначим через $c_{i,j}$ ячейку ТТ, i— номер столбца, j — номер строки.

В общем случае для каждого столбца i, $i \neq 0$, в ячейку $c_{i,\,i+1}$ помещается *список добавлений* A_i *оператора* F_i. В ячейки $c_{i,\,i+k}$, $k=2, 3, \ldots n+1-i$ (n— число операторов в плане), помещаются те предписания списка

добавлений оператора F_i, которые остаются неизменными после применения операторов F_{i+1}, F_{i+2}, ..., F_{i+k-1}. Они обозначаются через $A_{i,\ i+1,i\ +2,\ldots..,\ i+k-1}$. В нашем примере $A_{1/2}$ означает те предписания A_1, которые остались неизменными после применения F_2, $A_{1/2,3}$ — те предписания $A_{1/2}$, которые остались неизменными после применения F_3 и т. д.

Рассмотрим теперь строку j ТТ. Эта строка (исключая ячейку $c_{0,\ j}$) содержит список добавлений, получаемый применением последовательно F_1, F_2, . . ., F_{j-1}, или (j—1)*-й «шапкой» плана.* В частности, (n+1)-я строка содержит список добавлений, полученный после выполнения всего плана.

Назовем множество предписаний, используемых для доказательства предусловий оператора, *поддержкой предусловий*. Необходимо, чтобы строка j содержала все ппф в поддержке предусловий F_j.

Часть таких ппф будет добавлена (j—1)-й шапкой плана и поэтому будет включена в строку j. Остающиеся необходимые ппф находятся в начальной модели и не были вычеркнуты ни одним из F_k, k=1, 2, . . ., j—1. Эти предписания, обозначенные нами ПУ$_j$, и помещаются в столбце i, i=0. Следовательно, нулевой столбец ТТ содержит те предписания из начальной модели, которые были использованы в доказательствах предусловий для плана. Отметим, что ПУ$_j$, j=1,2, . . ., n, отнюдь не содержат полное описание начальной модели.

Назовем предписания, входящие в поддержку предусловий F_j, j=1, 2, . . ., n, *отмеченными предписаниями*. По построению ТТ все предписания в ячейке $c_{0,\ j}$, j=1, 2,. . . . . ., n, являются отмеченными. Однако не обязательно все предписания в $c_{0,\ j}$, $i\neq0$, являются отмеченными.

На рис. 7.5 приведена ТТ для плана решения задачи, рассмотренной в § 7.2. Звездочка указывает отмеченные предписания.

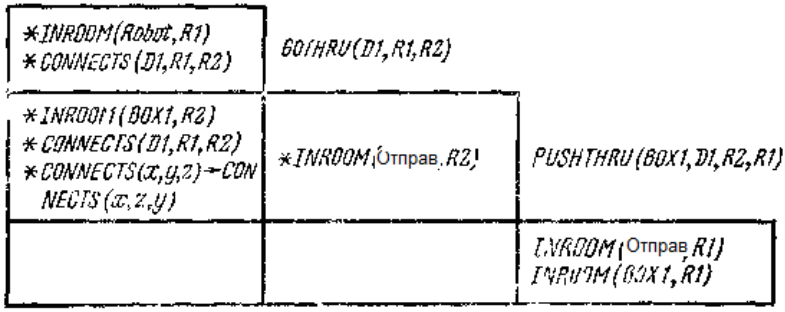


Рис. 7.5. Треугольная таблица для примера (§ 6.2).

Таким образом, отмеченные предписания в *j*-й строке составляют поддержку предусловий F_j. Представляет интерес исследование условий применимости для последовательности операторов $F_j, F_{j+1}. . ., F_n$, т. е. *j-го остатка плана*. Очевидно, что *j*-й остаток плана применим к модели, если она содержит ту часть поддержки предусловий F_k, $k=j, j+1, \ldots, n$, которая не вырабатывается внутри самого остатка.

Назовем *j-м ядром ТТ* такую прямоугольную подтаблицу ТТ, что она содержит ячейку $c_{0,\ n+i}$ и строку *j*. Мы утверждаем, что *j*-й остаток плана применим к модели, если все отмеченные предписания в *j*-м ядре истинны в этой модели.

Действительно, если все отмеченные предписания в *j*-м ядре истинны, то F_j применим к модели. В результате применения F_j получится новая модель, в которой будут истинны предписания A_j. Поскольку по построению таблицы и отмеченные предписания внутри *j*-го ядра истинны, то все отмеченные предписания в строке $j+1$ истинны; следовательно, F_{j+1} также применим. Продолжая аналогичные рассуждения относительно строк $j+2, . . ., n$, полччаем, что *j*-й остаток плана применим к модели.

Таким образом, доказано достаточное условие применимости *j*-го остатка плана.

Заметим, что первое ядро устанавливает достаточные условия применимости плана в целом, т. е. конъюнкция всех предписаний в нулевом столбце представлюет собой предисловия для всего плана. В свете доказанного достаточного условия выполнение плана может рассматриваться как последовательное преобразование ядер ТТ, т. е. для всех j, $F_j(K_j)=K_{j+1}$, где K_j — *j*-е ядро ТТ.

7.3.2. Обобщение планов

Опишем процедуру обобщения плана, представленного в виде ТТ. Эта процедура осуществляется в три шага.

На первом шаге мы преобразуем исходный конкретный план в наиболее общую форму. Этот шаг осуществляется следующим образом:

1) Каждое появление константы в первом ядре, в том числе одной и той же, замещается отдельным параметром.

2) Остальные столбцы озаглавливаются описаниями операторов-схем с параметрами.

3) В предписаниях в столбцах i, i=l, 2, . . ., n, исходного плана все константы заменяются соответствующими параметрами операторов-схем F_i.

В нашем примере ТТ (рис. 7.5) преобразуется в ТТ рис. 7.6).

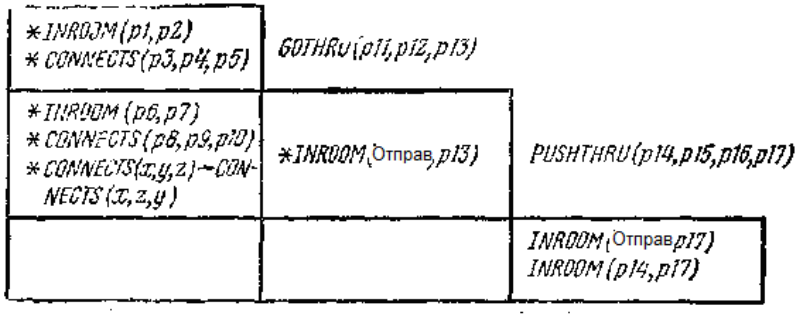


Рис.7.6. Треугольная таблица для примера (§ 7.2) после первого шага процедуры обобщения.

На втором шаге мы вводим такие ограничения на полученные параметры, чтобы, с одной стороны, отмеченные предписания в строке j, j=1, 2, . . ., n, были поддержкой оператора F_j и, с другой стороны, чтобы исходный план оставался частным случаем получающейся ТТ. Этот шаг осуществляется следующим образом:

1) Извлекаются графы опровержения, построенные в процессе доказательства предусловий всех операторов исходного плана.

2) Для каждого такого графа, соответствующего оператору F_j, строится изоморфный образ выполнением на каждом шаге резолюций тех же предписаний и унификаций тех же литер, причем в качестве аксиом используются отмеченные предписания из строки j, а в качестве доказываемого предписания — предусловия оператора-схемы F_j из ТТ, полученной на первом шаге.

3) Все получаемые в процессе доказательства подстановки вносятся в полученную на первом шаге ТТ.

На рис 7.7 и 7.8 приведены деревья опровержения для предусловий операторов ТТ (рис. 7.6).

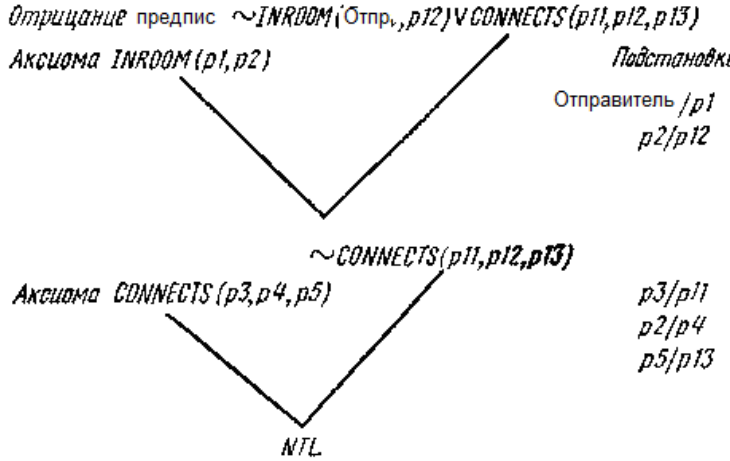


Рис. 7.7. Дерево опровержения для предусловий оператора GOTHRU (p11, p12, p13).

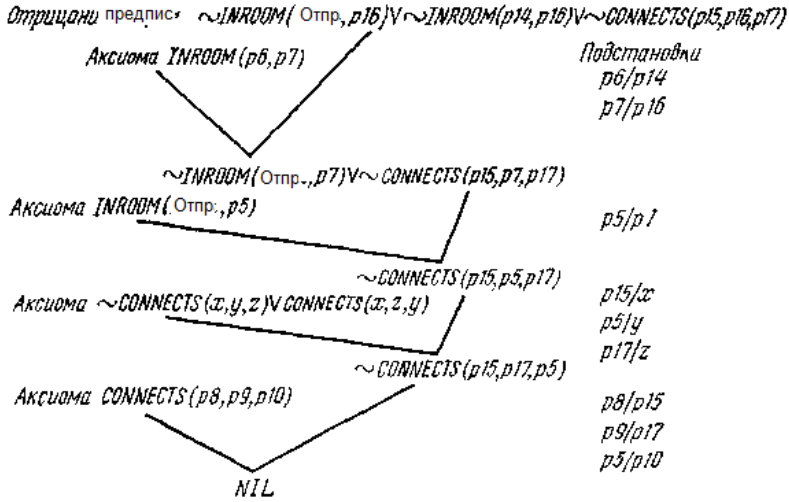


Рис. 7. 8. Дерево опровержения для предусловий оператора PUSHTHRU (p14, p15, p16, p17).

После соответствующих подстановок ТТ (рис 7. 6) преобразуется в ТТ (рис. 7.9),

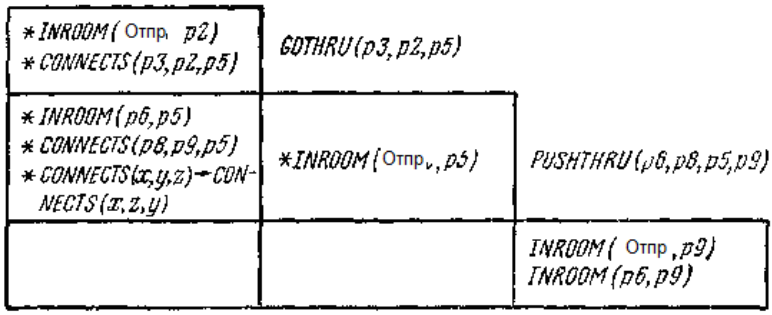


Рис. 7. 9. Обобщенная треугольная таблица для примера (§7.2).

На третьем шаге мы проводим два дополнительных преобразования, имеющие своей целью исключить излишние обобщения, а также один вид возможного противоречия.

Источником излишнего обобщения является случай, когда одно и то же предписание в начальной модели плана входит в поддержку более чем одного оператора. Тогда на первом шаге процедуры обобщения это предписание преобразуется в два различных предписания. Во многих случаях это приводит к полезным обобщениям. Так, например, произошло в нашем примере (рис. 7. 9), когда CONNECTS (D1, Rl, R2) преобразовалось в CONNECTS *(p3, p2, p5)* и CONNECTS *(p8, p9, p5)*, что обобщило исходный план в том отношении, что теперь отправитель может передвинуть ящик в некоторую третью комнату, а не обязательно в ту, в которой первоначально находился сам.

Однако, если бы различные параметры (в нашем примере *p2* и p9), полученные из одной константы, не использовались бы оба как аргументы операторов в плане, их без ущерба можно было бы связать вместе с последующей подстановкой одного вместо другого в ТТ.

Источником возможных противоречий в обобщенной ТТ является сделанное нами на шаге 1, п. 3, молчаливое предположение о том, что вычеркивания в наиболее общей ТТ могут быть теми же самыми, что и в исходной ТТ.

Рассмотрим случай, когда в исходной ТТ имеются два одинаковых предиката с различными аргументами (например, At (BOX1, Rl) и At(BOX2, R2)). На первом шаге обобщения они превратятся в одинаковые предикаты с различными параметрами (например, *At(p1, p2)* и At(p3, p4)). Предположим, что в результате второго шага обобщения эти параметры продолжают оставаться различными, хотя, возможно, и отличаются от *p1, p2* и *p3, p*4 соответственно. Далее,

вполне возможно, что при использовании обобщенного плана и нахождении его частного случая часть соответствующих параметров в предикатах будет связана с одной и той же константой, в то время как другая часть соответствующих параметров предикатов свяжется с различными константами. Это и приводит к противоречию. В приведенном выше примере эта ситуация возникла бы, если бы *p1* и p3 были связаны с одним объектом, скажем, BOX7, а *p2* и *p4* — с различными местоположениями (R4 и R8 соответственно). В результате мог бы быть доказан план (или его часть), в котором BOX7 одновременно находился бы в R4 и R8. Ясно, что в конкретном плане предикат At(pl, *p2)* был бы вычеркнут, прежде чем была установлена истинность предиката At(p3, *p4)*.

В общем случае мы должны провести коррекшровку обобщенной ТТ путем анализа списков вычеркивания всех операторов, входящих в план. Алгоритм коррекции обобщенной ТТ работает с полученной на втором шаге таблицей и анализирует последовательно списки вычеркивания операторов F_1, F_2, ..., F_n. Рассмотрим совместно высказывания в строке j и список вычеркиваний F_j. Очевидно, что применение списка вычеркивания к строке будет менять ее тогда, когда некоторые из предписаний строки будут унифицироваться с некоторыми из литерсписка вычеркиваний и когда для унификации потребуется, чтобы некоторый параметр *p1* замещался другим параметром p2 или константой С. Если $p1=p2$ или $p1=C$, то предписание *R* вычеркивается, иначе оно остается неизменным и переходит в строку j+1. Это условное вычеркивание разрешается путем замещения предписания в строке j+1 одной из импликаций:

а) $p1 \neq p2 \rightarrow R$,

б) $p1 \neq C \rightarrow R$. (7.5)

Предположим далее, что замещенное предписание в (j+1)-й строке отмечено, т. е. входит в поддержку предусловий F_{j+1} (если оно не отмечено, алгоритм заканчивает работу). Тогда для того, чтобы поддержка предусловий F_{j+i} сохранилась после введения одной из импликаций (7.5), мы должны добавить в ячейку $c_{0,\ j+1}$ отмеченное предписание вида

а) $p1 \neq p2$

или (7-6)

б) $p1 \neq C$,

так что прежнее в предписание легко выводится из (7.5, *а)* и (7.6, *а)* или (7.5, *б)* и (7.6, *б)*.

На этом третий шаг обобщения плана завершается, и полученный обобщенный план может быть занесен в список операторов системы. Мы будем называть такой план *макрооператором.*

7.3.3. Особенности планирования с макрооператорами

Рассмотренный в предыдущем параграфе процесс образования макрооператоров представляет собой типичный пример обучения ПС в ходе решения ею задач.

Этот процесс обеспечивает непрерывное обогащение возможностей системы при условии, что она будет способна

1) использовать макрооператор и любую его часть в процессе планирования;

2) свести к минимуму в каждом конкретном использовании избыточность макрооператора, являющуюся естественным следствием его обобщенного предписания;

3) регулировать процесс накопления макрооператоров в системе.

Предположим, что в процессе планирования для уменьшения различия выбирается некоторый макрооператор, представляющий последовательность операторов $F_1, F_2, \ldots, F_n$. Такой макрооператор содержит n различных *списков добавления* A_{1j}, j=2, 3, . . ., n, соответствующих j-й шапке плана. Эти списки добавлений могут быть использованы обычным образом для выбора наиболее пригодного из них с позиции уменьшения различия. В случае выбора равнокачественных списков добавлений естественно выбирать список с наименьшим j, поскольку ему будет соответствовать более короткая последовательность.

Пусть выбран список добавлений A_{1j}. Очевидно, что для целей планирования нас не интересует (j+1)-й остаток полного плана, так что из j-й шапки плана можно без ущерба удалить те операторы, которые предназначены для формирования предписаний, входящих в поддержку предусловий всех F_k, $k=j+1, \ldots, n$. Кроме того, для целей планирования не нужны и те операторы, результатом которых являются предписания, не используемые для установления пригодности списка добавлений A_{1j}.

Эти соображения приводят нас к следующей формулировке алгоритма упрощения макрооператора:

1) Производится отметка тех предписаний из A_{1j}, которые необходимы для установления пригодности $A_{1,j}$ (т. е. входят в неполный вывод, являющийся различием).

2) Снимаются метки у всех предписаний, являющихся поддержкой предусловий F_{j+1}.

3) Находится первый столбец справа, начиная с j-го, не содержащий отмеченных предписаний. Пусть таким столбцом будет i_k-й, $i_k \leq j$.

Тогда снимаются метки у всех предписаний в i_k-й строке, а оператор F_{i_k} исключается из j-й шапки плана.

4) Процесс, указанный в п. 3), повторяется вплоть до первою столбца. Оставшаяся в результате подпоследовательность операторов выдается как макрооператор, пригодный для уменьшения установленного различия.

Пример. Рассмотрим ТТ (рис. 7.10).

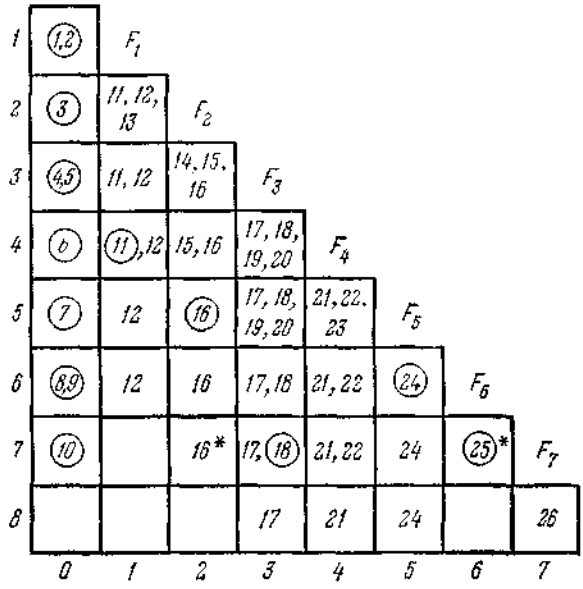


Рис. 7.10. Пример ТТ для иллюстрации алгоритма упрощения макрооператора.

Числа в ячейке представляют предписания, отмеченные предписания обведены кружками, предписания, пригодные для уменьшения различий, отмечены звездочками. Структура ТТ может быть прослежена с помощью таблицы 6.3, где I — начальное, а G — конечное состояние плана.

Таблица 7.3

Оператор	Источник поддержки предусловий	Адресат поддержки предусловий	Оператор	Источник поддержки предусловий	Адресат поддержки предусловий
F_1 F_2 F_3 F_4	I I I I, F_1	F_4 F_5 F_7, G G	F_6 F_8 F_7	I, F_2 I, F_5 I, F_3, F_6	F_8, G F_7 G

В нашем примере j=6. Алгоритм работает следующим образом (предписания 16 и 25 отмечены как уменьшающие различие).

1) Снимается метка у предписания 18.
2) Столбец 4 — первый справа, не содержащий отмеченных предписаний. Снимается метка у предписания 11, оператор F_4 исключается.
3) Столбец 3 — следующий столбец, не содержащий отмеченных предписаний (метка у предписания 18 уже снята). Оператор F_3 исключается.
4) Столбец 1 — следующий, не содержащий отмеченных предписаний (метка у предписания 11 уже снята). Исключается оператор F_1.

В результате получается последовательность операторов F_2, F_5, F_6, представляющая собой упрощенный макрооператор. На рис. 6.11 представлена ТТ для этого макрооператора, а таблица 7.4 показывает структуру ТТ (рис. 7.11).

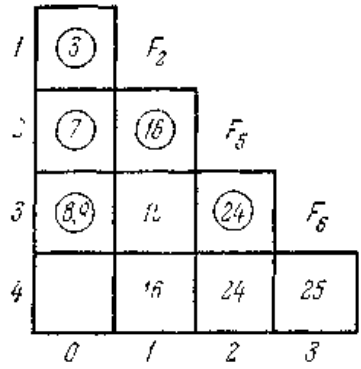


Рис. 7.11. Результат упрощения ТТ (рис. 7.10)

Таблица 7.4

Оператор	Источник поддержки предусловий	Адресат поддержки предусловий
F_2	I	F_5, G
F_5	I, F_2	F_6, G
F_6	I, F_5	G

Отметим, что мы не снимали метки у предписаний в нулевом столбце, так как соответствующие предписания исключаются автоматически при исключении операторов.

Рассмотрим теперь, как может применяться упрощенный макрооператор для преобразования моделей. Главный вопрос состоит в выборе предусловий для макрооператора.

Если бы мы ограничились применением полного плана, выраженного макрооператором, то очевидно, что в качестве предусловий следовало бы выбрать конъюнкцию предписаний в первом ядре ТТ. Однако в случае, если весь макрооператор неприменим, мы хотим использовать частичную возможность применения макрооператора, поскольку она тоже может привести к полезным преобразованиям модели. Таким образом, мы приходим к необходимости установления предусловий для каждого остатка макрооператора. В п. 7.3.1 мы установили достаточное условие применимости j-го остатка плана. Таким образом, мы должны установить истинность всех отмеченных предписаний в j-м ядре. Это может быть осуществлено с помощью процедуры сканирования ячеек ТТ.

Идея процедуры заключается в следующем. Чем меньше i и чем больше j, тем в большее число ядер одновременно входит ячейка $c_{i,j}$ ТТ. Таким образом, просматривая все ячейки $c_{i,j}$ в порядке возрастания i и убывания j и проверяя истинность всех отмеченных предписаний в этих ячейках доказательством их из текущей модели, мы будем последовательно находить те ядра в порядке убывания их номеров, все отмеченные предписания в которых истинны (если такие ядра имеются). Как только мы находим некоторую ячейку $c_{i,j}$, в которой хотя бы одно предписание не может быть доказано из текущей модели, мы исключаем из рассмотрения все ячейки $c_{k,j}$, $k \geq i$, так как при этом мы можем найти требуемое ядро с номером не более, чем i—1.

По мере доказательства высказываний в ячейках ТТ из текущей модели мы осуществляем соответствующие подстановки параметров макрооператора. Важно заметить, что сделанные для доказательства высказываний в одной ячейке подстановки должны немедленно распространиться на всю таблицу. Если имеются альтернативные возможности, то следует хранить дерево возможностей и, в случае необходимости, осуществлять возврат к точке ветвления.

С помощью описанной процедуры сканирования ТТ мы находим *j*-й остаток плана с наивысшим *j*, применимый к модели. В этом случае последовательность операторов $F_j, F_{j+1}, \ldots, F_n$ может быть выполнена, преобразуя текущую модель в некоторую новую модель. Если же неприменим весь план (т. е. ни один из *j*-х остатков плана неприменим, *j*==1,2,..., *n),* то конъюнкция отмеченных предписаний в первом ядре устанавливается как новая подцель.

Чтобы завершить изложение особенностей планирования с помощью макрооператоров, нам необходимо описать возможные подходы к решению задачи регулирования количества макрооператоров в системе. Хотя, казалось бы, богатый набор макрооператоров повышает возможности системы, наличие множества альтернативных решений задачи всегда будет в определенной степени снижать эффективность поиска плана. Конечно, трудно указать точно границы разумного компромисса между указанными факторами. Во всяком случае, есть два очевидных пути разумного сокращения количества макрооператоров в ПС.

Первый из них — исключение тех макрооператоров, которые являются частью более общих макрооператоров. Это исключение не только разумно, по и необходимо, поскольку ПС, работающая с макрооператорами, все время порождает подобные случаи.

Действительно, пусть решая задачу, ПС включает в план некоторую часть макрооператора или весь макрооператор. Получив полный план решения задачи, ПС обобщает его, порождая новый макрооператор, включающий в себя весь старый макрооиератор пли его часть. Если подстановки параметров в старом и новом макрооператорах соответствуют друг другу, то очевидно, что каждый список добавлений старого макрооператора является подмножеством некоторого списка добавлений нового макрооператора. В этом случае старый макрооператор может быть исключен из системы. Аналогичным образом следует поступать и в случае использования в новом макрооператоре частей старого, исключая, однако, при этом те списки добавлений нового макрооператора, которые являются подмножествами списков добавлений уже существующего макрооператора.

В общем процедура исключения избыточных списков добавлений формулируется следующим образом: если любой частный случай *j*-й шапки одной ТТ является также частным случаем некоторой последовательности операторов в этой же или любой другой ТТ, то все списки добавлений *j*-й шапки (т. е. начиная со 2-й и кончая (*j*+1)-й строкой) ТТ исключаются.

П р и м е р. Рассмотрим две последовательности операторов:

A: $F_1(p1)$, $F_2(p1,p2)$, $F_3(p3)$, $F_4(p3,C1)$, $F_1(p3)$, $F_2(p4,p5)$.

B: $F_3(p6)$, $F_4(p6,\mathrm{Cl})$, $F_1(p7)$, $F_5(p6,p7)$.

1) $F_1(p1)$ и $F_2(p1, p2)$ в последовательности *A* могут быть исключены, так как в конце плана есть $F_1(p3)$ и $F_2(p4,p5)$.

2) $F_3(p6)$ и $F_4(p6, \mathrm{Cl})$ в последовательности *B* могут быть исключены, так как в последовательности *A* имеются $F_3(p3)$ и $F_4(p3, C1)$.

3) $F_1(p7)$ в последовательности *B* не может быть исключено последовательностью *A*, так как $F_3(p6)$, $F_4(p6, C1)$, $F_1(p7)$ и $F_3(p3)$, $F_4(p3, C1)$, $F_1(p3)$ могут иметь разные частные случаи.

Следует отметить, что определенная выше процедура исключения не учитывает случая одинаковых операторов, включенных в разные макрооператоры, но имеющих различные поддержки предусловий. Вообще говоря, ни один из таких операторов не должен исключаться.

Второй путь сокращения количества макрооператоров в ПС — введение механизмов «забывания». Подход к решению этой задачи заключается в наборе статистики использования макрооператоров и исключении тех из них, которые используются реже, чем заранее установленный порог частоты использования.

7.4. Обобщение пространств поиска решений и планирование в абстрактных пространствах

7.4.1. Принцип образования иерархии пространств

Обобщение планов путем автоматического создания макрооператоров является важным шагом на пути к созданию универсальных и эвристически эффективных решателей коммуникационных задач, обладающих свойством самоусовершенствования. Однако накопление набора методов решения коммуникационных задач и укрупнение самих методов составляют лишь часть общей проблемы функционирования АКС. Мы отмечали, что решающим шагом на пути к созданию мощных АКС является глобальное исследование пространства поиска, имеющее своей целью «понимание» решателем

задач общих закономерностей этого пространства. Обращаясь к рассмотренной ранее задаче о миссионерах и людоедах, мы видим, что укрупнение операторов позволило нам лишь исключить некоторые промежуточные состояния. Наиболее существенный выигрыш был получен за счет введения в пространстве поиска решения некоторого комплексного образа, позволившего решить задачу на абстрактном уровне.

Главным недостатком метода обобщения планов (п. 7.3.2) является тот факт, что, укрупняя макрооператоры, мы оставляем другой элемент предписания — модели — на том же, весьма детальном уровне, что и в случае обычных оператором. В понятиях формальной модели, последовательное обобщение макрооператоров приводит к настолько большим конъюнкциям предусловий, составляющих первое ядро ТТ, и настолько большим спискам добавлений, что эффективность решения достаточно сложных задач может оказаться весьма малой.

Поэтому для планирования решения сложных задач в сложных областях необходимо описывать области в некотором существенно обобщенном пространстве, которое, значительно упрощая описание области, игнорировало бы незначительные детали. При этом полный процесс решения задачи можно было бы представить как пошаговый процесс в иерархии пространств от наиболее абстрактного к базовому пространству, в котором получалось бы окончательное решение. Существенной характеристикой такого процесса является поиск приблизительного наброска решения задачи в абстрактном пространстве и, при его достаточной перспективности, преобразование этого пространства в пространство более низкого уровня, уточнение наброска решения и т. д. Такой процесс должен значительно увеличивать эвристическую эффективность системы, в то же время допуская достаточно легкое преобразование предписаний от абстрактного к базовому.

Рассмотрим, каким образом приведенные выше соображения могут быть преломлены в контексте описания моделей области с помощью ппф в исчислении предикатов первого порядка и описания операторов с помощью предусловий, списков вычеркивания и добавления, также выраженных в виде ппф. Прежде всего возникает вопрос, каким образом следует отличать пространства друг от друга. Очевидно, что было бы целесообразно опускать часть описаний моделей и операторов. Однако нам необходим критерий «детальности» выражений, входящих в эти описания. Таким критерием может быть ранжирование фактов об области по степени их неизменности (см. п. 7.1.1.). Эвристическим основанием использования такого ранжирования являются следующие соображения. С точки зрения

построения плана существование предметов и их свойства (наличие комнат и ящиков, расположение дверей) являются более важными фактами, чем положение предметов, которые могут передвигаться отправителем и тем более, чем положение самого отправителя (отправитель не умеет строить двери и ящики!). Поэтому именно эти важные факты должны использоваться в абстрактном пространстве для частичных описаний модели, и после построения наброска плана он может наполняться такими подробностями, как положение предметов и т. д.

Второй вопрос заключается в том, какие из элементов описания моделей и операторов ранжировать по их неизменности. По-видимому, нет смысла разбивать на классы предписания, относящиеся к описанию модели: второстепенные детали в модели могут просто игнорироваться. Точно так же нет особого смысла ранжировать списки добавлений и вычеркиваний операторов: те из операторов, действие которых сводится к изменению деталей, не будут выбираться как пригодные в пространствах высокого уровня. Кроме того, изменения в списках вычеркивания и добавления приведут к тому, что в различных пространствах результаты действий операторов окажутся весьма различными, поэтому преобразования между пространствами могут оказаться сложными.

В то же время, ранжируя предусловия операторов, мы добиваемся весьма важного результата: в пространствах высокого уровня мы будем всегда выбирать те операторы, которые производят наиболее существенные преобразования моделей области.

Таким образом, мы выбираем предусловия оператора как средство различения уровня пространства поиска решения. Могут быть предложены различные методы присвоения весов литерам, входящим в предусловия операторов. Один из них заключается в следующем.

На все предикаты, описывающие область функционирования коммуникационной системы, накладывается некоторое частичное упорядочение, которое определяет порядок, в котором будут исследоваться литеры предусловий всех операторов, используемых в этой области. Исследование литер происходит следующим образом:

1) Всем литерам, чье значение истинности не может быть изменено никаким оператором, присваивается максимальный вес.

2) Оставшиеся литеры исследуются по порядку. Если литера может быть достигнута из состояния, где все ранее исследованные литеры истинны, с помощью короткого плана, она считается деталью и ей присваивается вес, равный ее рангу в частичном упорядочении. Если такого плана нет, то ей присваивается вес, больший, чем наивысший ранг в частичном упорядочении.

7.4.2. Планирование в иерархии пространств

Рассмотрим возможную схему процесса планирования в образованной взвешиванием лигер предусловии операторов иерархии пространств. Схема системы, приведенная на рис. 7.12, носит рекурсивный характер.

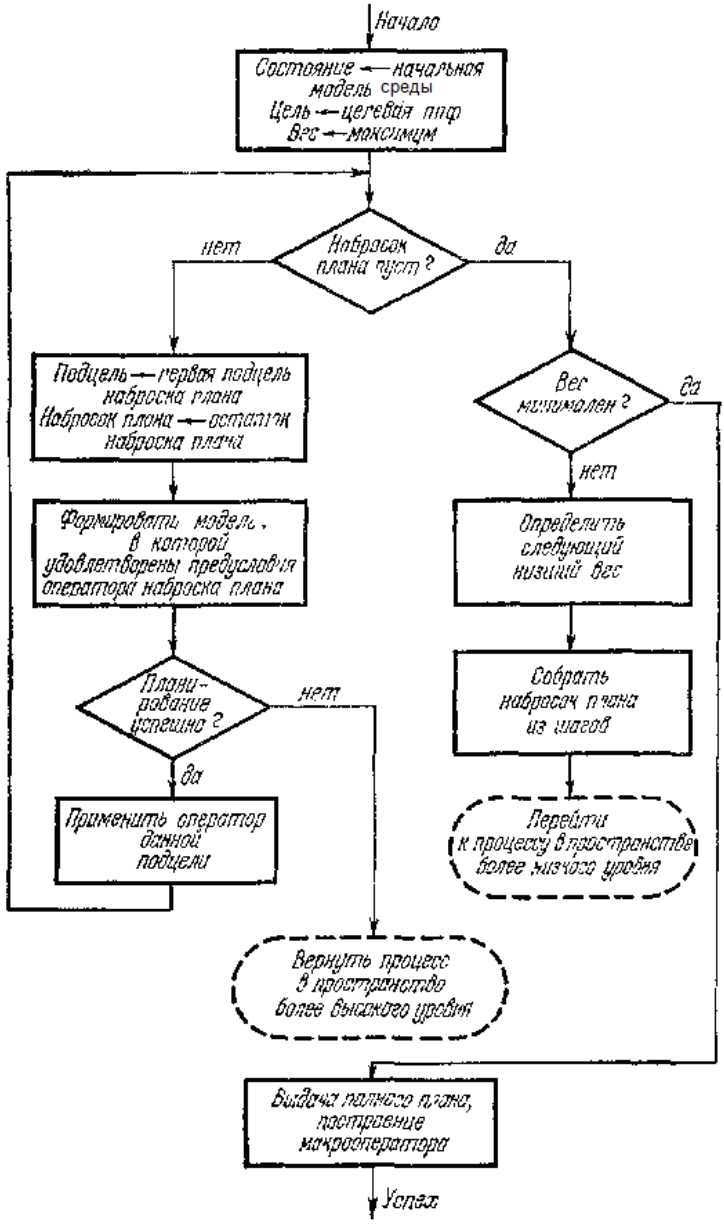


Рис. 7 12. Схема работы планирующей системы в иерархии пространств.

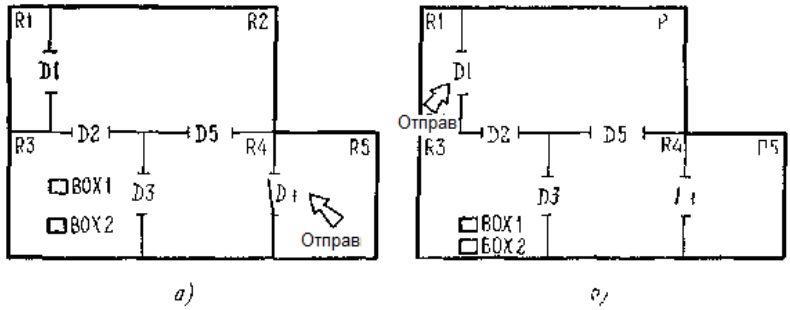
Сначала схема работает в абстрактном пространстве высшего уровня, строя набросок плана достижения главной цели. Когда этот план найден, система определяет вес следующего более низкого по уровню пространства, в котором требуется планирование, и строит скелет вершин, к которым применялись операторы на высшем уровне пространства. Затем система вызывается рекурсивно, решая подзадачи соединения шагов в наброске плана и обеспечения применимости этих шагов в соответствующих точках нового плана. Когда все подзадачи решены, система вновь вызывает себя рекурсивно для планирования в пространстве еще более низкого уровня. Этот процесс продолжается, пока не будет построен полный план в базовом пространстве, т. е. пространстве самого низкого уровня.

Описанный процесс осуществляет планирование в каждом абстрактном пространстве вплоть до главной цели, прежде чем осуществится переход к планированию на более низком уровне. Если какая-либо из подзадач в некотором пространстве не может быть решена, то управление возвращается в пространство более высокого уровня, вершина, послужившая причиной неудачи, исключается из рассмотрения и продолжается поиск успешного плана.

Заметим, что действия системы при неудаче аналогичны описанному ранее механизму возврата к точке ветвления со всеми его преимуществами и недостатками. В связи с этим требования к качеству поиска на высших уровнях иерархии пространств должны быть весьма высоки. Описываемая система должна управляться оценочными функциями, позволяющими строить допустимые стратегии в смысле приведенного ранее определения, возможно, в ущерб эффективности поиска, поскольку каждый лишний шаг в абстрактном пространстве может привести к большому количеству лишних шагов в пространстве более низкого уровня. В частности, на самом высоком уровне планирования используется оценочная функция при $\omega=1/2$ с постепенным увеличением веса эвристической функции по мере перехода в пространства более низкого уровня.

Проблема равнокачесгвенных решений в пространствах высокого уровня также становится весьма серьезной, поскольку произвольный выбор уменьшения различий может привести к неправильным решениям в пространствах более низкого уровня, где при выяснении деталей эти решения могут оказаться не эквивалентными, что приведет к необходимости возврата на более высокий уровень планирования. Поэтому при наличии равнокачественных альтернатив решение откладывается до планирования на низшем уровне путем частичной подстановки значений параметров оператора.

7.4.3. Пример. На рис. 7.13 изображена область отправителя в начальном и конечном состоянии.



а) *б)*

Рис. 7.13. Начальная (а) и конечная (б) область отправителя в примере (7.4.3).

Эта область достаточно сложна, однако работа системы может быть продемонстрирована только на примере повышенной сложности. В интересах краткости изложения мы не будем давать все формальные шаги построения плана, компенсируя эти пробелы словесным описанием. Мы хотим показать этим примером общий , характер планирования в иерархии пространств. Нам нужны следующие 5 операторов-

F_1: PUSHB *(bx, by)* — передвинуть *bx* к объекту *by*.

Предусловия:

{6} TYPE *(by,* OBJECT) *(by* типа «объект»),

{6} PUSHABLE *(bx) (bx* можно передвигать),

(∃ *rx*)[{5} INROOM (*bx*, *rx*) ∧ {5} INROOM *(by, rx)* ∧

{5} INROOM (Отправитель, *rx)},*

{1} NEXTTO (Отправитель, *bx)* (отправитель стоит вслед за *bx)*.

Вычеркивания:

AT (Отправитель, $), NEXTTO (Отправитель, $), AT (*bx*,$),

NEXTTO *(bx,* $), NEXTTO ($, *bx)*.

Добавления:

*NEXTTO(by, *bx)*, *NEXTTO(*bx, by)*,

NEXTTO (Отправитель, *bx)*.

F_2: GOTHRUDR *(dx, rx)* — идти через *dx* в *rx*.

Предусловия:

{6} TYPE(*dx*, DOOR) *(dx* типа «дверь»),

{6} TYPE(*rx*, ROOM) *(rx* типа «комната»),

{2} STATUS (*dx*, OPEN) (состояние двери «открыта»),

$(\exists ry)$ [{5} INROOM (Отправитель, *ry)* $\wedge$ {6} CONNECTS (*dx, ry, rx*)].
Вычеркивания:
AT (Отправитель, $), NEXTTO (Отправитель, $),
INROOM (Отправитель, $).
Добавления
* INROOM (Отправитель, *rx).*
F_3: GOTOD *(dx)* — идти к двери *dx.*
Предусловия:
{6} TYPE(*dx,* DOOR),
$(\exists\ rx)(\ \exists ry)$ [{5} INROOM (Отправитель, *rx*) $\wedge$
{6} CONNECTS (*dx, rx, ry)}.*
Вычеркивания:
AT (Отправитель, $), NEXTTO (Отправитель, $).
Добавления:
*NEXTTO (Отправитель, *dx).*
F_4: GOTOB *(bx)* — идти к объекту *bx.*
Предусловия:
{6} TYPE(*bx,* OBJECT),
$(\exists rx)$[{5} INROOM (bx, *rx)* $\wedge$
{5} INROOM (Отправитель, *rx*)]
Вычеркивания:
AT (Отправитель, $), NEXTTO (Отправитель, $).
Добавления:
* NEXTTO (Отправитель, *bx).*
F_5: OPEN *(dx)* — открыть дверь *dx.*
Предусловия:
{6} TYPE (*dx,* DOOR),
{5} STATUS (*dx,* CLOSED) (состояние двери «закрыта»),
{5} NEXTTO (Отправитель, *dx).*
Вычеркивания:
STATUS (*dx,* CLOSED).
Добавления:
*STATUS(*dx,* OPEN).

Примечания. 1. Числа, стоящие в фигурных скобках, соответствуют весу литерпредусловий.

2. Звездочки, стоящие перед добавлениями, указывают первичные добавления, т. е. те, которые рассматриваются системой при поиске пригодного оператора.

Мы не приводим здесь полного описания начальной модели. Часть этого описания очевидна из рис. 7.13, *а*. Заметим, что все предикаты типа TYPE, STATUS, CONNECTS, PUSHABLE входят в начальную модель. Кроме того, предполагается наличие двух аксиом:

$(\forall\ x)$ (PUSHABLE $(x) \rightarrow$ TYPE $(x,$ OBJECT)),
т. е. то, что можно передвинуть, является объектом, и
$(\forall\ x)$ (STATUS $(x,$ CLOSED)$\leftrightarrow\ \sim$ STATUS $(x,$ OPEN)),
т. е. если дверь закрыта, то она не открыта, и наоборот.
Мы решаем задачу: «пододвинуть ящик BOX1 к ящику BOX2, отправителю перейти в комнату R1». Целевая ппф имеет вид
NEXTTO (BOX1, BOX2) $\wedge$ INROOM (Отправитель, Rl).
Поиск начинается в пространстве высшего уровня (веса 6). Ищется различие между целью и начальной моделью. Различием является сама целевая ппф. Пригодным для уменьшения первого члена различия является оператор F_1 при bx=BOX1, by=BOX2. Поскольку предусловия веса 6 истинны в начальной модели, оператор применяется, и в новой модели BOX1 придвинут к BOX2. Различие между целью и этой новой моделью—INROOM (Отправитель, Rl). Пригодным для его уменьшения является оператор F_2 при rx=Rl. Его предусловия в пространстве веса 6
TYPE (R1, ROOM) $\wedge$ TYPE $(dx,$ DOOR) $\wedge$ $(\exists\ ry)$CONNECTS$(dx,\ ry,$ Rl)
удовлетворяются при dx=Dl. Применение F_2: GOTHRUDR (Dl, Rl) порождает модель, в которой удовлетворяется целевая ппф. Таким образом, мы получаем набросок плана в абстрактном пространстве высшего уровня в виде

PUSHB (BOX1, BOX2), GOTHRUDR (Dl, Rl). (7.7)

Производится переход к планированию в пространстве веса 5. Первая потдель — предисловия веса 5 первого оператора наброска плана, PUSHB (BOX1, BOX2). Различие между начальной моделью и предусловиями будет

INROOM (Отправитель, R3).

Пригодным для уменьшения этого различия является F_2 при rx=R3. Однако F_2 неприменим к начальной модели. Различиями являются INROOM (Отправитель, R2) или INROOM (Отправитель, R4). Для уменьшения первого различия уместен F_2 при rx=R2, однако он неприменим к начальной модели. Для уменьшения второго различия уместен F_2 при rx=R4 и он применим при dx =D4 Таким образом, применяется GOTHRUDR (D4, R4), формируя модель, к которой применим F_2 при rx=R3, dx=D3. Оператор GOTHRUDR (D3, R3) формирует модель, в которой удовлетворяют предусловия PUSHB (BOX1, BOX2) веса 5 Этот оператор применяется.
Тепечь устанавливается новая подцель — предусловия второго оператора (7.7). Сравнение текущей модели с его предусловиями вырабатывает различие INROOM (Отправитель, R2). Пригодным для уменьшения этого различия является оператор F_2 при rx=R2. Этот оператор применим при dx=D2, так что GOTHRUDR (D2, R2)

формирует новую модель, в которой подцель удовлетворена. Тогда применим GOTHRUDR (Dl, Rl), формируя модель, в которой вновь удовлетворяется целевая ппф. Мы получаем набросок плана в абстрактном пространстве веса 5:

GOTHRUDR (D4, R4); GOTHRUDR (D3; R3);
PUSHB (BOX1, BOX2); GOTHRUDR (D2, R2);
GOTHRUDR (Dl, Rl). (7.8)

Переходим к планированию в абстрактном пространстве веса 2. Первая подцель — предусловия веса 2 первого оператора (7.8). Сравнение их с начальной моделью вырабатывает различие STATUS (D4, OPEN). Пригодным для уменьшения этого различия является оператор F_5 при dx=D4, однако он неприменим. Различие между его предусловиями и начальной моделью — NEXTTO (Отправитель, D4). Пригодным для уменьшения этого различия является F_3 при $dx=D4$. Этот оператор применим к начальной модели и преобразует ее в модель, в которой применим и F_5. Поскольку, как легко проверить, все остальные подзадачи непосредственно удовлетворены, мы приходим к модели, в которой удовлетворяется целевая ппф с помощью наброска плана в абстрактном пространстве веса 2:

GOTOD (D4); OPEN (D4); GOTHRUDR (D4, R4);
GOTHRUDR (D3, R3); PUSHB (BOX1, BOX2);
GOTHRUDR (D2, R2); GOTHRUDR (Dl, Rl). (7.9)

Переходим к планированию в базовом пространстве (веса 1). Первые четыре оператора (7.9) применяются по очереди, однако пятый оператор неприменим. Различием является NEXTTO (Отправитель, BOX1). Пригодным для уменьшения этого различия является F_4 при bx =BOXl. Этот оператор применим, вырабатывая модель, в которой применим PUSHB (BOX1, BOX2). Окончательный план выглядит следующим образом:

GOTOD (D4); OPEN (D4); GOTHRUDR (D4, R4);
GOTHRUDR (D3, R3); GOTOB (BOXl);
PUSHB (BOX 1, BOX2);
GOTHRUDR (D2, R2); GOTHRUDR (Dl, Rl). (7.10)

Если внимательно проследить за ходом построения плана, легко заметить, что система автоматически строит сначала критические части плана, а потом переходит к построению плана для легко преодолеваемых областей базового пространства, действуя точно таким образом, как мы поступали на заключительном этапе анализа задачи о миссионерах и людоедах.

7.5. Стратегии выполнения действий

7.5.1. Исполнительные макрооператоры

Как указывалось в п. 7.1.2, ИС должна осуществлять выполнение планов и корректировку моделей области в соответствии с действительными результатами их выполнения. Очевидно, что стратегией, которая обеспечивала бы на каждом шаге наилучшее соответствие модели области самой области и плана его действительным результатам, была бы стратегия полного перепланирования после каждого шага выполнения. Однако мы отвергаем такую стратегию как крайне неэффективную. Настолько же неэффективной была бы и стратегия полного игнорирования результатов выполнения действия и их несоответствия «задуманному» плану.

Разумный компромисс между этими крайними альтернативами заключается в том, чтобы

1) при получении новой информации, обнаруживающей несоответствие *j*-го остатка плана, распознать ее и исключить из рассмотрения ИС этот остаток;

2) при неудаче *j*-й шапки плана с точки зрения ожидаемых результатов следует распознать неудачу и либо перевыполнить некоторую часть плана, либо осуществить перепланирование.

Такая стратегия обеспечит выполнение простых или составных планов (полных или неполных), т. е. явится основой для создания ИС класса А — Н, в зависимости от эффективности поиска планов с помощью ПС и сложности предъявляемых задач.

Рассмотрим воплощение указанной стратегии в системе, использующей макрооператоры. Мы предполагаем, что, построив план, система обобщает его и передает полученный макрооператор для выполнения.

Таким образом, первоначально мы должны осуществить частичные подстановки в макрооператоре, обеспечивающие доказательство целевой ппф. Важно отметить, что условия, при которых он может быть выполнен, остаются в обобщенном виде.

Алгоритм подготовки макрооператора для выполнения работает следующим образом:

1) В ячейку $c_{o,\ n+1}$ помещаются все предписания из начальной модели, которые были использованы в процессе построения плана при доказательстве целевой ппф.

2) Предписания из $(n+1)$-й строки используются для доказательства целевой ппф. Подстановки, произведенные в процессе этого доказательства, распространяются на весь макрооператор.

3) Предписания в $(n+1)$-й строке, представляющие собой поддержку доказательства целевой ппф, отмечаются. Полученный макрооператор готов для управления выполнением действий.

Рассмотрим пример из п.7.2. Обобщенная ТТ для этого примера приведена на рис. 7.9. Осуществляя шаги алгоритма, мы

1) помещаем в $c_{0,3}$ BOX (BOX1);

2) используем BOX (BOXl), INROOM (Отправитель, *p9)* и INROOM (*p6*, *p9)* для доказательства

G_o: $(\exists\ x)$ [BOX $(x) \wedge$ INROOM $(x,$ R1)];

3) осуществляем подстановки, произведенные во время доказательства, а именно *p6* =BOX1 и *p9*=*R1* во всей ТТ (рис. 7.9);

4) отмечаем предписания BOX (BOX 1) и INROOM (BOX1, Rl).

Подготовленная для выполнения ТТ приведена на рис. 7.14.

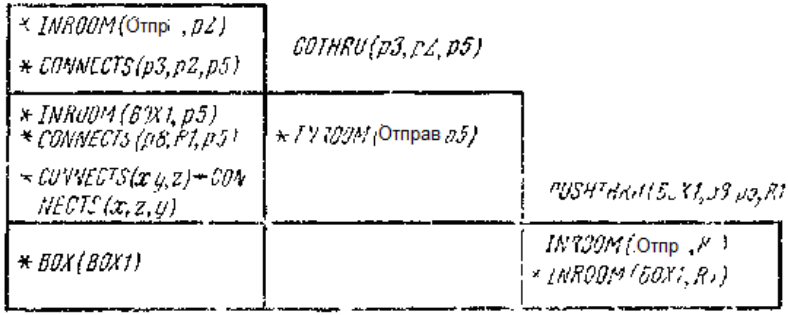


Рис. 7.14. Подготовленный для выполнения макрооператор (пример из п.7.2).

Полученный *макрооператор* мы будем называть *исполнительным.*

Управление выполнением плана с помощью исполнительного макрооператора основано на рассмотренном в п. 7.3.1 достаточном условии применимости *j*-го остатка плана. Таким образом, для продолжения выполнения действий на каждом шаге необходимо иметь хотя бы одно такое ядро, все предписания в котором были бы истинны. Заметим, что в начале выполнения плана ИС исходит из начальной модели, так что первое ядро содержит только истинные предписания. Однако в дальнейшем непредвиденные в плане обстоятельства могут либо неожиданно приблизить нас к цели, либо, наоборот, сделать план невыполнимым.

Для иллюстрации работы с исполнительным макрооператором мы предполагаем, что хотя бы часть плана может быть использована для выполнения (в противном случае должна начаться фаза полного перепланирования). Производится проверка ядер, начиная с конца, т. е. с ядра с наивысшим номером ((*n*+1)-е ядро — это (*n*+1)-я строка ТТ). Если все предписания в (*n*+ 1)-м ядре истинны, план выполнен. В противном случае мы проверяем на истинность предписаний *n*-е, (*n*—1)-е и т. д. ядра. Пусть первое ядро, все предписания которого истинны, имеет номер *j*. Тогда выполнены предусловия *j*-го остатка плана, F_j, F_{j+1} . . ., F_n. Мы применяем F_j и вновь ищем ядро с наивысшим номером, все предписания в котором истинны. В случае полного соответствия ожидаемых результатов планирования действительным, указанная процедура просто выполняет все операторы плана последовательно. Однако она имеет возможность устранить ненужные операторы и ликвидировать неудачи повторным выполнением операторов, например, путем других подстановок параметров в соответствии с изменившейся по каким-либо причинам моделью.

Это последнее обстоятельство представляется особенно важным, поскольку позволяет ИС осуществить фактическое перепланирование без обращений к ПС (при наличии альтернативных возможностей).

Очевидно, что процедура аналогична механизму возврата к точке ветвления, причем точка ветвления находится автоматически в соответствии с текущей моделью области.

Остается заметить, что проверка ядер на истинность всех предписаний, входящих в них, осуществляется с помощью процедуры сканирования, описанной в п. 7.3.3.

7.5.2. Обобщенные процессы планирования и выполнения

Как отмечалось в п. 7.1.2, постановка задачи совместной оптимизации процессов планирования и выполнения действий связана с обобщением этих процессов в единый процесс. Рассмотрим две возможности такого обобщения.

Первая из них связана с рассмотрением ПС как некоторой программы действий, которая оказывает воздействие не на внешнюю область, как это делает ИС, а на абстрактную среду, состояниями которой являются планы. Если оказывается возможным оценить полезность построенного неполного плана, то в каждом состоянии в пространстве планов ИС с помощью некоторого метаоператора может выбрать наиболее выгодное направление планирования. Более того, если оценки для

вновь вырабатываемого плана хуже оценок для уже выработанного плана (полного и неполного), то ИС осуществляет действие, а если лучше, то продолжает планирование. В такой модели описания процессов планирования и выполнения ИС представляет собой по существу метапланирующее устройство, осуществляющее на абстрактном уровне рациональный выбор между планированием и действием и выдающее заказы на планирование ПС. В свою очередь ПС играет роль генератора абстрактных квазидействий, формирующего по запросу из ИС некоторые, возможно, гипотетические планы с оценками их полезности. В этой трактовке ИС изменяет состояния модели и отбрасывает неподходящие для этой модели планы, а ПС, не меняя модели, расширяет существующий для нее план или создает новый план. **Формально действие и планирование могут быть описаны как функциональные отношения в пространстве, состояния которого объединяют состояния модели и плана.**

Другая возможность обобщения процессов планирования и выполнения связана с идеей планирования в иерархии абстрактных пространств. Экстраполируя эту идею от базового пространства, в котором строятся планы, на уровни еще большей детализации, вплоть до непосредственного управления подсистемами АКС, мы получаем полностью объединенного систему планирования и выполнения действий. Выполнение процесса в такой системе можно представить себе следующим образом

Базовое пространство ПС рассматривается как абстрактное пространство для ИС. Таким образом, уточняя детали, отсутствующие в этом базовом пространстве, общая система постепенно опускается до базового пространства ИС. Можно предполагать, что изменения в заданной области будут всегда в большей степени воздействовать на решения, принятые на низких уровнях иерархии пространств, чем на решения на более высоких уровнях. Тогда на каждом более высоком уровне план можно считать более близким к эффективному. Конечно, в такой трактовке взаимодействия планирования и выполнения действий требования к качеству решений на высоких уровнях иерархии существенно высоки. Перед началом погружения в пространства низкого уровня план должен быть максимально близок к эффективному. Однако важно отметить, что это может быть достигнуто не только за счет введения эффективных эвристик, но и за счет загрубления плана (путем исключения деталей). Стратегия выполнения основана на существенно неполном планировании. Выбирается небольшое число шагов плана как набросок плана действий, и он постепенно, по мере перехода к более низким уровням иерархии

пространств ИС, наполняется деталями становясь, возможно, при этом менее эффективным. На некотором уровне детали начинают вводиться только в начальную часть выработанного к этому моменту субплана и т. д. Наконец, вырабатывается короткий субплан в базовом пространстве ИС. Этот субплан выполняется. Расхождения между предполагаемыми на уровне субплана и действительными результатами наиболее вероятно являются деталями для построенного плана в базовом пространстве ПС и тем более для набросков плана на высших уровнях планирования. Тогда можно считать, что эти планы пока близки к эффективному.

Далее подобным образом строится новый субплан в базовом пространстве ИС, отражающий точно все появившиеся расхождения. В процессе построения нового субплана возможен возврат на тот уровень пространства, в котором это расхождение еще не является деталью.

В крайнем случае, когда расхождения оказываются достаточно серьезными, сигнал о неудаче распространяется точно до того уровня в иерархии пространств, в котором это расхождение является деталью.

Перепланирование начинается именно с этого уровня, оставляя наброски плана на высших уровнях неизменными.

Представляется, что использование подобных простых и коротких субпланов особенно эффективно в сложных областях, где степень неопределенности достаточно высока.

7.6. Особенности планирования при неполном описании области (пространства, среды)

Для описанных выше ПС в конечном счете нужна исчерпывающая детализация описания внешней области (пространства, среды). Очевидно, что для большинства реальных коммуникационных задач, с которыми может столкнуться отправитель (получатель), детализация описания области (пространства, среды) не может быть обеспечена. Кроме того, подход, связанный с точным запоминанием всех деталей окружающей обстановки и других, возможно, выведенных фактов, может оказаться неудовлетворительным. Во-первых, детальная информация может очень часто меняться. Во-вторых, в реальных мирах количество информации может оказаться существенно большим, чем в состоянии запомнить система.

Таким образом, наряду с дальнейшим развитием ПС, работа которых основывается на представлении всех фактов об окружающем пространстве, представляется необходимым поиск других подходов, в принципе основанных на неполном знании. Одним из таких подходов

является использование альтернативных планов, основанных на использовании составных и сложных (полных или неполных) планов Рассмотрим возможность построения таких планов в рамках ПС класса описанных в п.7.2. Расширим стандартное описание оператора с целью преобразования его в сложный оператор, т. е. оператор с несколькими выходами, каждому из которых приписана некоторая вероятность. В этом расширении каждый выход оператора будет описываться своими списками добавлений и вычеркиваний, а также указанной выше вероятностью

Типичным примером таких операторов могут служшь операторы сбора информации о состоянии мира, в частности, следующий оператор, проверяющий состояние двери (открыта или закрыта дверь):

CHECKDOOR (DX) — проверить дверь *Dx*.

Предусловия:

TYPE *(Dx*, DOOR); NEXTTO (Robot, DA:).

Выход 1	Выход 2
Список вычеркиваний:	Список вычеркиваний:
ПУСТО.	ПУСТО.
Список добавлений:	Список добавлений:
DOORSTATUS (D*x*, OPEN).	DOORSTATUS (D*x*, CLOSED).
Вероятность:	Вероятность:
0,5.	0,5.

Вообще говоря, в операторах сбора информации предусловия должны содержать совет о том, когда следует применять оператор (например, в виде требования, чтобы собираемая информация не была известна перед применением оператора). Такое требование гарантировало бы, что полученная оператором информация будет добавляться к модели, и, таким образом, позволило бы отличать оператор сбора информации от операторов, производящих сходные действия (в нашем примере— оператор CHECKDOOR от операторов типа F, в п. 6. 4. 3) Заметим, что такого рода требования представляют собой новый класс предусловий, так как требуют получения неудачи как в доказательстве ппф (дверь закрыта), так и в доказательстве отрицания ппф (дверь открыта), выражая, таким образом, неопределенность ситуации. Для таких предусловий следует определить свой синтаксис, а также добавить возможности получения подобных доказательств.

При использовании многовыходных операторов пространство поиска может быть представлено в виде пропозиционального графа, где вершины соответствуют состояниям окружающей среды, а каждая ветвь — выходу операторов, применяемых к вершине, причем выходы

из различных операторов образуют дизъюнктивные ветви, а выходы одного оператора — конъюнктивные Тогда план может быть представлен в виде поддерева, состоящего из начальной вершины дерева и, для каждой вершины плана, тех дочерних вершин, которые получены в результате применения выходов одного оператора. Каждая конечная вершина плана соответствует модели среды, удовлетворяющей целевой ппф (если план успешен).

Этот план является условным планом с ветвлением, поскольку в каждой вершине производится проверка, каков действительный выход ИС, и в соответствии с ним выбирается та или иная ветвь плана. При этом вероятность появления каждой вершины равна произведению вероятностей выходов операторов, определяющих путь из начальной вершины дерева к данной вершине.

Возникает вопрос: каковы должны быть условия окончания для алгоритма построения условного ветвящегося плана? Двумя крайними и потому тривиальными вариантами являются достижение абсолютного успеха (все конечные вершины удовлетворяют целевой ппф) и достижение абсолютной неудачи (ни одна из конечных вершин не удовлетворяет целевой ппф). Однако нас интересуют промежуточные случаи. Один из таких случаев будет иметь место, когда дерево полностью построено, и некоторые из его конечных вершин (но не все) удовлетворяют целевой ппф. Следовательно, можно в принципе выбрать такой выход каждого оператора в плане, чтобы план был успешен. Поскольку дерево полностью построено, то произойдет окончание работы алгоритма, и в качестве результата разумно выбрать такой план, для которого сумма вероятностей появления вершин плана, удовлетворяющих целевой ппф, будет максимальной.

Рассмотренные варианты являются полными планами. Для неполного плана необходимо иметь условия окончания, отличные от полного построения графа или достижения абсолютного успеха.

Можно принять за условие окончания нахождение такого плана, вероятность успеха которого превышает некоторый порог. Здесь вероятность успеха неполного плана определяется как сумма вероятностей вершин, построенных к этому моменту. Разумность такого критерия аргументируется исключением лишнего планирования в ситуации, когда план, близкий к успешному, уже найден.

При построении неполного плана, как обычно, возникает задача взаимодействия ПС и ИС. Вновь мы хотели бы иметь стратегию, позволяющую осуществлять продолжение планирования или перепланирование после каждого шага выполнения действий; однако такая стратегия слишком непрактична. Другими возможностями взаимодействия являются:

— вызов ПС, когда конечная вершина плана найдена, а задача не решена;

— вызов ПС, когда найдена вершина, ни одна из дочерних вершин которой не удовлетворяет целевой ппф.

Следует также включить в условия окончания и такие ясные случаи, которые должны останавливать построение плана, вероятность неудачи которого превышает некоторый порог. Это условие может быть выражено двояким образом:

1) когда каждый план в графе имеет вероятность достижения терминальной вершины, не удовлетворяющей целевой ппф, большую, чем некоторый предел;

2) когда каждая нераскрытая вершина в графе имеет вероятность, меньшую, чем некоторый малый предел.

Аргументом в пользу такого условия окончания является предположение, что ни один из планов, вероятно, не приведет к успеху.

При планировании целесообразно использовать стратегию выбора наиболее перспективного плана с последующим выбором в нем вершины для раскрытия. При этом в качестве оценок планов могут выступать вероятности успеха и неудачи для вершин плана и эвристические функции в принятом ранее смысле этого понятия. Оценки вершин могут основываться и на эвристических функциях, а также на вероятностях их появления.

Проиллюстрируем сказанное выше на примере (рис. 7.15).

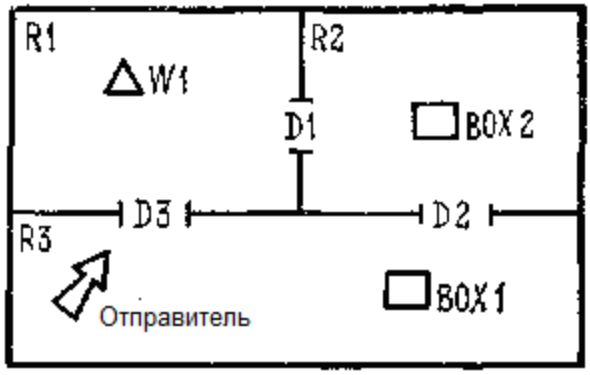


Рис. 7.15. Пример планирования при неполном описании пространства.

Задача отправителя состоит в том, чтобы передвинуть ящик BOX 1 в комнату R1 так, чтобы призма W1 и любой ящик не были в одной

комнате. Предположим, что отправитель не знает состояние дверей Dl, D2 и D3 (открыты они или закрыты) и не знает, есть ли призма W1 в комнате R1. План, использующий операторы GOTHRU и PUSHTHRU из примера в п.7.2 в сочетании с введенным выше оператором CHECKDOOR и новым оператором CHECKOBJECT (Wl, R1) (проверить, есть ли W1 в комнате R1), показан на рис. 7.16.

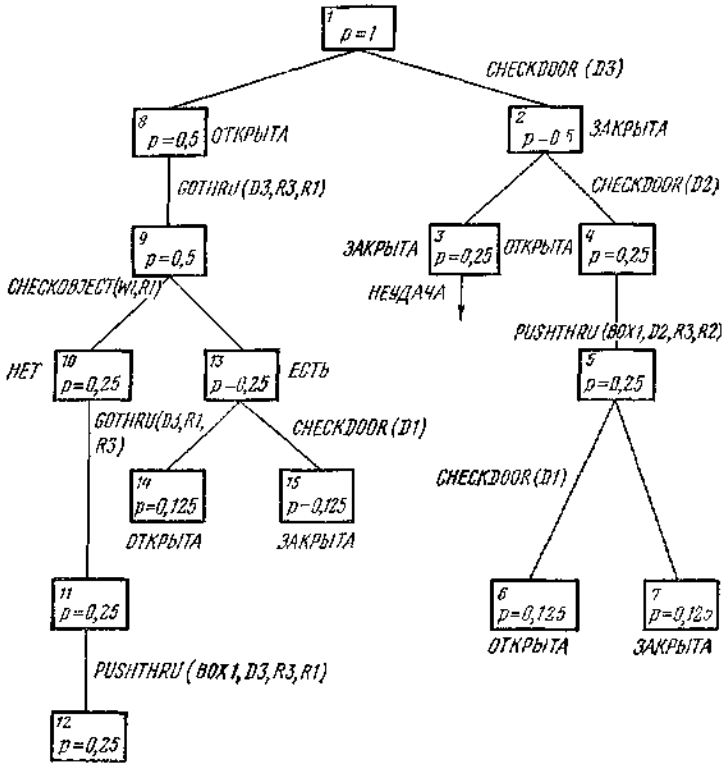


Рис. 7.16. Пример плана со сложными операторами.

Здесь предполагается, что планирование обрывается, когда все нераскрытые вершины имеют вероятность, меньшую 0,2. Номера в квадратах указывают последовательность раскрытия вершин. Мы предоставляем читателю возможность самостоятельно просмотреть этот план. Отметим только, что вероятность успеха для этого плана равна 0,25, вероятность неудачи — также 0,25, оставшиеся

возможные выходы не раскрыты, так как вероятности соответствующих вершин лежат ниже установленного порога.

Дополнительные возможности планирования с многовыходными операторами раскрываются при планировании в иерархии абстрактных пространств. Использование иерархического предписания может упростить построение условных планов, потому что неопределенность выходов операторов сохраняется лишь до определенного уровня детализации. В пространстве высшего уровня предусловия и действия операторов могут выражаться с помощью стандартного описания, т. е. без неопределенности. Однако некоторые из действий на этом уровне могут описываться в понятиях параметров, что усложнит преобразование планов в планы в пространствах низших уровней. Тем не менее простота представления сложных операторов делают эту схему перспективной.

7.7. Планирование в процедуральных предписаниях

Рассмотренный в предыдущих параграфах метод построения планов, основанных на декларативном предписании моделей в виде ппф в исчислении предикатов первого порядка и механизме редукции GPS в качестве стратегии поиска решения, обладает рядом достоинств. К числу их прежде всего относятся:

1) универсальность формализма, позволяющая решать задачи в значительном разнообразии миров;

2) возможность формального обобщения планов и использования макрооператоров как для исполнения построенного плана, так и построения других, более сложных планов;

3) возможность представления задачи в иерархии абстрактных пространств с последовательной детализацией первоначально построенных набросков плана.

В настоящем параграфе мы проиллюстрируем на примере некоторые особенности построения планов в процедуральном предписании, используя ПОЯ QA4.

7.7.1. Построение простых планов

Рассмотрим следующую задачу. Отправитель и ящик BOX 1 находятся в одной комнате. Требуется повернуть выключатель. Эта цель может быть достигнута, если отправитель пододвинет ящик к выключателю, встает на него, а затем повернет выключатель.

Для решения этой задачи нам потребуются следующие операторы (мы описываем операторы в декларативном предписании):
F_6: CLIMBONBOX *(bx)* — отправитель встает на ящик *bx*.
Предусловия:
TYPE (*bx*, BOX), ONFLOOR, NEXTTO (Отправитель, *bx)* (ONFLOOR — на полу).
Список вычеркиваний:
At (Отправитель, $), ONFLOOR.
Список добавлений:
ON (Отправитель, *bx*)
F_7: CLIMBOFFBOX (*bx*)— отправитель слезает с ящика *bx*.
Предусловия:
TYPE (*bx*, BOX), ON (Отправитель, *bx*).
Список вычеркиваний:
ON (Отправитель, *bx)*.
Список добавлений:
ONFLOOR.
F_8: TURNONLIGHT(*m*)— отправитель поворачивает выключатель *m.*
Предусловия:
TYPE (*m*, LIGHTSWITCH), ON (Отправитель, *bx*), NEXTTO (*bx*, *m)*.
Список вычеркиваний:
STATUS(*m*, OFF) *(m* выключен).
Список добавлений:
STATUS (*m*, *ON) (m* включен),
Операторы GOTOB *(bx)* и PUSHB(*(bx*, *m)* описываются аналогично операторам F_4 и F_1 соответственно (п. 7.4.3).
ПС STRIPS дает следующее решение указанной выше задачи:
GOTOB (BOXl); CLIMBONBOX (BOX1); CLIMB-
OFFBOX (BOXl);
PUSHB (BOX1, LIGHTSWITCH1); CLIMBONBOX
(BOX1);
TURNONLIGHT(LIGHTSWITCHl).

(7.11)

Заметим, что второй и третий операторы последовательности (7.11) реализуют взаимоисключающие действия. Эти действия были бы исключены при подготовке обобщенного плана в виде макрооперaюра к исполнению (п.7.5).
Рассмотрим теперь описание оператора TURNONLIGHT в ПОЯ QA4:
(LAMBDA (STATUS ← *M* ON)
(PROG (DECLARE *N)*
(EXISTS (TYPE $ *M* LIGHTSWITCH))
(EXISTS (TYPE ← *N* BOX))

(GOAL DO(NEXTTO $N $M))
(GOAL DO (ON (Отправитель $N)))
($ DELETE ('(STATUS $M OFF)))
(ASSERT (STATUS $M ON))
($BUILD('(:$ TURNONLIGHTACTION $M))))).
(7.12) ,

Общая структура довольно прозрачна, особенно для читателя, знакомого с языком LISP. Оператор выбирает выключатель и ящик и требует, чтобы ящик был поставлен рядом с выключателем, а отправитель встал на этот ящик. Затем осуществляются соответствующие вычеркивания и добавления, после чего сам оператор присоединяется к последовательности действий, выполняемых отправителем.

Оператор может быть применен для целей, сопоставляющихся образцу (STATUS ← *M* ON). Здесь ←*M* обозначает переменную, которая может быть сопоставлена любому предписанию, после чего *M* будет связана с этим предписанием. Если мы в соответствии с (7.11) хотим повернуть выключатель LIGHTSWITCH 1, то цель выражается как (STATUS LIGHTSWITCH I ON). Эта цель сопоставляется с образцом под знаком LAMBDA, и *M* связывается с LIGHTSWITCH 1. Таким образом, образец (STATUS ←*M* ON) играет роль списка добавлений оператора F_8.

Предусловия F_8,, выражены в форме предписаний под знаком EXISTS. Поиск выражений, сопоставляющихся с предписаниями EXISTS, производится в базе данных. Здесь $M означает, что переменная *M* может быть сопоставлена только с уже присвоенными *M* значениями. Таким образом, сопоставление с третьей строкой даст выражение (TYPE LIGHTSWITCH I LIGHTSWITCH). Этот факт, если он истинен, хранится в базе данных под знаком ASSERT. В общем, модель области в процедуральном представлении состоит из всех выражений ASSERT. Заметим, что в случае, если выражение, сопоставляющееся с EXISTS, не будет найдено в базе данных, то это вызовет сигнал о неудаче, включающий механизм возврата.

Второе выражение под знаком EXISTS будет истинным, если есть хоть один ящик. Для случая B0X1 переменная N будет связана с B0X1. Эта строка оператора выбирает ящик для последующих действий.

Следующее предусловие оператора выражено в виде цели (GOAL DO(NEXTTO $*N* $M)). Сначала производится проверка, нет ли выражения формы (NEXTTO $*N* $*M)* в базе данных (действие аналогично EXISTS). В случае положительного ответа цель достигнута и производятся соответствующие связывания *N* и *M* (заметим, что для этого, в силу префикса $, в базе данных должно быть выражение

(NEXTTO B0X1 LIGHTSWITCH1)). Если такого выражения в базе данных нет, то цель рассматривается как подцель. Управление ПС в процедуральном предписании осуществляется специальной управляющей программой, получающей цели и подцели и выбирающей в соответствии с некоторой семантической информацией и рекомендациями пригодные для достижения подцели операторы.

Оператор снабжен рекомендациями относительно того, какие операторы следует использовать для достижения подцели. В данном случае имеется два класса операторов: F_4 принадлежит к классу GO, а F_1, F_6, F_7, F_8 — к классу DO.

Поэтому для установленной подцели (GOAL DO (NEXTTO *$N $M*)) можно использовать один из четырех упомянутых операторов. Эти операторы пригодны, однако применимым из них будет тот, чья форма под знаком λ-выражения будет сопоставляться с выражением, стоящим под знаком GOAL. Приведем теперь запись оператора F_1 в языке QA4 (Мы используем слегка измененную формулировку оператора F_1, исключая из его предусловий TYPE (by, OBJECT) и включая предикат без параметров ONFLOOR.)*:*

```
(LAMBDA (NEXTTO←M ←N)
(PROG (DECLARE X)
(EXISTS (PUSHABLE $M))
(EXISTS (ONFLOOR))
(EXISTS (INROOM $M ←X))
(EXISTS (INROOM $N $X))
(EXISTS (INROOM Отправитель $X))
(GOAL GO (NEXTTO Отправитель $M))
(MAPC (QUOTE (TUPLE (AT (Отправитель ←X))
                    (AT ($M ←X))
                    (NEXTTO (Отправитель ←X))
                    (NEXTTO ($M ←X))
                    (NEXTTO ($X $M))))
                        $DELETE)
(ASSERT (NEXTTO $M $N))
(ASSERT (NEXTTO $N $M))
ASSERT (NEXTTO Отправитель $M)))
($ BUILD (' (:$ PUSHBACTION (TUPLE $M $N))))).
```

(7.13)

Очевидно, что оператор (7.13) применим для достижения подцели (GOAL DO (NEXTTO *$N $M)*), т. е. переменная *M* в (7.13) связывается с BOX1, а *N* с LIGHTSWITCH1. Оператор (7.13) устанавливает другую подцель класса GO, так что делается попытка применить F_4. Эта попытка будет успешной.

Вернемся к оператору (7.12). Третье его предусловие выражается в виде (GOAL DO (ON (Отправитель $*N*))). Эта подцель может быть достигнута с помощью F_6. Мы не будем в интересах краткости изложения описывать F_4 и F_6 в виде процедур QA4, поскольку общий стиль конструирования этих процедур, вероятно, достаточно ясен.

Мы также не будем описывать синтаксические подробности описания процессов вычеркивания и добавления в операторах (7.12) и (7.13). Они осуществляются выражениями под знаком DELETE и ASSERT соответственно. Последняя строка операторов, как указано в самом начале, присоединяет оператор с соответствующими подстановками параметров в общую последовательность действий, осуществляемых отправителем.

Следует обратить внимание на использование в описаниях операторов дополнительной информации, получаемой от пользователя. Об использовании классификации операторов в рекомендациях мы уже упоминали. Заметим, что выражения типа GOAL в (7.12) установлены в таком порядке, что отправитель не будет пытаться встать на ящик до того, как придвинет его к выключателю. Это позволяет избежать таких взаимоисключающихся последовательностей операторов, как в (7.11). Конечно, и в декларативной формулировке задачи можно было тоже наложить упорядочение на использование предусловий.

В настоящее время работы по созданию законченных ПС в процедуральных предписаниях находятся в начальной стадии, повторяя в значительной степени задачи того же класса, что и STRIPS .

7.7.2. Построение условных и циклических планов

Предположим, что нам задано множество связанных предикатов *P1, P2, . . ., Pn,* значения которых неизвестны во время планирования, однако могут быть установлены во время выполнения плана. Предикаты связаны в том смысле, что по крайней мере один из них должен принимать значение ИСТИНА во время выполнения плана. Тогда общая форма условного плана может быть представлена в виде

IF PI THEN Al ELSE
IF P2 THEN A2 ELSE

. . . . . .

IF *P* (*n*—1) THEN *A* (*n*—1) ELSE *An*.

Заметим, что *Pn* не проверяется на истинность, так как он должен проверяться только в том случае, если *P1, P2, . . ., P(n*—1) принимают значение ЛОЖЬ. Но в этом случае *Pn* должен быть истинным.

Один частный пример условного плана рассмотрен в п. 7.6.2 для случая многовыходных операюров, выходы которых управляются соответствующими вероятностями.

Рассмотрим более общий случай условного плана, где ветвления обуславливаются значениями предикатов *P1, P2, . . ., Pn,* истинность которых устанавливается некоторым источником действий, возможно, независимым от отправителя. Тем самым мы делаем первый шаг на пути к построению планов в динамической среде, одновременно демонстрируя некоторые характерные черты построения относительно сложных планов в процедуральном предписании. Поскольку при построении условных планов значения предикатов не определены, то следует составить такую программу, которая создавала бы последовательность локальных баз данных, или контекстов, в которых все большее количество предикатов получало бы определенные значения, а затем строить планы в этой последовательности контекстов. Эта программа, называемая в дальнейшем ALTPLAN (ALTPLAN (alternative plan) — альтернативный план), воспринимает в качестве аргументов предикат, или условие, и цель и в случае, если это условие истинно, указывает, что план не является условным. В случае, если условие не определено, ALTPLAN создает новый контекст, в котором условие ложно, решает задачу в этом контексте и выдает план решения в качестве результата.

Прежде чем перейти к описанию процесса планирования в ПОЯ QA-4, введем некоторые дополнительные определения, касающиеся этого языка.

Структуры данных. Мы будем использовать два типа структур — тупль (TUPLE) и множество (SET). TUPLE — это выражение формы (TUPLE *e*1. . .*en),* где $e_1, \ldots, e_n$— произвольные выражения, являющееся аналогом списка. Значение тупля есть тупль, элементами которого являются значения элементов исходного тупля. SET — это выражение формы SET (*e*l. . .*en),* причем порядок элементов и число вхождений одного и того же элемента безразлично. Например, SET *(ABC),* SET(*BCA*) и SET*(CAACB)* идентичны, Значением множества является множество, элементами которого являются значения элементов исходного множества.

Встроенные функции. Нам потребуется встроенная функция «равно» (EQUAL). Аргументом EQUAL является множество. EQUAL принимает значение ИСТИНА (TRUE), если значения всех элементов аргумента идентичны. Например, EQUAL (*A* $*X*) принимает значение TRUE, если *X* имеет значение *A*.

Переменные. Переменные — это идентификаторы, снабженные префиксами (нам потребуются префиксы ←, $, ⇐ и $$). Значения

первых двух префиксов объяснены в п. 7.7.1. В отличие от переменных с префиксами ←или $, сопоставляющихся одному элементу, переменные с префиксом ⇐ и $$ являются сегментными, т. е. могут сопоставляться некоторому фрагменту множества или тупля. В остальном префиксы ⇐ и $$ работают так же, как префикс ← и $ соответственно.

Функция квази-QUOTE обозначается ' и имеет формат ('*e*). Эта функция подставляет значения переменных в *e*, не вычисляя последнего. Например, если $X \Rightarrow 2$ (*X* имеет значение 2), то

('(PLUS $X 5)) ⇒ (PLUS 2 5), но не 7.

Операция DENY (отрицать) имеет формат DENY (синтаксическая форма) и присваивает значение FALSE (ЛОЖЬ) выражению с данной синтаксической компонентой, т. е. делает это выражение ложным в модели пространства.

Оператор SETQ имеет формат SETQ *(pe),* где *p*—образец, и сопоставляет образец с результатом вычисления выражения *e,* связывая переменные в образце с подвыражениями, которым они были сопоставлены.

Условные выражения. Мы будем использовать условные выражения IF (если) и ATTEMPT(пробовать).

Выражение IF имеет общую форму (IF *e*1. . .*en* THEN *e*'l . . .*e'm* ELSE *e*"l. . .*e"k)*. Выражения $e_1, \ldots, e_n$ вычисляются по порядку. Если значение последнего выражения e_n FALSE, то вычисляются по порядку $e"_1, \ldots, e"_k$ и значение $e"_k$ выдается как результат условного выражения IF. Если значение e_n отлично от FALSE, то вычисляются по порядку $e'_1, \ldots, e'_m$ и значением IF является значение e'_m. Например, значением выражения

IF (EQUAL $X 4) THEN (SETQ ← *Y3)* (PLUS $X *$Y)*
ELSE (SETQ←*Y* 6) (PLUS 4$ *Y*)

будет 7, если X=4, и 10 в противном случае. Выражение ATTEMPT имеет общую форму

(ATTEMPT *e1* . ..*en* THEN *e'1*. . . *e'm* ELSE *e"1* . . . *e"k)*.

Выражения $e_1, \ldots, e_n$ вычисляются по очереди, и если одно из них генерирует сигнал о неудаче, то управление передается в ELSE (если ELSE отсутствует, выдается сигнал FALSE). Если ни одно из $e_1, \ldots, e_n$ не выдает сигнала о неудаче, управление передается в THEN. Дальнейшие вычисления подобны вычислениям выражения IF.

Выражение RETURN (возврат) используется для выхода из программы PROG, имеет форму (RETURN *e)* и выдает в качестве результата программы выражение *e* с подставленными значениями переменной. Например, если BOX имеет значение BOX1, то результатом

программы по выходу RETURN (NEXTTO Отправитель $BOX) будет (NEXTTO Отправитель BOX1).

Выражение (CONTEXT PUSH CURRENT) [PUSH CURRENT — протолкнуть текущий] создает новый контекст, дочерний по отношению к текущему в дереве контекстов.

Связывание выражений с контекстом $CONTEXT осуществляется приписыванием к выражению конструкции WRT $CONTEXT (WRT (with respect to) — относительно).

Используя приведенные выше определения, построим программу ALTPLAN:

```
ALTPLAN(LAMBDA(TUPLE←CONDITION←GOAL)
          (PROG (DECLARE NC Y ALT Z)
          (ATTEMPT (EXISTS $CONDITION)
          THEN (RETURN (TUPLE)))
          (EXISTS (UNCERTAIN (SET $CONDITION←← Y)))
          (SETQ←NC (CONTEXT PUSH CURRENT))
          (DENY (UNCERTAIN (SET $CONDITION
                            $$Y)) WRT $NC)
          (DENY $CONDITION WRT $NC)
          (ATTEMPT (SETQ (SET←Z)$$Y)
          THEN (ASSERT $Z WRT $NC)
          ELSE (ASSERT (UNCERTAIN $Y) WRT $NC)
          (SETQ ←ALT (GOAL $GOALCLASS $GOAL WRT $NC)
          (RETURN $ALT))).
```

(7.14)

Первое выражение ATTEMPT проверяет, существует ли входное условие CONDITION в модели, и выдает в качестве результата пустой тупль (TUPLE), если это условие существует. Это означает, что альтернативный план не нужен. В противном случае EXISTS извлекает из базы данных множество неопределенных (UNCERTAIN) условий, связывая Y со всеми такими условиями, кроме входного. Затем программа создает новый контекст NC, отрицая в нем множество неопределенных условий как утверждения в текущем контексте. Кроме того, мы предполагаем, что в новом контексте NC условие CONDITION ложно. Второе выражение ATTEMPT проверяет, осталось ли в множестве неопределенных условий только одно такое условие. В случае, если условие одно, оно утверждается относительно нового контекста. В противном случае утверждается неопределенность меньшего множества условий. Затем программа решает задачу в новом контексте и выдает план.

Рассмотрим работу приведенной программы на следующем примере. Пусть отправитель хочет развлечься (HAVEFUN) и имеет для этого две

возможности: если идет дождь (RAINY), то отправитель идет в кино (MOVIE), если же светит солнце (SUNNY), он идет на пляж (BEACH). В результате мы хотим получить план вида

IF SUNNY THEN BEACH ELSE MOVIE. (7.15)

Запишем программы действий отправителя — BEACH и MOVIE.

```
BEACH (LAMBDA HAVEFUN
(PROG (DECLARE ALT) (SETQ ←ALT ($ALTPLAN SUNNY HAVE-
FUN))
 (IF (EQUAL (TUPLE) $ALT)
     THEN(RETURN BEACH)
     ELSE
     (RETURN ('(IF SUNNY THEN BEACH
     ELSE $ALT))))));
MOVIE (LAMBDA HAVEFUN
          (PROG (DECLARE ALT)
          (SETQI←ALT ($ALTPLAN RAINY HAVEFUN))
         (IF (EQUAL (TUPLE) $ALT)
          THEN (RETURN MOVIE)
           ELSE
          (RETURN ('(IF RAINY THEN MOVIE
           ELSE $ALT))))))).
```

Эти программы аналогичны с точностью до условий, поэтому мы опишем работу одной из них, например BEACH. Программа вызывает ALTPLAN, запоминая его в ALT. Если ALTPLAN выдаст пустой тупль (условие SUNNY истинно), то программа BEACH выдаст план BEACH. В противном случае ALTPLAN решает задачу в предположении, что SUNNY ложно, и программа BEACH выдает план IF SUNNY THEN BEACH ELSE $ALT.

Приведем протокол выработки условного плана (7.15) с помощью программ ALTPLAN, BEACH и MOVIE. Итак, отправитель хочет развлечься, так что общая цель может быть представлена в виде

(GOAL $HAVEFUN HAVEFUN), (7.16)

где переменная HAVEFUN указывает, что цель относится к классу целей (GOALCLASS) HAVEFUN. В нашем случае к этому классу относятся две программы действий — BEACH и MOVIE.

Общий план построения условного плана сводится к следующему. Общая цель вызывает, например, BEACH, которая вызывает ALTPLAN. ALTPLAN упрощает неопределенное условие и вновь вызывает BEACH. BEACH опять вызывает ALTPLAN. На этот раз ALTPLAN терпит неудачу, так что вызывается MOVIE. Поскольку условие RAINY истинно, план MOVIE передается через ALTPLAN в

BEACH, который вкладывает ее в общий условный план. Ниже приводится подробный протокол.

1. Ввод в базу данных (ASSERT (UNCERTAIN (SET RAINY SUNNY))).
2. Ввод в базу данных (GOAL $HAVEFUN HAVEFUN).

Примечание. Этот ввод осуществляется извне пользователем.

3. $HAVEFUN ⇒ (TUPLE BEACH MOVIE) — процедуры класса HAVEFUN.
4. HAVEFUN отсутствует в базе данных, неудача.
5. Вызов LAMBDA BEACH.
6. Вызов LAMBDA ALTPLAN.
7. Связывание переменной CONDITION с SUNNY, GOAL с HAVEFUN.
8. Проверка (EXISTS SUNNY), неудача.
9. Проверка (EXISTS(UNCERTAIN (SET SUNNY ⇐ Y))), связывание Y с RAINY.
10. Создание нового контекста NC.
11. Отрицание (UNCERTAIN (SET SUNNY RAINY)) относительно нового контекста NC.
12. Отрицание SUNNY относительно нового контекста NC.
13. Определение того, что осталось только одно условие RAINY, связывание Z с RAINY.
14. Утверждение RAINY в новом контексте.
15. (GOAL (TUPLE BEACH MOVIE) HAVEFUN) устанавливается относительно нового контекста.
16. HAVEFUN отсутствует в базе данных, неудача части EXISTS механизма GOAL.
17. Вызов LAMBDA BEACH.
18. Вызов LAMBDA ALTPLAN.
19. Связывание переменной CONDITION с SUNNY, GOAL с HAVEFUN.
20. Проверка (EXISTS SUNNY). В этот момент SUNNY ложно (см. п. 12), неудача.
21. Проверка (EXISTS (UNCERTAIN (SET SUNNY ⇐ Y))), это множество отсутствует в NC (см. п. 11), неудача, возврат к точке ветвления (п. 17)
22. Вызов LAMBDA MOVIE.
23. Вызов LAMBDA ALTPLAN.
24. Связывание переменной CONDITION с RAINY, GOAL с HAVEFUN.
25. Проверка (EXISTS RAINY), RAINY истинно в NC (п. 14).

26. ALTPLAN выдает (TUPLE), переменная ALT в MOVIE связывается с (TUPLE).
27. MOVIE выдает константу MOVIE, цель (п. 15) удовлетворена, переменная ALT в программе ALTPLAN принимает значение MOVIE, возврат к программе BEACH (п. 5).
28. Переменная ALT в BEACH принимает значение MOVIE.
29. BEACH выдает план (' (IF SUNNY THEN BEACH ELSE MOVIE)).

Перейдем теперь к рассмотрению процесса построения циклических планов. К сожалению, задача автоматического синтеза циклических планов, включающая в себя определение того, когда и какие части планов следует представлять в виде циклов, еще далека от своего решения. Мы покажем лишь, каким образом построить циклический план, если его цикличность задана явным образом.

Другими словами, если нам задан предикат *P* и действие *A,* содержащие одну и ту же свободную переменную *X,* то задача состоит в том, чтобы построить план, содержащий действие *A* для всех объектов в множестве, определяемом предикатом *P*.

Интуитивно ясно, что задача построения циклического плана может быть решена в три этапа.

Во-первых, необходимо иметь метод перечисления объектов, удовлетворяющих *P*. Во-вторых, необходимо выработать план, который в предположении об истинности *P* будет выполнять действие *A*. Наконец, следует ввести свободную переменную в *P* и *A,* которая к этому моменту является константой в QA-4, соответствующей связанной переменной как в *P,* так и в *A*, после чего объединить новые формы для *P* и *A* в общий циклический план.

Указанные этапы построения циклического плана реализуются в приводимой ниже программе LOOPPLAN (циклический план):

```
LOOPPLAN (LAMBDA (FA ←X (IF ←P THEN ←A))
                 (PROG (DECLARE NC BODY NV1 NV2)
                 (EXISTS (GENERABLE $P))
                 (SETQ←NC (CONTEXT PUSH CURRENT))
                 (ASSERT $P WRT $NC)
                (SETQ ←BODY (GOAL $GOALCLASS
                 $A WRT $NQ)
                 (SETQ (TUPLE ←NV1 ←NV2) ($GENVAR (TUPLE)))
                  (SETQ_←P(SUBST $P (TUPLE $X $NV1)))
                 (SETQ←BODY(SUBST $BODY (TUPLE $X $NV2)))
               (RETURN (REPEAT (GOAL: $FIND $P) (DO $BODY))))).
                                                        (7.17)
```

Эта программа требует ряда пояснений. Начнем с пояснений синтаксического характера Форма, стоящая под знаком LAMBDA,

содержит идентификатор FA(FA (for all) — для всех), читающийся как «для всех». SUBST — обычная функция языка LISP.

Выражение REPEAT (REPEAT —повторить.) имеет форму (REPEAT nde DO *e*l. . .*en)*, где *nde*— недетерминистическое выражение, $e_1, \ldots, e_n$ — выражения, и в нашем случае означает: «повторять выполнение программы $BODY для всех целей класса $FIND, сопоставляющихся образцу $P».

Программа действует следующим образом:

1) Устанавливает перечислимость объектов, удовлетворяющих *P* (GENERABLE $*P)*.

2) Создает новый контекст, утверждает в нем *P* и устанавливает цель *A* в этом контексте, вырабатывая план, выполняющий действия *A* для объектов, определяемых *P*.

3) После выполнения плана подставляет произвольную переменную вместо свободной константы в *P* и *A,* т. е. создает план-схему.

4) Выдает в качестве результата итеративный план выполнения действия *A* для всех целей класса $FIND, т. е. осуществляющих поиск объектов, сопоставляющихся со значением *P*.

Рассмотрим пример. Предположим, что отправитель должен перенести все ящики, находящиеся в комнате ROOM1, в комнату ROOM2, т. е. установим цель класса $DO:

(GOAL $DO(FA BOX (IF (INROOM BOX RCOM1)
THEN(INROOM BOX RQOM2))))), (7.18)

и мы хотим построить план достижения этой цели. Предположим, что к классу $DO относятся программа LOOPPLAN и программа, передвигающая ящики,— MOVEBOX. Произвольно предположим также, что поиск ящиков (класс $FIND) осуществляется программами LOCATE (обнаружить) и CHECKMAP (проверить по карте). Построение циклического плана может быть представлено следующим протоколом:

1 Ввод в базу данных (GOAL $DO (FA BOX (IF (INROOM (TUPLE BOX ROOM1)) THEN (INROOM (TUPLE BOX ROOM2))))).

2. $DO $\Rightarrow$ (TUPLE LOOPPLAN MOVEBOX).

3. Цель не найдена в базе данных, неудача.

4. Вызов LAMBDA LOOPPLAN.

5. Связывание переменных: *X* с BOX (свободная константа), *P* с (INROOM(TUPLE BOX ROOM1)), *A* с (INROOM (TUPLE BOX ROOM2)).

6. Проверка (EXISTS (GENERABLE (INROOM (TUPLE BOX ROOM1)))), устанавливается генерируемость.

7. Создание нового контекста NC.

8. Утверждение (INROOM (TUPLE BOX ROOM1)) в контексте NC.

9. Устанавливается (GOAL (TUPLE LOOPPLAN MOVEBOX) (INROOM (TUPLE BOX ROOM2))) относительно контекста NC.
10. Вызов LAMBDA MOVEBOX (программа MOVEBOX не приводится, предполагается, что ее образец (INROOM (TUPLE ←BOX ←ROOM)) сопоставляется с целью, переменная BOX связывается с BOX, переменная ROOM связывается с ROOM2, и программа выдает план (PUSHTO BOX ROOM2)).
11. Переменная BODY принимает значение (PUSHTO BOX ROOM2).
12. Переменная *P* принимает значение (INROOM (TUPLE ←*X* ROOM1)).
13. Переменная BODY принимает значение (PUSHTO *$X* ROOM2).
14. LOOPPLAN выдает результат (REPEAT (GOAL (TUPLE LOCATE CHECKMAP) (INROOM (TUPLE ←*X* ROOM1))) DO (PUSHTO $X ROOM2)).

7.8. Многоцелевое планирование

7.8.1. Общая постановка

В большинстве реальных применений коммуникаций необходимо решать одновременно множество коммуникационных задач, а не одну изолированную задачу. Очевидно, что такое планирование становится многоцелевым. В этой постановке следует формулировать цель для ПС в виде множества целевых ппф (выражений GOAL в процедуральном предписании), каждой из которых должен быть присвоен свой вес, или мера полезности. При этом мы должны допустить существование нежелательных целей, описывающих недопустимые состояния пространства, и присваивать им отрицательный вес. Кроме того, следует задать функционал, определяющий общую целенаправленность действий, и обеспечить возможность построения планов, которые минимизируют этот функционал. Таким образом, на каждом шаге планирования необходимо оценивать текущее состояние работы по достижению каждой из целей и принимать решения, над какой из целей работать, на основании вычисления функционала.
Поставленная таким образом задача может потребовать весьма сложных вычислений в каждом состоянии пространства. Кроме того, в общем случае, полезности различных целей должны быть функциями времени и общего состояния процесса планирования. Мы рассмотрим в настоящем параграфе две частные задачи многоцелевого

планирования, не требующие сложных манипуляций, — планирование с ограничениями и кооперация отправителей.

7.8.2. Планирование с ограничениями

Рассмотрим задачу планирования с двумя целями — основной и нежелательной. Мы должны построить план достижения основной цели, избегая при этом любого состояния пространства, удовлетворяющего нежелательной цели. К решению этой задачи можно подойти двояким образом: включить ограничения, представленные ппф нежелательной цели, в предусловия оператора или допустить выполнение операторов без ограничений, но распознавать недопустимые состояния в процессе планирования и направлять поиск по другому пути. Включение нежелательных целей в предусловия нарушает общность и гибкость использования операторов, не говоря уже о дополнительных сложностях, вводимых в механизм работы с предусловиями.

Поэтому мы пойдем по второму пути, принимая следующий план рассуждений. При выборе пригодного для уменьшения различия оператора производится проверка, не явится ли он сам причиной появления недопустимого состояния. Если предусловия оператора имплицируют ппф нежелательной цели, то нет смысла использовать этот оператор, поскольку, устанавливая его предусловия в качестве подцели, мы заранее обрекаем эту часть плана на неудачу. Если список добавлений сам содержит ппф нежелательной цели, оператор также должен быть отвергнут, так как он приведет пространство в недопустимое состояние.

Теперь остается случай, когда ппф нежелательной цели будет удовлетворяться конъюнкцией выражений, уже имеющихся в модели пространства и не вычеркиваемых оператором, и выражений из списка добавлений оператора. В этом случае недопустимое состояние распознается путем доказательства его из текущей модели пространства (после применения оператора). Далее должны быть выделены директивы, использованные в этом доказательстве. Отрицание одного из тех выражений, которые явились поддержкой ппф нежелательной цели, но не были добавлены оператором, вводится временно как дополнительное предусловие оператора. После этого подцель доказательства предусловий этого оператора пересматривается в свете добавлений. Может потребоваться пересмотр целой цепи предшествующих вершин плана (см. рис. 7.1), содержащих предусловия оператора в списках целей. Кроме того, если поддержка ппф нежелательной цели содержит несколько кандидатов, удовлет-

воряющих указанным выше условиям, создаются отдельные вершины для каждого из отрицаемых директив, которые добавляются в предусловия оператора.

Проиллюстрируем сказанное выше на примере (рис. 7.15). Пусть основная цель

$$\text{INROOM(R1, BOX1)} \vee \text{INROOM(R1, BOX2)}, \quad (7.19)$$

а нежелательная цель

$$(\forall x \forall y)[\sim (\text{INROOM } (rx, by) \vee \text{INROOM } (rx, \text{W1})], \quad (7.20)$$

т. е. ни в одной комнате не должны стоять одновременно ящик и призма W1.

Для построения плана мы используем операторы GOTHRU и PUSHTHRU, определенные в п.7.2. ПС типа STRIPS решает применить оператор PUSHTHRU (BOXI, D3, R3, Rl), однако это приводит к недопустимому состоянию. Поддержкой доказательства (7.20) являются директивы

$$\text{INROOM(Rl, W1)}, \quad (7.21)$$

$$\text{INROOM (Rl, BOX1)}. \quad (7.22)$$

Поскольку (7.22) получено в результате применения PUSHTHRU, то мы добавляем в предусловия конкретного случая применения оператора PUSHTHRU отрицание (7.21). Тем самым создается подцель извлечения призмы W1 из R1, например, в R2. Эта подцель может быть достигнута применением PUSHTHRU (Wl, Dl, Rl, R2), что приводит опять к недопустимому состоянию. Поэтому создается подцель извлечения BOX2 из R2 в R3. В результате будет построен план

GOTHRU (D2, R3, R2); PUSHTHRU (BOX2, D2, R2, R3);
GOTHRU(D3, R3, Rl); PUSHTHRU(W1, Dl, Rl, R2);
GOTHRU(D2,[1] R2, R3); PUSHTHRU (BOX 1, D3, R3, Rl).

(7.23)

Заметим, что в случае использования макрооператоров необходима проверка всех промежуточных состояний пространства, вырабатываемых компонентами макрооператора. В случае обнаружения недопустимого состояния следует либо запланировать его исключение перед использованием макрооператора либо пытаться обойти его имея по-прежнему в качестве цели результат макрооператора.

Некоторые проблемы возникают и при использовании обобщенного макрооператора для выполнения плана с ограничениями. Во время выполнения обобщенного плана мы должны быть уверены, что процесс подстановки не приведет к недопустимому состоянию. Более того, если даже частный случай плана выполняется в соответствии с планом, то всегда имеется опасность, что в процессе выполнения может появиться новая информация, которая не мешает достижению

основной цели, но вызывает недопустимые состояния пространства. Так, очевидно, что если отправитель (рис. 7.15) не знает, что INROOM (Rl, Wl), и обнаруживает наличие призмы, перемещая BOX1 в R1, необходимо перепланирование. Следовательно, ИС должна распознавать недопустимые состояния и использовать эту информацию для обращения к ПС.

Особый случай возникает, когда ограничения являются постоянным свойством пространства, и, следовательно, недопустимые состояния могут возникнуть при решении любой задачи в этом пространстве. Тогда может оказаться более эффективным построить список частных случаев операторов, применение которых при определенных условиях запрещено. ПС и ИС просто проверяют каждое применение оператора на допустимость, причем ИС исключала бы недопустимые операторы, обращаясь к ПС для перепланирования.

7.8.3. Кооперация отправителей

Назовем *кооперацией отправителей* совокупность нескольких операторов, объединенных общей планирующей системой, избегая при этом термина «многоцелевой отправитель» (этот термин может относиться и к отправителям, решающим одновременно несколько задач одним отправителем). Следует отличать понятия кооперации отправителей от нескольких независимых отправителей, каждый из которых снабжен своим решателем задач и может решить свою задачу вне кооперации с другими отправителями.

В противоположность этому, наши отправители, как мы предполагаем, решают кооперативно одну задачу. Представляется целесообразным выяснить возможность независимого выполнения действий по решению этой задачи каждым отправителем, т. е. каким образом можно распараллелить общий план на субпланы, выполняемые отдельными отправителями.

Если планы представляются ТТ (п. 7.3.1), то задача сводится к разбиению исходной ТТ на подтаблицы, относящиеся к различным отправителям, так, чтобы ИС могла независимо и параллельно управлять их действиями.

Рассмотрим ТТ, представленную на рис. 7.17.

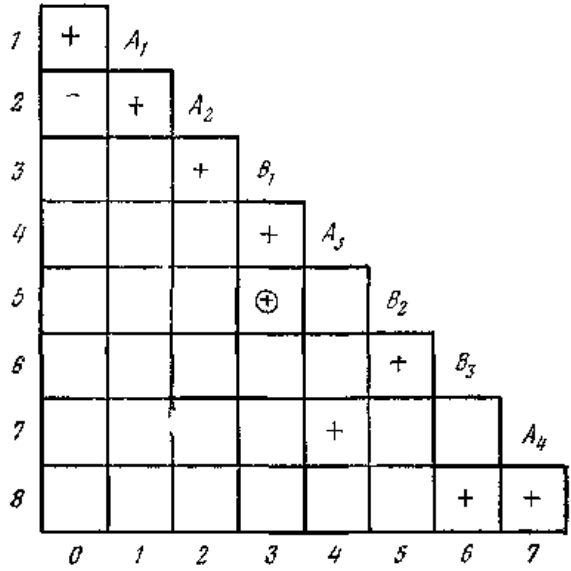


Рис. 7.17. Исходная треугольная таблица.

Здесь A и B — различные отправители, так что A_1, A_2, A_3, A_4— некоторые операторы, относящиеся к отправителю A, а B_1, B_2, B_3 — операторы, относящиеся к отправителю B. Знаком «+» обозначены отмеченные директивы, $\oplus$ в $C_{3,5}$ просто отличает две отмеченных директивы в 4-м столбце. Для каждого из отправителей все отмеченные директивы могут быть разбиты на три класса:

1) Результаты действия B_i, которые являются предусловиями A_j или частью списка добавлений TT (класс 1).

2) Результаты действия B_j, которые являются предусловиями B_j (класс 2).

3) Предусловия B_i, которые получаются как результат действия A_j или из начальной модели ТТ (класс 3).

Директивы класса 1 являются внешними выходами для подтаблицы отправителя B и поэтому они должны быть помещены в ячейки $c_{i,\ n+1}$ подтаблицы, n — количество операторов субплана отправителя B. Директивы класса 2 являются внутренними для подтаблицы отправителя B, и они должны быть помещены в ячейки $c_{i,\ j}$ подтаблицы. Наконец, директивы класса 3 являются внешними входами для подтаблицы отправителя B, и поэтому они помещаются в ячейки $c_{0,i}$,

подтаблицы. Аналогичным образом может быть построена и подтаблица для отправителя *A*.

Обращаясь к нашему примеру (рис. 7.17) и действуя, как описано выше, мы получим две подтаблицы для отправителя *A* (рис. 7.18, *а)* и отправителя *B* (рис. 7.18, *б)*.

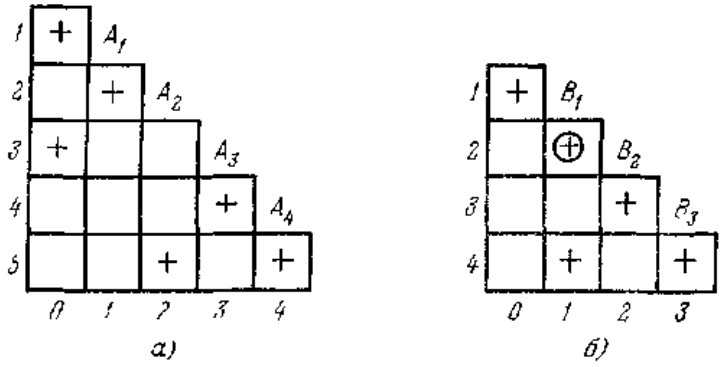


Рис. 7.18. Подтаблицы для отправителей *A* и *B*.

Рассмотренный алгоритм может быть легко обобщен на любое число отправителей последовательным разбиением ТТ на подтаблицы A_1 и $A_2, A_3, \ldots, A_n$, затем подтаблицы $A_2, A_3, \ldots, A_n$ на A_2 и $A_3, A_4, \ldots, A_n$ и т. д.

7.9. Особенности представления планов в динамическом пространстве

В п. 7.1.1 мы указали основные факторы, определяющие динамичность пространства: наличие независимых от отправителя источников действия и явная зависимость элементов модели пространства от времени. В связи с этим возникает вопрос: каковы адекватные средства представления такого пространства в ПС для отправителей? Наиболее естественным решением этого вопроса является описание пространства в виде совокупности параллельно идущих, независимых, равноправных и взаимодействующих процессов, каждый из которых может некоторым образом влиять на пространство, изменяя его. Тогда ПС АКС, работающей в динамическом пространстве, должна быть снабжена специальным процессором, в функции которого входит управление процессами, в частности их инициацией, прерыванием и

завершением, и обновление модели пространства в соответствующие моменты времени.

В общих чертах работа такого процессора может быть описана следующим образом. Взаимодействуя с моделью пространства, процессор определяет условия, необходимые и достаточные для инициации тех или иных процессов. Если процесс должен быть инициирован, то процессор запускает его, определяет необходимые связки для переменных процесса, после чего следит за условиями его прерывания или естественного завершения, которые определяются другими процессами, находящимися под управлением процессора. В случае прерывания или естественного завершения процесса процессор производит изменения в модели пространства, соответствующие достигнутым процессом результатам его действия. Взаимодействие процессора с процессами и моделью пространства осуществляется в специальном модельном времени аналогично тому, как это делается в языках моделирования.

Результатом работы процессора является план взаимодействия процессов, необходимый для достижения заданной цели.

Описанная система может быть реализована как в декларативном, так и в процедуральном предписаниях. В качестве процессора может выступать специальная мониторная программа. В настоящем параграфе мы рассмотрим особенности построения планирующей системы, работающей в динамическом пространстве и основанной на базовых представлениях системы STRIPS. Затем мы опишем процедуральный подход к построению децентрализованной системы взаимодействующих процессоров — акторов, основанный на двух основных принципах — неразделимости потока данных и управляющей информации и полной локализации всей обработки информации, включая управление взаимодействием акторов, в самих акторах.

Рассмотрим систему, состоящую из трех основных частей: **модели пространства, множества описаний, или сценариев процессов, и упомянутой выше мониторной программы.** В соответствии с нашими представлениями о ПС, работающих в динамическом пространстве, мониторная программа включает в себя динамически изменяемое множество блоков управления процессами (БУП). Каждый БУП описывается обращением к определенному **сценарию процесса и множеством связок параметров процесса**. Заметим, что если одновременно протекает несколько идентичных процессов, то для каждого из них создается свой БУП, однако с обращением к одному и тому же сценарию. Таким образом, мониторная программа децентрализует управление идущими процессами и, в то время как

БУП производят соответствующие сценариям изменения в модели пространства, другая часть мониторной программы определяет возможность запустить процессы, а также ожидает от БУП сигналов о прерывании или естественном завершении процесса и в этом случае ликвидирует соответствующий БУП. Эту часть монигорной программы назовем *супервизором.* Опишем структуру и дадим примеры сценариев процессов, а затем более подробно в понятиях сценариев процессов опишем ряд функций мониторной программы. При этом будем рассматривать два основных класса процессов.

Первый класс процессов — это процессы, происходящие мгновенно во времени (с точки зрения ПС). Такие процессы можно описывать с помощью стандартной формы операторов системы STRIPS, т. е. **наименования, предусловий и результатов в виде списков добавлений и списков вычеркиваний**. Имеется лишь два отличия. Во-первых, поскольку мы представляем динамическое пространство как совокупность равноправных процессов, то в сценариях, управляющих отправителем, необходимо включать в предусловия указание о процессе выбора отправителем данного сценария из множества альтернатив. Будем задавать это указание в виде (SELECT *rf*), где *r* — отправитель (в общем случае — один из представителем, так чго *r* является параметром), *f* — сценарий процесса.

Указание предусловия SELECT делает всю совокупность предусловий необходимыми и достаточными условиями для начала процесса, в отличие от предусловий операторов STRIPS, где предусловие определяют только необходимые условия (отправитель может выбрать данный оператор, но не обязан этого делать). Таким образом, как только предусловия сценария процесса удовлетворены, процесс запускается.

Во-вторых, **допустимы два вида предусловий — предусловия, выраженные в исчислении предикатов, или символические предусловия,** которые мы использовали при рассмотрении системы STRIPS, **и предусловия, выраженные в виде отношений, заданных в области действительных переменных, или аналитические предусловия**. Аналитические предусловия необходимы для обеспечения взаимодействия в общем плане мгновенных процессов и процессов второго класса.

Второй класс процессов — это процессы, явно зависящие от времени (в том числе, непрерывные), т. е. процессы, которые воздействуют на модель пространства постепенно. В сценариях таких процессов могут присутствовать результаты действий, выраженные как отношения и функции, заданные в области действительных переменных и принимающие значение также в области

действительных переменных (в отличие от списков вычеркивания и добавлений в обычных результатах действий). Для таких процессов мы будем различать **символические результаты, т. е. выражаемые в исчислении предикатов, и аналитические результаты, выраженные в виде указанных функций и отношений**. Кроме того, для процессов, моделирующих постепенные изменения пространства, мы должны задавать условия их продолжения, нарушение которых приводит к их естественному завершению или прерыванию процесса мониторной программой. Эти условия также могут быть символическими и аналитическими.

Рассмотрим следующий пример: «отправитель должен повернуть кран и наполнить ведро водой». Введем сценарии процессов поворота крана (TURNVALVE) и наполнения ведра (FILLBUCKET).

Наименование сценария: TURNVALVE (r, v, V^c) — отправитель r поворачивает кран v так, чтобы установилась скорость потока V^c.

Предусловия:

символические: (SELECT r TURNVALVE (r, v, V^c));
(V_{max} v V^c_{max}); (AT r n); (AT v n);

аналитические: $0 \le V^c \le V^c_{max}$.

Результаты-М:

вычеркивания: $(V\ v\ \$)$;

добавления: $(V\ v\ V^c)$.

Процесс, определяемый сценарием TURNVALVE, является мгновенным. В связи с этим его результаты, обозначенные буквой «М», представляют собой обычный список вычеркиваний и добавлений. Значение $, как и во всех определениях операторов, безразлично. Все параметры, связанные со скоростью потока (V^c, V^c_{max}), принимают численные значения; индекс наверху означает, что переменные связываются при запуске процесса и остаются в течение процесса неизменными.

Наименование сценария: FILLBUCKET (b, v) — ведро b наполняется из крана v.

Предусловия:

символические: $(V\ v\ V^c)$; (AT v n); (AT b n);
(ORIENTATION b UP); $(C_b\ b\ C\%)$\
$(C\ b\ C_0)$;

аналитические: $0 < V^c$;
$C_0 < C_b$

Результаты-П:

символические: $(C\ b\ C^Y)$;

аналитические: $C^Y = C_0 + V^c \cdot \tau$.

Условия продолжения:

символические: $(V \vee V^c)$; $(AT\ b\ n)$;

(ORIENTATION *b* UP);

аналитические: $C^Y < C_b$.

Этот сценарий определяет постепенно изменяющий свои результаты процесс. В связи с этим его результаты снабжены буквой «П» и имеют символическую и аналитическую часть. Через С обозначено содержимое ведра, C_o— начальное содержимое, C_b— емкость ведра. Индекс «Y» у переменной *C* означает, что эта переменная изменяется самим процессом (в отличие от переменных с индексом С). Таким образом, символическая компонента результатов означает, что в результате процесса содержимое ведра станет C^Y, а аналитическая компонента — что C^Y определяется через начальное содержимое, скорость потока и длительность процесса τ. Предикат (ORIENTATION *b* UP) означает, что ведро должно быть поставлено отверстием вверх.

Условия продолжения указывают, что процесс может быть прерван, если изменится скорость потока, ведро не будет находиться под краном или будет перевернуто. Процесс придет к естественному завершению, если ведро наполнится $(C^Y \geq C_b)$.

Заметим, что, поскольку процесс наполнения независим от отправителя, в его предусловиях отсутствует SELECT.

Рассмотрим теперь работу супервизора мониторной программы на примере наполнения ведра.

Каждый раз, когда супервизор обнаруживает, что предусловия процесса удовлетворены, он создает БУП и записывает в него все значения параметров, при которых удовлетворяются предусловия процесса, и время начала процесса τ_0. Далее, рассматривая символические компоненты предусловий и результатов-П, супервизор определяет отношения, которые будут изменяться процессом. В нашем случае таким отношением является (С ВКТ C_0), где ВКТ — конкретное ведро (значение параметра *b*). Все такие отношения удаляются из множества символических отношений в модели пространства и добавляются к множеству аналитических отношений модели пространства с заменой значения на переменную с индексом *Y*. В нашем примере будет удалено (С ВКТ C_o) и добавлено (С ВКТ C^Y). Каждое из таких новых аналитических отношений связывается указателем с БУП, который моделирует процесс, определяющий *Y*-переменные и отношения, в данном случае FILLBUCKET. Таким образом, процесс запущен. Теперь супервизор должен обеспечить наблюдение за условиями продолжения. Оно осуществляется различным образом для символической и аналитической компонент условий.

С каждым отношением символической компоненты условий продолжения связан список указателей к тем БУП, которые

моделируют процессы, продолжение которых зависит от этого отношения. Как только это отношение вычеркивается из модели пространства, производится прерывание всех процессов, чьи БУП были связаны с этим отношением.

При создании БУП супервизор конструирует систему уравнений из аналитических компонент результатов — П и условий продолжения. Из этой системы могут быть определены условия прерывания процесса относительно одной из Y-переменных или времени. В нашем примере создается система

$$\begin{cases} C^Y = C_0 + V^c \cdot \tau, \\ C^Y < C_b, \end{cases}$$

из которой может быть определен критический момент естественного завершения процесса $t = \tau_0 + \tau$, где

$$\tau = \frac{C_b - C_0}{V^c}.$$

Время t записывается в БУП и может быть использовано супервизором.

Рассмотрим, как происходит прерывание при нарушении символического отношения $(V\ v\ V^c)$. Это условие может быть нарушено процессом TURNVALVE. Если скорость потока изменилась до нуля, то отношение $(V\ v\ V^c)$ вычеркивается из символической части модели пространства. Супервизор проверяет список указателей этого отношения, находит указатель к БУП FILLBUCKET и прерывает этот процесс. При этом аналитическая компонента результатов должна быть преобразована в символические отношения в модели пространства. Исходя из времени прерывания τ, вычисляется C^Y и вводится символическое отношение $(C$ ВКТ $C^Y_\tau)$, где C^Y_τ — значение C^Y в момент τ. Затем БУП FILLBUCKET ликвидируется.

Если скорость потока изменяется до некоторой новой положительной величины, то это изменение также прервет FILLBUCKET, но в данном случае он будет вновь запущен с другим параметром V^c и новым начальным содержимым C'_0, так как все его предусловия будут удовлетворены.

Нам осталось рассмотреть работу супервизора со временем. Как указывалось выше, с каждым БУП связано время прерывания, т. е. число, определяющее самый ранний момент времени, в который процесс должен быть прерван. Супервизор определяет самый ранний из таких моментов для всех действующих процессов, например T_0. Далее супервизор ищет среди всех сценариев процессов такой, который при некоторых значениях его параметров мог бы быть запущен раньше, чем в момент T_0. Если такого сценария нет, то супервизор

устанавливает модельное время в T_0 и выполняет прерывание. Если же таких сценариев несколько, то супервизор находит тот из них, который должен быть запущен раньше других, соотносит модельное время моменту запуска этого процесса, создает для него БУП и переходит к запуску следующего по времени процесса.

Описанный механизм осуществляет моделирование параллельных во времени, независимых и взаимодействующих процессов и действий. Очевидно, что **над этим механизмом следует надстроить планирующую систему, которая могла бы на основе поставленной цели и начальной модели пространства определять последовательность совокупностей процессов, приводящую к цели оптимальным в некотором смысле образом.** Основной для построения такой ПС в значительной степени могут служить обычные ПС типа STRIPS. Однако наличие аналитических компонент предусловий, результатов и условий продолжения приводит к серьезным осложнениям. В частности, главной проблемой является необходимость решения систем уравнений, которые могут быть весьма сложными. Поскольку мы хотели бы, чтобы ПС была достаточно универсальной, она должна быть в этом случае снабжена решателем таких систем, обладающим весьма общими (а следовательно, и слабыми) методами, что существенно снизит эффективность ПС

Выходом из положения может быть моделирование человекоподобного поведения. Действительно, человек, осуществляя одновременно несколько действий в реальном мире, как правило не решает систем уравнений для точного определения моментов времени или переменных процессов, соответствующих завершению или прерыванию этих процессов. Хорошая хозяйка, готовя обед из нескольких блюд, может заниматься уборкой квартиры и делает это, изредка окидывая взглядом все поле своей деятельности, а отнюдь не производя аналитических выкладок.

Мы приходим к выводу о важнейшей роли взаимодействия процессов восприятия, приблизительной оценки и планирования действий в динамическом пространстве. Однако эта задача и связанная с ней задача планирования в динамическом пространстве еще далеки от своего решения.

В настоящее время развивается несколько другой подход к планированию и выполнению действий в динамическом пространстве. Главной особенностью его является децентрализованное взаимодействие специальным образом построенных процедур, или скелетов (фреймов). Этот подход представляет собой вполне естественное развитие процедуральных предписаний в направлении локализации знаний и решения задач в процедурах.

Мы опишем основные идеи, связанные с построением одной из наиболее перспективных, на наш взгляд, реализаций скелетов (фреймов)— **системы взаимодействующих акторов**. Взаимодействие между акторами выражается в передаче предписаний (директив), и это единственный вид внешнего поведения акторов (внешней коммуникацией).

Внутреннее поведение акторов (внутренняя коммуникация) определяется рядом механизмов:

— целью, которая проверяет удовлетворимость всех начальных условий актора, которому передано предписание (директива);

— рядом процедур, обеспечивающих связывание переменных и присвоение последним соответствующих значений;

— синхронизатором, определяющим, когда в действительности должен начать работать актор после передачи ему предписания (директивы), а также предписывающим такой порядок действия акторов, при котором удовлетворяются их цели;

— монитором, воспринимающим все предписания (директивы), переданные актору, и определяющим порядок их обработки;

— «банкиром», управляющим распределением ресурсов памяти и времени для актора.

Следует отметить, что каждый из указанных механизмов определен локально и работает только при вызове актора, что **обеспечивает максимальную степень параллелизма**. Кроме того, локализация, как уже отмечалось ранее, обеспечивает решение проблемы границ.

Рассмотрим схему передачи предписания (директивы) актору. Мы должны различать при этом актор, **посылающий** предписание (директиву) (**отправитель, источник**), и актор, **воспринимающий** его (**получатель, адресат**). В структуру предписания (директивы) входит собственно предписание (директива) *M* и продолжение процесса *C,* которое должно быть активировано после выполнения действий получателем (отсутствие продолжения означает, что отправитель закончил свою работу). Итак, если отправитель *R* посылает получателю *T* предписание (директиву) *M* с продолжением *C,* то схема передачи предписания (директивы) может выглядеть следующим образом:

1) Вызов «банкира» *R,* санкционирующего расходы ресурсов отправителем.

2) Посылка предписания (директивы) «банкиром» *R* синхронизатору *T.*

3) Передача предписания (директивы) от синхронизатора T мониторам *T.*

4) Посылка предписания (директивы) мониторами *T* к целям *T.*

5) Цели *T* посылают предписание (директиву) *M* продолжениям *T.*

6) Продолжение *T* осуществляет некоторую работу (в том числе передачу предписания (директивы) следующему получателю или, в случае успешного завершения работы, возврат к продолжению C).

Следует отметить ряд особенностей такой схемы.

Во-первых, передача предписания (директивы) актору-получателю сопровождается созданием нового актора, являющегося частным случаем получателя, управляемым предписанием (директивой). В частности, это предполагает повторное использование старого актора, если таковой окажется.

Во-вторых, передача предписания (директивы) включает в себя все необходимые управляющие функции, свойственные процедуральному предписанию, например, вызовы функций, повторения, вызовы параллельных процессов и т. д.

В-третьих, общение между акторами осуществляется децентрализованно, т. е. без каких-либо посредников.

В-четвертых, полная локализация позволяет, за счет отсутствия боковых эффектов, осуществлять одновременное общение актора с несколькими другими.

В-пятых, передача предписаний (директив) носит полностью ненаправленный характер, поскольку не предполагает заранее, что продолжение отправителя когда-либо получит предписание (директиву) (требуемый процесс может быть осуществлен другим способом или отвергнут как неудачный).

В заключение отметим ряд особенностей, связанных с накоплением и реорганизацией знаний в формализме акторов. Эти особенности в большой степени относятся к процедуральному предписанию вообще.

Общей проблемой для коммуникационной системы является добавление информации так, чтобы уже накопленная информация могла остаться без изменений (конечно, если нет необходимости в изменениях). В предписаниях в логических исчислениях мы в большинстве случаев легко делаем это путем добавления аксиом. Процедуральные механизмы не позволяют делать этого так легко. Поэтому очень важно развивать гибкие механизмы накопления процедуральной информации.

Формализм акторов пытается обеспечить следующие возможности:

1) Процедуральное вложение информации, т. е. средства, которыми информация может быть вложена в процедуры.

2) Консервативное развитие, т. е. структурирование информации в виде отдельных боксов ориентированных знаний. При этом имеется возможность создания новых боксов знания и связывания их с уже существующими боксами.

3) Модульная связность, т. е. возможность реорганизовать связи между боксами знаний. При этом модульная эквивалентность означает, что один бокс может быть заменен другим, если все связи между этим и остальными боксами знаний сохраняются теми же.

В настоящее время теория процедурального накопления и тем более обобщения знаний сформулирована недостаточно четко.

Можно выделить ряд исследуемых общих подходов в теории процедурального накопления и обобщения знаний. К ним относятся:

1) Процедуральная абстракция, сводящаяся к обобщению частных случаев (в стиле обобщения треугольных таблиц, п. 7.3.2). Грубо говоря, задача заключается в связывании с полученными частными случаями обобщенных образцов в определенных контекстах. Сюда же относится параметризация некоторых часто встречающихся процедур

2) Исключение абстрактно невозможных случаев Эта техника связана с процецуральной абстракцией и заключается в связывании альтернативе частными случаями и выявлением противоречий

3) Протокольная абстракция, заключающаяся в анализе и связывании протоколов выполнения процедур и преобразовании деревьев протоколов в графы путем исключения неразличимых с точки зрения образцов вершин.

8. Языковые формы формирования предписаний (процедур)

8.1. Структура и задачи подсистемы языковых форм формирования предписаний (процедур)

В данном разделе описана структура и общие принципы построения подсистемы языковых форм формирования предписаний (процедур).

Кажется естественным потребовать, чтобы АКС в части языковых форм формирования предписаний (процедур) умела решать по крайней мере следующие задачи:

1. Понимать предложения естественного языка (возможно, ограниченного).
2. Выводить ответ на основании имеющихся у АКС фактов.
3. Выражать ответ в ограниченном естественном языке.
4. Накапливать и корректировать свои знания на основании информации, воспринимаемой в процессе диалога.

На рис. 8.1 приведена структура подсистемы языковых форм формирования предписаний (процедур), удовлетворяющая перечисленным выше требованиям.

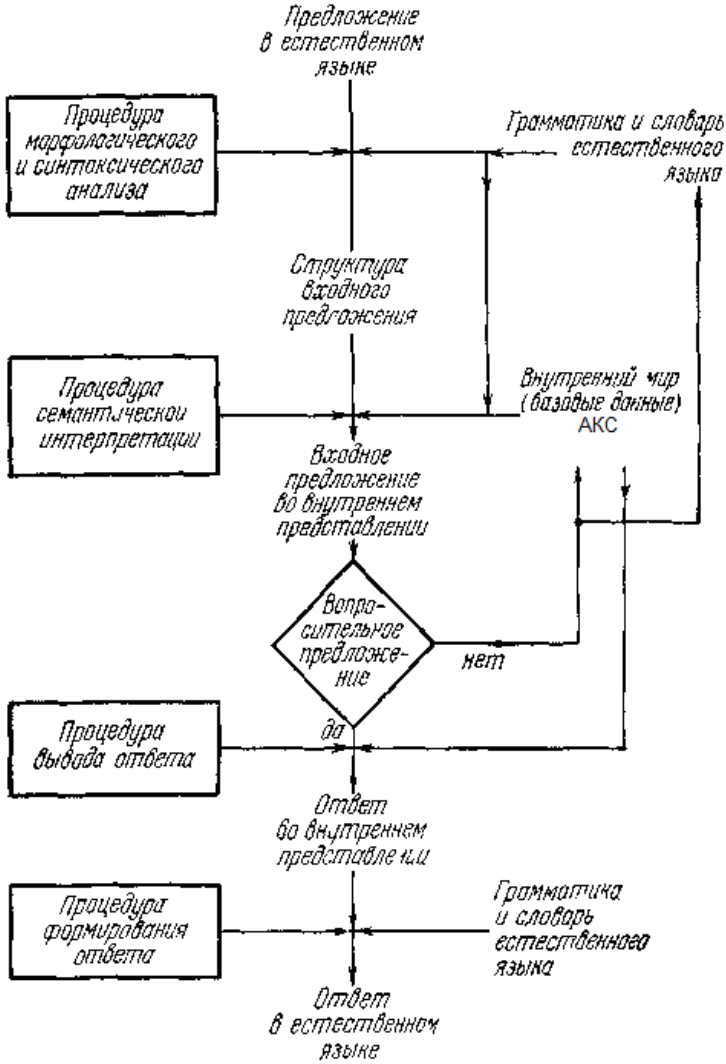


Рис. 8.1. Структура подсистемы языковых форм формирования предписаний (процедур).

Подсистема, приведенная на рис. 8.1, совпадает по структуре и решаемым задачам с вопросно-ответными системами (ВОС) общего типа.

В дальнейшем в основном будет использоваться термин ВОС вместо термина «подсистема языковых форм формирования предписаний (процедур)» Этим мы хотим подчеркнуть, что изложенный в разделе подход может быть использован не только при построении подсистем АКС, но и при разработке изолированных информационно-поисковых или вопросно-ответных систем, воспринимающих ограниченный естественный язык.

Рассмотрим структуру подсистемы языковых форм формирования предписаний (процедур) (рис. 8.1), подразделяемую в соответствии с традициями ВОС на следующие основные этапы.

1. Синтаксический анализ. Синтаксис понимается в широком смысле с включением морфологии. Задача этапа — определить структуру входного предложения в соответствии с грамматикой языка и идентифицировать слова предложения со словарем системы.

2. Семантическая интерпретация. Задача этапа— понять входное предложение, т. е. однозначно выразить его в понятия внутреннего пространства АКС. Если входное предложение является *директивой,* то требуется объединить его с базовыми данными (внутренним пространством). Если входное предложение является *вопросом,* то требуется выделить из базовых данных информацию, необходимую для ответа на поставленный вопрос.

3. Вывод ответа. Задача этапа — вывести из выделенного подмножества базовых данных ответ на заданный вопрос. Ответ выражается в понятиях внутреннего представления.

4. Формирование ответа. Задача этапа — перевести ответ из внутреннего представления в естественный язык.

В данном разделе нашей основной задачей является выделение семантической информации входного предложения и установление взаимосвязи ее с информацией АКС. Решение указанной задачи в значительной степени не зависит от морфологии и частично от синтаксиса языка. Поэтому мы не будем рассматривать вопросы морфологии и будем касаться синтаксиса только в его связи с семантикой.

Системы, воспринимающие естественный язык, основываются на том положении, что для понимания языка необходимо не последовательное применение этапов обработки входного предложения (**синтаксис, семантика, вывод**), а интегральное использование всех аспектов языка. Так, например, для понимания синтаксиса может потребоваться обращение к семантике, к дедукции или к общим знаниям пользователя. Тем не менее мы, в целях методичности изложения, будем описывать этапы работы ВОС последовательно.

На наш взгляд, еще не существует общей теории вопросно-ответных систем, воспринимающих естественный язык. Выполнение ряда этапов ВОС (семантическая интерпретация, дедуктивный вывод) в значительной степени зависит от многих факторов и особенно от представления базовых данных в виде исчисления предикатов (системы с общим выводом, см. п. 8.3.4) или в виде семантических сетей и процедурального представления (системы с ограниченной логикой, см. п. 8.3.3). Указанное обстоятельство нашло отражение и в изложении материала в данноі разделе. Мы будем описывать этапы семантической интерпретации и дедуктивного вывода отдельно для систем с **общим выводом (исчисление предикатов) и систем с ограниченной логикой (семантические сети).**

Прежде чем перейти к описанию вопросно-ответных систем, отметим, что обработка естественного языка является основной задачей не только ВОС, но и области научных исследований, называемой «машинным переводом» (МП). Несмотря на общность ряда этапов ВОС и МП, мы не будем проводить параллели между используемыми в них принципами. По нашему мнению, различие целей ВОС и МП вызывает (по крайней мере в настоящее время) различие в используемых методах.

Перечислим основные различия ВОС и МП.

1. Основной задачей ВОС по обработке языка является преобразование текста в его смысл (при обработке входного предложения) и преобразо вание смысла в текст (при формировании ответа

При МП полного проникновения в смысл фразы может не требоваться. Действительно, если осуществляется перевод с языка Я1 на язык Я2 (рис. 8.2), то априори не видно причин, почему необходимо полное понимание смысла текста (путь Я1→С→Я2 на рис. 8.2), а не частичное (путь Я1 →С1→С2→Я2).

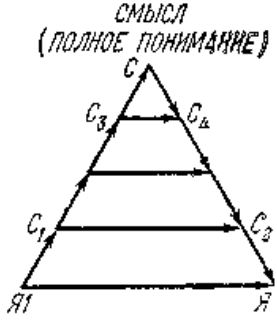


Рис. 8.2. Интерпретация задач машинного перевода и вопросно-ответных систем.

Использование подхода СМЫСЛА $\overrightarrow{\leftarrow}$ ТЕКСТ наметилось и в МП.

2. При создании ВОС мы можем ограничить естественный язык как по тематике, так и по грамматике, и при этом не потеряем практической ценности работы. При МП требуется иметь дело с естественным языком. Попытки ограничиться определенной тематикой не делают задачу настолько простой, чтобы удалось получить практически значимые результаты. Как следствие указанного фактора, в ВОС могут быть использованы более формализованные лингвистические модели, существенно упрощающие естественный язык и позволяющие получить эффективные алгоритмы обработки текста.

Методы, используемые при разработке большинства этапов ВОС, в существенной степени зависят от лингвистической модели, принятой для описания языка. Поэтому в очередном параграфе приведены сведения из лингвистики, необходимые для понимания последующего материала.

8.2. Формальные грамматики

8.2.1. Основные определения

Будем называть *языком* конечное или бесконечное множество предложений, каждое из которых имеет конечную длину и построено с помощью операции соединения из конечного множества элементов. Это определение включает как естественные, так и искусственные языки логики и программирования.

Чтобы точно определить язык, необходимо установить общие принципы, которые отделяют последовательности атомарных элементов, являющихся предложениями, от последовательностей, таковыми не являющихся. Это различие нельзя выразить просто списком, так как для всех систем, представляющих интерес, не накладывается ограничений на количество предложений и их длину. Традиционные грамматики естественных языков не содержат всей информации, необходимой для выполнения указанной задачи. Стремление к созданию более строгой лингвистической теории привело к разработке формальных грамматик, основоположником которых считается Н. Хомский. Он указал на возможность использовать для описания естественных языков некоторые исчисления, рассматривавшиеся ранее в теории алгоритмов, придав им удобную для этого форму (за ними закрепилось данное Н. Хомским название «порождающих грамматик»).

Следуя Хомскому, предъявим к лингвистической теории два требования:
1) порождать все те и только те предложения, которые доставляют описываемый данной грамматикой язык;
2) определять метод, с помощью которого можно было бы дать единственное и единообразное структурное описание каждого предложения, порожденного грамматикой. Было бы удобно, чтобы структурное описание получалось автоматически при порождении предложения.

8.2.2. Формальные грамматики

Введем понятия цепочек и языков, а затем определим понятие порождающей грамматики.
Пусть *V*— непустое конечное множество, которое мы будем называть *словарем (алфавитом),* а его элементы — *символами (буквами)*. Произвольную конечную последовательность элементов ω будем называть *цепочкой* в словаре *V*. Пустую цепочку будем обозначать символом Λ. Число символов в цепочке будем называть *длиной цепочки* и обозначать |ω|. Над цепочками определяется *операция конкатенации.* Конкатенацией непустых цепочек $b_1 \ldots b_n$ и $c_1 \ldots c_p$ называется цепочка $d_1 \ldots d_{n+p}$, где $d_1 = b_1, \ldots, d_n = b_n, d_{n+1} = c_1, \ldots, d_{n+p} = c_p$. Конкатенацию цепочек ω и φ мы будем обозначать ωφ. Кроме того, конкатенация цепочки ω и Λ, равно как Λ и ω, считается по определению равной ω.
Если для каких-либо цепочек ω, φ, η_1, η_2 в словаре *V* имеет место равенство $\omega = \eta_1\varphi\eta_2$, то будем называть цепочку $\eta_1*\varphi*\eta_2$, где *с*имвол * не принадлежит *V, вхождением цепочки* φ в ω. Вхождение символов в цепочку будем называть ее *точками*. Если $\alpha = \eta_1*b*\eta_2$ и $\beta = \xi_1*c*\xi_2$— точки одной и той же цепочки $\omega = \eta_1*b*\eta_2 = \xi_1*c*\xi_2$ и если при этом $|\eta_1| < |\xi_1|$, то мы будем писать $\alpha < \beta$ или $\beta > \alpha$ и говорить, что α расположена левее β, а β — правее α. Для любых двух точек α и β цепочки ω таких, что $\alpha \le \beta$, мы будем называть множество точек ξ, удовлетворяющих неравенствам $\alpha \le \xi \le \beta$, *отрезком цепочки* ω.
Произвольное множество цепочек в словаре *V* мы будем называть *языком* в этом словаре. Естественно, что задание языков практической сложности перечислением всех цепочек, составляющих язык, нецелесообразно. Языки задаются с помощью формальных грамматик, порождающих все цепочки данного языка и только их.
Порождающая грамматика — это упорядоченная четверка

$\Gamma = (V_т, V_н, S, P)$, где $V_т$ и V_H — непересекающиеся непустые конечные множества; S — некоторый элемент из V_H;

P —конечное множество правил вида $\varphi \rightarrow \psi$, где φ и ψ — произвольные цепочки словаря $V_T \cup V_н$ и символ $\rightarrow$ не входит в $V_T \cup V_H$.

Множества V_T и V_H называются соответственно *терминальным (основным)* и *нетерминальным (вспомогательным) словарями,* а их элементы соответственно терминальными и нетерминальными символами грамматики Г. Объединение $V_T \cup V_H$ будем называть *полным словарем* грамматики Г.

S называется *начальным символом* Г. Это выделенный нетерминальный символ, обозначающий класс всех тех языковых объектов, для описания которых предназначена данная грамматика. Иногда символ S называют *целью грамматики.*

Отметим, что при изучении естественного языка в аспекте теории формальных грамматик терминальные символы интерпретируются как *словоформы* (словоформами называют единицы языка, рассматриваемые одновременно в плане выражения (последовательность букв от пробела до пробела) и в плане содержания (совокупность значений).) или *морфемы* (морфемами называют наименьшие осмысленные единицы языка (корни, суффиксы и т. п.)), нетерминальные символы — как названия *классов слов* и *словосочетаний,* а начальный символ— как *предложение.*

P называют *схемой грамматики* Г, а цепочки вида $\varphi \rightarrow \psi$ называют *правилами* Г.

Пусть $r = \varphi \rightarrow \psi$ — некоторое правило грамматики Г и $\xi_1 * \varphi * \xi_2$ — вхождение φ в цепочку $\omega = \xi_1 \varphi \xi_2$ в словаре $V_T \cup V_H$. Говорят, что цепочка $\eta = \xi_1 \varphi \xi_2$ получается из ω применением правила r к вхождению φ в цепочку ω. Если цепочка η получается из цепочки ω применением какого-либо правила Г, будем говорить, что η *непосредственно выводима* из ω в Г и будем писать $\omega \models \eta$.

Последовательность цепочек $D = (\omega_o, \omega_1, \ldots, \omega_n)$, $n \geq 1$, называется *выводом* ω_n *из* ω_o *в грамматике* Г, если для каждого i, $1 \leq i \leq n$, имеет место $\omega_{i-1} \models \omega_i$. Число n называется длиной вывода D. Если существует вывод цепочки η из ω в Г, то будем говорить, что η *выводима из* ω в Г, и писать $\omega \vdash \eta$.

Множество цепочек в основном словаре грамматики Г, выводимых из ее начального символа, называется *языком, порождаемым грамматикой* Г, и обозначается $L(T)$.

Из приведенных выше определений видно, что **грамматика — это исчисление, т. е. разрешение производить некоторые операции —**

в данном случае подстановки, а не алгоритмы, т. е. **предписание производить операции (директивы).**

Грамматики, на правила которых не наложены никакие ограничения, способны порождать любые множества цепочек, порождаемые каким-либо автоматом. В теории алгоритмов такие множества называются *рекурсивно-перечислимыми.* Языки, порождаемые этими грамматиками, образуют слишком широкий класс и не представляют интерес для лингвистики.

Естественно полагать, что для множества цепочек, составляющих некоторый естественный язык, у носителей языка есть распознающий алгоритм, т. е. **способ узнавания принадлежности каждой предъявленной цепочки к цепочкам данного языка**. Более того, этот алгоритм должен давать ответ довольно быстро. **Множества, для которых существуют распознающие алгоритмы, называются *рекурсивными.***

Мы будем рассматривать грамматики, порождающие рекурсивные множества.

При изучении формальных грамматик выделяют три типа грамматик, представляющих наибольший интерес как в теоретическом, так и в практическом смыслах. Эти грамматики задаются путем наложения последовательно усиливающихся ограничений на правила P.

Грамматика Г называется *грамматикой составляющих* или *непосредственно составляющих (НС-грамматикой),* если каждое ее правило имеет вид $\xi_1 A \xi_2 \rightarrow \xi_1 \psi \xi_2$, где ξ_1 и ξ_2— произвольные цепочки в словаре $V_T \cup V_H$, $A \in V_H$ и ψ — произвольная непустая цепочка в $V_T \cup V_H$. При применении НС-правила одно вхождение символа A заменяется на ψ в зависимости от наличия нужного контекста. Поэтому данные грамматики называют также *контекстными,* или *контекстно-чувствительными. Грамматика* Г называется *бесконтекстной* или *контекстно-свободной* (КС), если все ее правила имеют вид $A \rightarrow \psi$, где A — нетерминальный символ, а ψ — произвольная непустая цепочка в $V_T \cup V_H$.

Грамматика Г называется *автоматной грамматикой* или A-грамматикой с конечным числом состояний, если каждое ее правило имеет вид $A \rightarrow aB$ или $A \rightarrow a$, где $a \in V_T$, A и $B \in V_H$.

В указанной последовательности классов порождающих грамматик, каждый последующий класс (в смысле общности правил грамматики) является частью предыдущего.

Языки, порождаемые грамматиками перечисленных классов, называются соответственно *языками непосредственно составляющих,*

или *контекстно-чувствительными, бесконтекстными и автоматными*.

Как было указано ранее, задача грамматики состоит не только в том, чтобы породить предложения языка, но и приписать каждому из них **структурное описание**. В современной лингвистике наиболее употребительны два способа описания синтаксической структуры предложения:

1) описание с помощью систем составляющих;
2) описание с помощью деревьев синтаксического подчинения.

Пусть x — произвольная непустая цепочка в словаре V. Множество С отрезков цепочки x называется *системой составляющих* этой цепочки, если оно удовлетворяет двум условиям:

1) множество C содержит отрезок, состоящий из всех точек цепочки x, и все одноточечные отрезки x;
2) любые два отрезка из С либо не пересекаются, либо один из них содержится в другом.

Элементы C называют *составляющими*. Одноточечные отрезки называют *точечными составляющими;* отрезок, состоящий из всех точек цепочки,— *полной составляющей;* полную и точечные составляющие называют *тривиальными*.

Для наглядности изображения системы составляющих иногда заключают в скобки каждую нетривиальную составляющую. Скобки можно не нумеровать, так как в силу пункта 2 определения системы составляющих каждой левой скобке можно однозначно указать соответствующую ей правую.

Если цепочка x интерпретируется как предложение естественного языка, то система составляющих может быть использована в качестве способа выражения информации о его синтаксической структуре. Такая информация может представлять собой список словосочетаний, то есть тех «кусков» предложения, которые в каком-то интуитивном смысле являются «синтаксически связанными». Эмпирические соображения позволяют сделать допущения, что словосочетания, во-первых, образуют отрезки и, во-вторых, не имеют «частичных пересечений», т. е. удовлетворяют п. 2 определения системы составляющих.

Таким образом, нетривиальными составляющими являются словосочетания (при подходящем выборе системы составляющих). Тривиальные составляющие добавлены для придания системе формальной законченности.

Среди многочисленных систем составляющих, которые имеет предложение естественного языка, лишь весьма немногие «правильны», то есть адекватно отражают синтаксическое строение

предложения. При этом понятие «правильной» системы составляющих не абсолютно, оно зависит от соглашений лингвистического характера, отражающих определенные содержательные представления о синтаксической структуре предложения данного языка. Нетрудно видеть, что системе составляющих можно поставить в соответствие дерево корнем которого служит полная составляющая, а висячими узлами — точечные составляющие. Это дерево называется деревом составляющих. Приведем пример (рис. 8.3) дерева составляющих для фразы

(Онегин, (добрый (мой приятель))),
(родился (на (брегах Невы))).

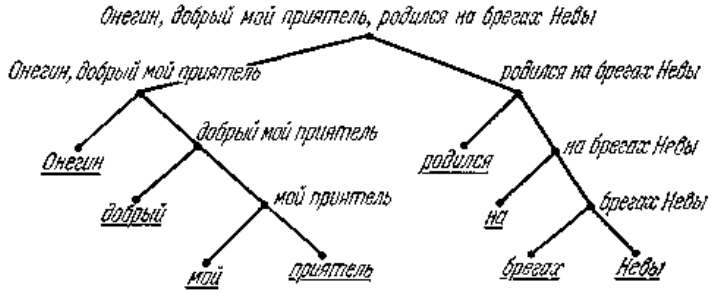


Рис. 8.3. Пример дерева составляющих.

Построенная система составляющих указывает в предложении словосочетания разных «уровней» (составляющие разной высоты), но не вводит при этом никакой иерархии среди словосочетаний одного «уровня». Между тем, в предложениях естественного языка часто интуитивно ощущается «главенствование» некоторого словосочетания над другими, в нем не содержащимися.

Чтобы в некоторой степени отразить этот факт, вводят иерархизованную систему составляющих следующим образом. Пусть C — система составляющих цепочки *x*. Для каждой неточечной составляющей $A \in C$ выделим во множестве всех составляющих, непосредственно вложенных в *A,* какую-либо одну составляющую *A',* которую будем называть *главной*. Множество всех главных составляющих обозначим C' и назовем *иерархизацией системы C*. Упорядоченную пару *(C, C')* назовем *иерархизованной системой составляющих*.

При описании предложений естественного языка с помощью системы составляющих *C* часто используют способ «ведения в эту систему дополнительной информации путем отображения *C* во множество всех

подмножеств некоторого конечного множества, элементы которого называются *метками* и содержательно интерпретируются как символы синтаксических классов слов и словосочетаний. Упорядоченную тройку (C, *W*, φ), где *C*—система составляющих, *W*— множество меток и φ — отображение *C* в 2^W, называют *размеченной системой составляющих*. Пусть множество меток *W*, состоит из элементов, указанных в таблице 8.1.

Таблица 8.1

Члены множества *W*	Содержательная интерпретация слов и словосочетаний
S	Предложение
VP_{xyvw}	Группа глагола } в роде *x*, числе *y*, времени
V_{xyvw}	Глагол } *v* и лице *w*.
NP_{xyz}	Группа существительного } рода *x*, в числе *y*
N_{xyz}	Существительное } и падеже *z*
A_{xyz}	Прилагательное в роде *x*, числе *y* и падеже *z*
PP	Предложная группа
ПР	Предлог «на»

В ней одновременно с перечислением меток приведена их содержательная интерпретация.

Тогда для приведенного ранее примера получим следующую «естественную» разметку (рис. 8.4).

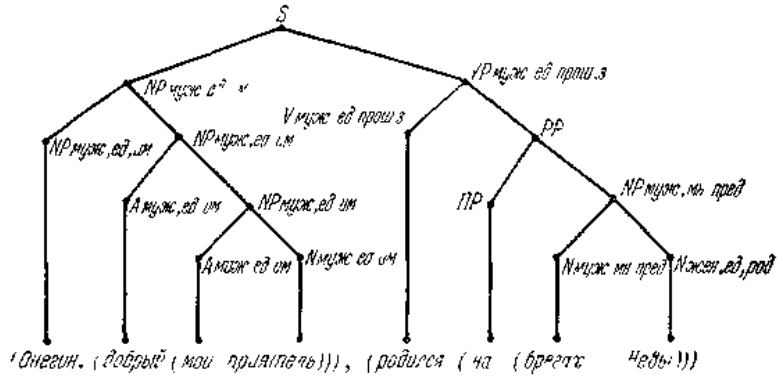


Рис. 8 4. Пример размеченного дерева составляющих.

Заметим, что вывод цепочки в НС-грамматике можно также представить в виде дерева. При этом, если каждому нетерминальному символу *A,* заменяемому в правиле грамматики $\xi_1 A \xi_2 \rightarrow \xi_1 \psi \xi_2$ на цепочку ψ, поставить в соответствие вершину, из которой исходят ребра к символам, образующим цепочку ψ, также интерпретируемым как вершины, то выводу цепочки будет соответствовать дерево. Нетрудно видеть, что это дерево является деревом составляющих. В теории порождающих грамматик его часто называют *С-маркером*
Тот факт, что НС-грамматика при порождении терминальных цепочек одновременно дает их дерево составляющих, делает НС-грамматики особенно интересными с лингвистической точки зрения.
Определим теперь дерево подчинения.
Пусть *x* — произвольная непустая цепочка в словаре *V* и X — множество всех точек в *x.* Произвочьное бинарное отношение → на *X,* при котором граф <*X;* →> является деревом, будем называть *отношением синтаксического подчинения* или просто *отношением подчинения* для *x.* Само дерево <*X;* →> будем называть *деревом (синтаксического) подчинения* для *x.*
При графическом изображении дерева подчинения точки цепочки *x* помещают на горизонтальной прямой (рис. 8. 5), и для всякой пары точек α, β, для которой α→β будем проводить из α в β стрелку, таким образом, чтобы все стрелки были по одну сторону от прямой.
На рис. 8.5 изображено несколько различных деревьев подчинения для одной и той же цепочки *abcdefg.*

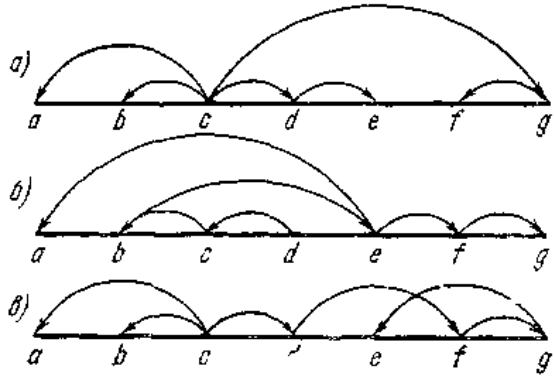
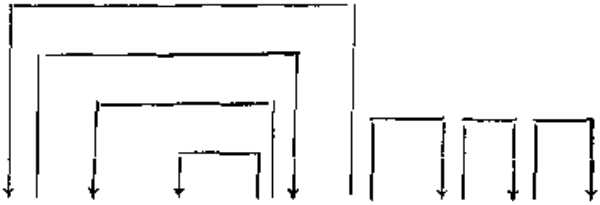


Рис. 8.5. Возможные деревья подчинения для цепочки *abcdefg.*

Деревья подчинения могут быть использованы как один из способов изображения синтаксической структуры предложения. Именно информация о синтаксическом строении предложения может

представлять собой набор сведений о «главенствовании» одних слов (точнее, вхождений слов) в предложении над другими; задать такой набор — значит задать некоторый граф на множестве точек цепочки. Из интуитивных соображений вытекает, что этот граф можно считать деревом.

Приведем пример дерева подчинения для предложения из рис. 8.4, соответствующего «естественным» представлениям о главенствовании одних слов над другими.



Понятие «правильного» дерева подчинения, так же как и системы составляющих, для естественного языка зависит, вообще говоря, от некоторых лингвистических соглашений.

Для анализа предложений естественного языка часто используются размеченные деревья подчинения, в которых, кроме подчинения, указывается его вид.

В классе всевозможных деревьев подчинения выделяют подкласс, который содержит подавляющее большинство «естественных» деревьев для предложений реальных языков. Это класс так называемых *проективных деревьев.*

Дерево подчинения $<X; \rightarrow>$ для цепочки x, а также соответствующее отношение подчинения $\rightarrow$, называется *проективным,* если для любых трех точек α, р, y цепочки x из того, что $\alpha \rightarrow \beta$ и y лежит между аир, следует, что 7 зависит от α (т. е. в дереве существует путь изаву).

Имеется еще один важный класс деревьев подчинения, являющийся расширением предыдущего: класс *слабо проективных деревьев.* Дерево подчинения $<X; \rightarrow>$ для цепочки x (и отношение $\rightarrow$) называется *слабо проективным,* если для любых четырех точек α, β, γ, δ цепочки x из $\alpha \rightarrow \beta$ и $\gamma \rightarrow \delta$ следует, что пары α, β и γ, δ не разделяют друг друга. *(Пары точек α,* β и γ, δ *разделяют друг друга,* если одна из точек γ, δ лежит, а другая не лежит между α и β.)

При графическом способе изображения слабая проективность равносильна возможности провести все стрелки так, чтобы никакие две из них не пересекались, а проективность, кроме того, обозначает, что корень дерева не лежит ни под какой стрелкой.

На рис. 8.5 дерево *а*) проективно, деревья *а), б)* слабо проективны, дерево *в*) непроективно.

Содержательный смысл условий проективности и слабой проективности обозначает, что слова, близкие синтаксически, близки и по положению в тексте.

В так называемой научной и деловой прозе, по крайней мере русской, естественные деревья подчинения подавляющего большинства предложений слабо проективны и даже проективны.

В рассматриваемом нами применении (отправители, получатели)) указанное ограничение не является существенным, но упрощает представление синтаксической структуры предложения.

Можно показать, что существуют правила перехода от систем составляющих к деревьям подчинения и обратно.

Рассмотрим вопрос о возможности использования введенных формальных грамматик для описания естественного языка. В литературе подробно освещен вопрос о недостаточности А-грамматик и КС-грамматик для описания естественных языков в полном объеме. Заметим, однако, что это не исключает их использования на определенных уровнях описания грамматики.

Что касается НС-грамматик и равных им по порождающей силе *неукорачивающих грамматик,* то есть грамматик с правилами $\varphi \rightarrow \psi$, удовлетворяющими тому требованию, что длина цепочки ψ не короче цепочки φ, то они достаточны (хотя и не обязательно удобны) для описания любых естественных языков в полном объеме. Это утверждение вытекает из следующих допущений:

1) любой естественный язык, точнее, множество его правильных фраз, есть легко распознаваемое множество, т. е. существует достаточно простой алгоритм распознавания правильности фраз. Данное допущение является практически очевидным, так как люди обладают таким алгоритмом;

2) алгоритм распознавания правильности фраз естественного языка должен обеспечивать такой процесс распознавания, при котором требуемый объем «оперативной памяти» сопоставим с длиной фразы, например, не превышает числа Mn, где n— длина фразы, а M — достаточно большая константа.

Последнее допущение подтверждается психологическими экспериментами. Автоматы, на работу которых накладывается такое ограничение, называются *линейноограниченными*. Как известно, *множество цепочек, представимых линейноограниченным автоматом, есть контекстно-чувствительный язык*. Таким образом, из наших допущений следует, что НС-грамматики в принципе способны описывать множество правильных фраз любого естественного языка.

Однако, приведенные выше рассуждения не гарантируют удобства описания любого естественного языка.

Отметим основные недостатки НС-грамматик:

1) с помощью НС-грамматик не удается естественно описывать фразы, содержащие непроективные конструкции;

2) НС-грамматика, как и любая порождающая грамматика в смысле данного нами определения, содержит только правила образования языковых выражений, но не содержит правил преобразования правильно построенных выражений.

В литературе по математической лингвистике кроме термина «порождающие грамматики» используется и термин *«распознающие грамматики»*. Деление грамматик на порождающие и распознающие имеет историческое объяснение, хотя по существу порождающие грамматики наиболее интересных типов, в частности, всех типов рассмотренных нами, могут быть использованы также и для распознавания. При этом, если вместо распознавания говорить о «допускании», то все порождающие грамматики могут трактоваться как *допускающие. Грамматика* Г *допускает язык L,* если для Г известна процедура, определяющая для любой цепочки x $(x \in L)$ ее принадлежность к языку L, если же x не принадлежит L, то от процедуры не требуется никакого ответа. При этом допускающая процедура для любой порождающей грамматики состоит просто в применении правил к данной цепочке справа налево, а не как обычно слева направо, до тех пор пока это возможно. Верно и обратное, что распознающие грамматики, например, категориальные грамматики, могут использоваться для порождения фактически опять же применением правила в обратном порядке.

Таким образом, формальные грамматики по существу нейтральны по отношению к порождению и допусканию, а также распознаванию для рассмотренных нами грамматик. Можно говорить просто о формальных грамматиках, рассматривая отдельно от самих грамматик аспект направления применения правил.

8.2.3. Трансформационные порождающие грамматики (ТПГ)

Владение языком предполагает умение не только построить правильную фразу, но и перейти от одной фразы к другим, либо полностью синонимичным ей, либо отличающимся от нее по смыслу на определенную «величину». Примером таких переходов является переход от утвердительного предложения к вопросительному или

отрицательному, переход от активной формы к пассивной, выражение одной и той же мысли разными способами и т. д.

В НС-грамматике все такие предложения будут порождаться более или менее независимо и, следовательно, не будут находиться в каких-либо явных отношениях друг к другу.

Соответствующая задача была впервые четко сформулирована Н. Хомским. Выдвинутая им концепция приобрела широкую популярность под названием «трансформационной грамматики». Фактически, дело заключается во введении еще одного семантического уровня описания языка. В самом деле, инвариантом всех трансформаций (преобразований) является смысл. Таким образом, теория трансформаций оказывается по существу теорией синонимии в языке.

Отметим, что НС-грамматики описывают синтаксический уровень в широком смысле, т. е. с включением морфологии, а трансформационные грамматики включают и семантические преобразования. Поэтому, когда говорят о недостаточности НС-грамматик для описания языка, следует понимать, что это верно только в смысле неохвата НС-грамматиками семантического уровня. На своем, чисто синтаксическом уровне, НС-грамматики оказываются принципиально вполне достаточными.

Основные положения ТПГ изложены в ряде работ Хомского. Согласно этим представлениям, ТПГ состоит из трех основных компонентов: с*интаксического, фонологического, семантического*. По Хомскому, собственно порождающей частью теории является только синтаксическая компонента, где по правилам грамматики происходит порождение *глубинной и поверхностной структур*. Семантическая и фонологическая компоненты интерпретируют соответственно глубинные структуры и поверхностные. Таким образом, центральная роль отводится синтаксической компоненте, состоящей из двух субкомпонент: *базовой* (или просто *базы)* и *трансформационной*. База синтаксической компоненты служит для порождения *глубинных структур*. В содержательном отношении глубинная структура является представлением логического содержания порождаемого предложения в терминах элементарных суждений. Задача трансформационной субкомпоненты состоит в том, чтобы перейти от глубинной структуры к предложению в естественном языке *(поверхностной структуре)* либо полностью синонимичному с ней, либо отличающемуся от нее по смыслу на некоторую величину.

Формально глубинные структуры представляют собой деревья *(С-маркеры),* порождаемые *категориальными правилами* (категориальная компонента) и лексиконом. Категориальные правила представляются в

виде правил НС-грамматики, не содержащих терминальные символы, т. е. порождаемые ими цепочки состоят только из категориальных символов (*N*— существительное, *V*— глагол, *D* — детерминатив и т. п.) Бесконечная порождающая способность грамматики категориальной компоненты вытекает из того факта, что допускается введение начального символа *S* (предложения) в строки вывода. Таким образом, правила подстановки могут, по существу, включать одни базовые С-маркеры в другие.

Подстановка терминальных символов в порождаемые предложения осуществляется с помощью трех видов правил: правила *субкатегоризации,* правил *селекции* и правил *лексического включения.* Субкатегориальные правила и правила селекции комбинируют категориальные символы вместе с субкатегориальными в *комплексные символы,* осуществляя замены типа *N*→[+*N,* + Нарицательность, + Исчисляемость, + Одушевляемость, — Абстрактность]. Правила лексического включения подставляют в порождаемое дерево вместо правой часги приведенного выше правила лексическую единицу с подходящими набором признаков, например «мальчик» (см. табл. 7.2)

Таблица 8.2

Фрагмент лексикона трансформационной грамматики

C	*D*
[+ *N*, + Нарицательность, + Исчисляемость, + Одушевленность, + Человечность]	мальчик
[*V* +, + Транзитивность, [+ Абстрактность] *AUX_DET* [+ Одушевленность]]	испугать
[+ *M*,]	мочь
[+ *N*, + Нарицательность, — Исчисляемость, + Абстрактность]	искренность

Итак, лексикон состоит из записей, связанных с трансформациями подстановки, которые вводят лексические единицы (лексемы) в цепочки, порожденные категориальным компонентом, образуя так называемый *обобщенный С-маркер.* Все контекстуальные ограничения в базе обеспечиваются этими трансформационными правилами лексикона *(лексическими трансформациями).*

Отметим, что даже простому предложению в глубинной структуре может соответствовать система из нескольких элементарных

суждений. На рис. 8.6 приведена глубинная структура простого предложения «Невидимый бог создал видимый мир».

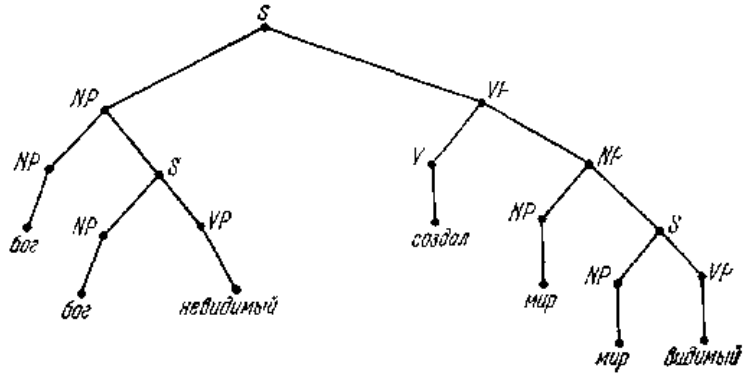


Рис. 8.6. Глубинная структура предложения «Невидимый бог создал видимый мир».

Как уже было указано ранее, задача трансформационной субкомпоненты состоит в том, чтобы из глубинной структуры, порожденной базой, получить поверхностную структуру, используя *грамматические (нелексические) трансформации.* Каждая трансформация представляется в виде правила, условие применимости которого задается в виде С-маркера.

Имея обобщенный С-маркер, мы строим трансформационный вывод, применяя нелексические трансформационные правила последовательно «снизу вверх». Другими словами, мы применяем последовательность правил к данной конфигурации только в том случае, если мы уже применили ее ко всем базовым С-маркерам, вставленным в эту конфигурацию.

Итак, мы определили ТПГ, которая включает систему правил структуры составляющих, порождающих деревьями систему трансформационных правил, отображающих деревья в деревья. Такая грамматика задает бесконечный класс конечных последовательностей деревьев *(P_1, . . ., P_n)*. Каждая последовательность, называемая выводом, удовлетворяет следующим условиям:

P_1—дерево, порождаемое правилами категориального компонента;

P_n — дерево, представляющее поверхностную структуру порожденного предложения;

P_i (для любого i, кроме i=l) образовано применением к дереву P_{i-1} одного из трансформационных правил.

Отметим, что с каждой лексической единицей связана определенная лексическая трансформация, включающая ее в дерево.

В соответствии с требованиями теории Хомского подобные лексические трансформации предшествуют нелексическим трансформациям, и границей между двумя видами трансформаций является глубинная структура.

Полученная в результате трансформации поверхностная структура отображается с помощью фонологических правил на фонетические представления. Система таких правил образует *фонологическую компоненту* (рис. 8.7).

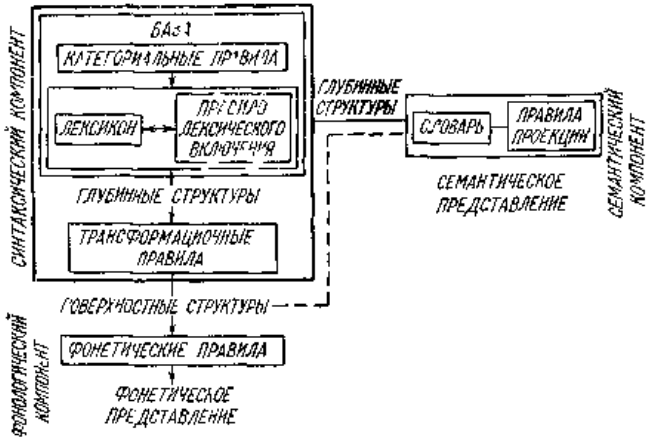


Рис. 8.7. Схема теории трансформационных порождающих грамматик («Стандартная теория»).

С другой стороны, глубинные структуры, после того как в них введены конкретные лексемы, отображаются с помощью семантической компоненты на семантические представления. Семантическая компонента состоит из *системы проекционных правил* и *словаря,* где каждая единица получает словарную статью, состоящую из стольких прочтений (readings), сколько у единицы значений.

Правила проекции идут снизу по дереву и сочетают прочтения единиц, образующих составляющие, учитывая при этом имеющиеся в прочтениях селекционные ограничения, запрещающие те или иные сочетания значений. Получившийся и результате суммарный смысл и представляет собой семантическую интерпретацию, или прочтение данной глубинной структуры

Катц и Постал показали, что возможна такая перестройка теории, при которой трансформации никогда не меняют смысла, и вся смысловая информация предложения содержится уже в глубинной структуре. Поэтому семантической компоненте достаточно ее только интерпретировать.

Изложенная выше теория ТПГ называется «стандартной теорией» Хомского В дальнейшем некоторые новые лингвисчические факты заставили Хомского признать в рамках «пересмотренной стандартной теории» зависимость семантического представления не только от дубинной, но и от поверхностной структуры (см. пунктир на рис. 8.7) Концепцию семантической компоненты, развитую в рамках «пересмотренной стандартной теории», называют *интерпретирующей семантикой* (ИС).

Существенный вклад в разработку теории глубинных структур в рамках интерпретирующей семантики внес Ч. Филмор. В отличие от Хомского Ч. Филмор считает, что основу предложения образует не субъектно-предикатная, а предикатно-аргументная структура. Аргументами такой структуры Филмор считает имена, для которых может быть указан глубинный падеж. При этом глубинные падежи выявляются в результате анализа так называемых скрытых категорий, проявляющихся в характере проведения трансформаций, перифразов и т. д. Падеж при таком понимании рассматривается как универсальное явление, присущее всем языкам независимо от того, имеется ли в них падеж в традиционном смысле или нет. Итак, ***глубинный падеж*** **— это обобщенное отношение между содержанием глагола и содержанием той или иной из его именных групп.** Ч. Филмор предлагает использовать семь глубинных падежей:

— агентивный *(А)* — одушевленный возбудитель лействия *(Джон* открыл дверь);

— инструментальный (*I*) — падеж неодушевленного предмета или силы, составляющих причину глагольного действия (состояния) *(Камень* разбил стекло.);

— дательный (D) — падеж одушевленного существа, затронутого глагольным действием (состоянием) *(Мальчик* получил удар в лицо.);

— фактитивный *(F)* — падеж предмета или существа, возникающего в результате действия (состояния) или входящего как часть в само глагольное действие (Мать варит *картошку.);*

— локативный *(L)* — место или пространственная ориентировка глагольного действия (состояния) (Джон идет *по улице.);*

— бенефактивный *(В)* — падеж пользователя (Мать варит картошку *для Джона.);*

— объективный (*О*) — немаркированный падеж, падеж любой вещи, представленной в виде имени, роль которой по отношению к глагольному действию (состоянию) определяется из семантического толкования самого глагола (Джон открыл *дверь.)*.

Филмор подчеркивает, что состав набора, а также характеристики и названия отдельных падежей не являются окончательными.

Глубинная структура по Ч. Филмору имеет следующий вид:

$S \rightarrow Aux + P$, где S— предложение, *Aux* — модальный показатель (modality), *P*— пропозиция (proposition). Пропозиция может быть развернута в формулу следующего вида:

$$P \rightarrow V + C_1 + \ldots + C_n,$$

где *V*— глагол, $C_1 + \ldots + C_n$ — глубинные падежи.

С помощью подобных правил порождаются *претерминальные цепочки,* графическое представление которых дано на рис. 8.8.

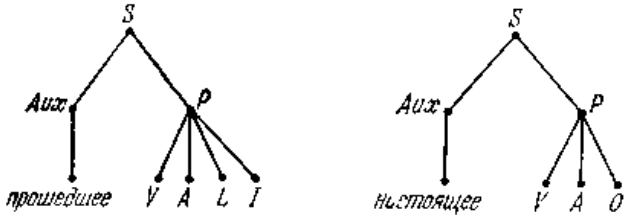


Рис. 8.8. Претерминальные цепочки падежной грамматики.

Отметим, что в данной грамматике, как и у Хомского, предполагается наличие *лексикона, правила лексического включения* и *трансформационных правил.*

Итак, по правилу лексического вывода, замена *комплексного символа С* претерминальной цепочки на некоторую словоформу *D* происходит при отождествлении этого символа с входной информацией *С* этой словоформы в словаре (табл. 8.3).

Таблица 8.3

Фрагмент лексикона в падежной грамматике

C	*D*
[+*A*]	*by NP*
[+*I*]	*with NP* (если во всей фразе уже есть *by*)
+[—*ALI*]	*smear* (*мазать*)
+[—*AOD*]	*show* (*показывать*)

Комплексный символ словоформы может содержать признаки двух родов: внутренние, характеризующие внутреннюю структуру словоформы [+X, +Y], и внешние, указывающие на признаки окружающих слов (с прочерком на месте данного слова) [__X]. В табл. 8.3 приведен фрагмент лексикона в падежной грамматике.
Пользуясь базовым компонентом грамматики, можно вывести лексически заполненные предложения, типа приведенного на рис. 8.9.

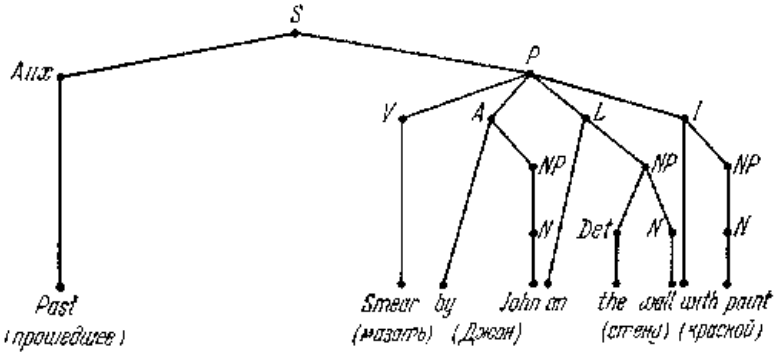


Рис. 8.9. Претерминальная цепочка падежной грамматики с лексическим заполнением.

К окончательному виду подобные предложения приводятся применением *общих и частных трансформационных правил*. Общие правила определяют выбор глубинных падежей на роль поверхностного субъекта и способы субъектного оформления слова. Частные правила определяют особенности состава и поверхностного оформления глубинных падежей у индивидуальных глаголов, если эти особенности не соответствуют типовым словарным предписаниям или общим трансформационным правилам.
Общие основания интерпретирующей семантики (ИС) были подвергнуты критике со стороны представителей *порождающей семантики* (ПС). Следует однако иметь ввиду, что и ИС, и ПС опираются на общую теорию трансформационных порождающих грамматик. Однако в ПС роль активного творческого элемента в процессе порождения высказывания принадлежит не синтаксису, а семантике. Соответственно этому, базовая компонентак в модели порождающей семантики представляет собой правила образования семантических представлений предложений. Затем эти представления преобразуются с помощью трансформационных правил в поверхностные структуры. Модель ПС кроме того содержит также лексикон, заменяющий комбинации элементарных семантических

смыслов лексемами. Схема лингвистической теории ПС представлена на рис. 8.10.

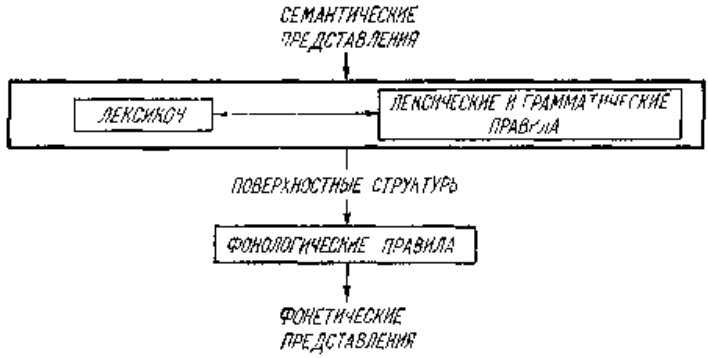


Рис. 8.10. Схема лингвистической теории в порождающей семантике

Следует отметить, что основные расхождения ПС и ИС касаются непризнания ПС уровня глубинных структур.

ПС не представляет собой такой целостной концепции, как ИС, и поэтому, несмотря на свои несомненные преимущества перед ИС, не нашла пока широкого использования в практических разработках.

Следует отметить, что при практической реализации указанные различия между ПС и ИС сглаживаются, и с точки зрения практики нам кажется целесообразно говорить о **двух уровнях языка: уровне глубинных структур в широком смысле и уровне поверхностных структур.** На первом уровне структуры характеризуются чисто семантическими свойствами, а на втором семантика осложнена синтаксическими категориями и преобразованиями. Первый уровень не зависит от конкретного языка, а второй воплощается в конкретном естественном языке. Что же касается необходимости введения промежуточного уровня между глубинным и поверхностным, то он может использоваться просто как удобный технический прием, облегчающий переход от поверхностного уровня к глубинному и обратно.

На этом мы закончим рассмотрение лингвистических аспектов, используемых при построении вопросно-ответных систем (ВОС) и перейдем к самим ВОС.

8.3. Классификация вопросно-ответных систем, понимающих естественный язык

Вопросно-ответные системы, понимающие естественный язык, по способам представления и использования знаний можно условно разбить на четыре типа:

1) системы, использующие форматы частного вида;
2) системы, основанные на запоминании текста;
3) системы с ограниченной логикой;
4) системы с общим выводом.

8.3.1. Системы, использующие форматы частного вида

К этому классу относятся наиболее ранние программы. Они обычно используют два частных формата — один для представления знаний, хранимых в системе, а другой для представления предложений в языке. Такие программы исходят из предположения, что необходимой информацией в предложении является только та, которая соответствует их частным форматам. Хотя программы данного типа могут иметь усложненные механизмы для использования этой информации, они создаются для частных целей и не обладают в управлении информацией гибкостью, которая бы могла позволить использовать программы для других целей.

8.3.2. Системы, основанные на запоминании текста

Стремление преодолеть ограничения рассмотренных выше программ привело к созданию систем, использующих текст в естественном языке во всем его разнообразии и общности как основу для представления знаний в системе. Запомненный текст снабжается различного рода схемами индексирования, предназначенными для упрощения поиска запрашиваемых предложений. Задача системы состоит в выдаче одного или нескольких предложений из знаний системы, имеющих отношение к запросу. Разработаны разнообразные методы для поиска уместных предложений и выбора тех, которые наиболее удовлетворяют запросу.

Данный подход имеет ряд трудно разрешимых проблем. К ним относятся:

1) невозможность получения ответа на любой вопрос, который требует некоторых выводов из более чем одной части запомненной информации;

2) качество ответов в большей степени зависит от формы, в которой текст и вопрос определены в естественном языке, чем от смысла текста и запроса.

8.3.3. Системы с ограниченной логикой

Системы данного вида разрабатывались с целью устранения недостатков систем, основанных на запоминании текста в естественном языке. В первую очередь в этих системах в качестве базовых знаний вместо предложений в естественном языке использовалась более формальная нотация, преследующая цель представить семантические отношения между данными. Коль скоро знания записаны в этом виде, система должна уметь переводить входные предложения в естественном языке в формат внутреннего представления, т. е. выполнять семантическую интерпретацию.

Основным недостатком ранних систем с ограниченной логикой был тот, что сложная информация выражалась в них в форме программ и для введения новых объектов требовалась разработка новых программ и их связь со старыми программами. В свою очередь каждое изменение в программе могло привести (и на практике обычно приводило) к изменению других программ. В результате система росла, теряла стройность, обозримость, и экспериментировать с такой системой становилось практически невозможно. Выход из указанных затруднений был найден за счет разработки новой техники программирования, способной использовать процедуральную информацию, но в то же самое время выражающую эту информацию способом, не зависящим от специфики программы или темы диалога.

Типичным примером систем подобного рода, называемых *системами с процедуральной дедукцией,* является система Винограда. Система выполнена в языках PLANNER, PROGRAMMER (модификация LISP) и LISP. Система отвечает на вопросы, выполняет команды и принимает информацию в процессе ведения диалога на английском языке. Система состоит из разборщика грамматики английского языка, выполненного в PROGRAMMERS, программ семантического анализа, выполненных в LISP, и общей системы принятия решений, выполненной в PLANNER'e. Система включает в себя также в виде теорем, записанных в языке PLANNER, детальную модель простого мира игрушек и упрощенную модель ее собственной разумности. Факты о текущей сцене представлены в виде утверждений PLANNER'a.

Система организует знания как набор программ, называемых «специалистами». Каждая из этих программ несет в себе частные

знания системы о структуре языка и окружающем мире. Так, например, существуют синтаксические специалисты, которые анализируют различные типы фраз. Кроме того, существуют семантические специалисты, выполняющие различные функции. Они выбирают значения индивидуальных слов в зависимости от контекста, определяют, какие комбинации слов имеют смысл, определяют виды ссылок. Семантические специалисты могут вызывать знания о мире, о ситуации, о беседе.

Каждый из специалистов может потенциально использовать любую информацию, собранную о предложении, контексте или ситуации реального мира другими специалистами. Эта гибкость необходима для управления английским языком, но она создает самостоятельную проблему, так как часный специалист интересуется частными аспектами процесса понимания языка и имеет специальную организацию информации. Для решения этой проблемы в системе используются развитые описывающие структуры и явное описание всех свойств текущей структуры.

8.3.4. Системы с общим выводом

Попытки увеличить дедуктивную мощность систем с ограниченным выводом привели к идее **выражать знания в некоторой математической нотации (например, в исчислении предикатов) и затем использовать результаты в области математической логики по формальному доказательству теорем. В таких системах вопрос, заданный в естественном языке, представляется в виде формулы исчисления предикатов и трактуется как теорема, которая должна быть доказана.**

Стимулом к развитию таких систем послужил разработанный Робинсоном принцип резолюции, являющийся полной универсальной процедурой доказательства для исчисления предикатов первого порядка. **На основании работы Робинсона стало возможным создание эффективных автоматических программ, доказывающих теоремы и обладающих двумя важными свойствами. Первое свойство состоит в том, что процедура доказательства в ней универсальна, т. е. не учитывает специфику данной области. Область задается множеством утверждений (аксиом). Второе свойство состоит в том, что если доказательство возможно с использованием правил исчисления предикатов, то процедура гарантирует, что оно будет найдено, хотя, возможно, за очень длительное время.**

Эти свойства удобны, но плата за их использование велика: снижение эффективности системы.

Подход к вопросно-ответным системам с использованием универсальной процедуры доказательств был развит в ряде разработок. В этих системах информация представляется единым образом, не зависящим от особенностей частной программы. Это дает возможность пользователю описывать знания в нейтральном виде, не приспосабливая их к особенностям и частностям вопросно-ответной системы, и гарантирует то, что система будет применима к любой области, представимой в исчислении предикатов.

Использование исчисления предикатов имеет и серьезные недостатки. Представление сложной информации в нейтральной форме игнорирует важный вопрос, как представлять знания. Человек не хранит в голове точно определенное множество логических аксиом, из которых он выводит знания с помощью процедуры доказательства. Скорее у него есть масса эвристик и процедур различной степени общности для решения и представления различных задач. Игнорирование этого факта приводит к неэффективности при представлении в исчислении предикатов объектов реальной сложности. К недостаткам использования исчисления предикатов носится и то, что определенные свойства естественного языка плохо представимы в двузначной логике. Устранить указанные недостатки можно, используя размытые множества и модальные логики. Однако для этих логик в настоящее время не известны эффективные разрешающие процедуры.

8.4. Синтаксический анализ

Задача этапа синтаксического анализа (СА) предложения, записанного в некотором языке, заключается в построении структуры предложения, определяемой грамматикой этого языка. Так, например, при записи предложения в языке, определяемом КС-грамматикой, задача СА заключается в получении дерева составляющих (дерева зависимостей) исходного предложения При обработке языка, описываемого трансформационной грамматикой, задачей СА уже будет не получение поверхностной структуры предложения, а построение его глубинной структуры. В связи с тем, что для удобства и компактности описания естественных языков используются трансформационные грамматики или их модификации, мы будем считать задачей СА получение глубинной структуры предложения, выражаемой, например, и виде дерева составляющих или в виде формул исчисления предикатов. Однако, для введения в историю вопроса и определения используемой в

дальнейшем терминологии, мы приведем краткую характеристику ранних анализаторов.

8.4.1. Синтаксические анализаторы КС-языков

Когда создавались первые *синтаксические анализаторы (разборщики)* предложений естественного языка, не существовало теории синтаксиса, приемлемой для использования на машинах. Анализаторы представляли набор подпрорамм, который постепенно развивался по мере того, как рамматика разширялась и управляла все более сложными предложениями. Анализаторы имели те же недостатки, что и любые программы, конструируемые таким образом. В результате анализаторы становились все сложнее и все труднее становилось понимать их внутренние взаимосвязи. Неудачные попытки в области раннего машинного перевода ясно показали, что задача обработки естественного языка без лучшего понимания основ лингвистики и математических свойств грамматики является преждевременной.

В последующих исследованиях по обработке естественного языка на ЭВМ можно выделить два подхода.

В первом подходе игнорируется синтаксис в его традиционном понимании, и для получения информации о предложении используется процесс сопоставления по образцу. Системы этого типа не предпринимали попытки полного синтаксического анализа входного предложения. Они или ограничивали входной язык небольшим множеством фиксированных форматов, или ограничивали понимание предложений в результате игнорирования синтаксиса.

Во втором подходе берется упрощенное подмножество естественного языка, которое может быть описано в хорошо изученной формальной грамматике, например, такой, как некоторая вариация КС-грамматики.

Не останавливаясь детально на существе ранних алгоритмов разбора, воспринимающих КС-грамматики, отметим, что они разделяются по двум основным параметрам:

1) направлению разбора «сверху-вниз» или «снизу-вверх»;

2) последовательному или параллельному методу генерации деревьев разбора.

Деление алгоритмов по направлению разбора основывается на делении грамматик на порождающие и распознающие. Алгоритмы «сверху-вниз» теоретически основываются на идее использования порождающей грамматики при генерации всех возможных предложений языка, пока не будет порождено предложение, соответствующее входному. Такой подход при прямолинейном

использовании требует слишком много времени, но существуют способы, повышающие эффективность этого метода.

Алгоритмы «снизу-вверх» основываются на распознающих грамматиках. Алгоритмы пытаются объединить различным образом элементы входной строки до тех пор, пока не будет найдено дерево, покрывающее всю входную строку.

Алгоритмы разбора делятся на последовательные или параллельные в зависимости от того, строят они одно дерево разбора (если оно не соответствует входному предложению, то строится новое дерево, до тех пор пока не будет получен правильный разбор) или одновременно все возможные деревья разбора.

Последовательные алгоритмы являются медленными, но требуют мало памяти. К числу наиболее интересных параллельных алгоритмов относится алгоритм Эрли, который рассматривает одновременно (параллельно) все возможные анализы и получает все варианты разбора входной строки за время, пропорциональное кубу длины входной строки. Коэффициент пропорциональности определяется рассматриваемой КС-грамматикой и не зависит от вида строки. Для грамматик специального вида данный алгоритм автоматически достигает времени разбора, пропорционального квадрату длины (или длине) входной строки.

Как было указано ранее, контекстно-свободные грамматики не могут описать естественный язык в полном объеме.

8.4.2. Анализаторы языков, описываемых трансформационными грамматиками

Все более поздние попытки по расширению мощности грамматики, описывающей естественный язык, так или иначе связывались с трансформационными грамматиками. Из определения трансформационной грамматики следует, что она ориентирована на генерацию предложений (синтез), а не на распознавание (анализ). Хотя существует алгоритм, использующий трансформационную грамматику для анализа предложений, он слишком неэффективен, и вопрос о его практическом использовании даже не встает. Алгоритм работает по принципу «анализ через синтез», т. е. генерирует все возможные предложения и ищет среди них анализируемое.

Две попытки разработать более практичные алгоритмы трансформационного распознавания привели к созданию алгоритмов, или требующих больших затрат времени, или теряющих свойство полноты. Оба алгоритма пытались анализировать предложения, применяя трансформации в обратном порядке.

8.4.3. Анализ естественных языков, описываемых расширенными сетями переходов

Далее были разработаны системы, воспринимающие входной язык, близкий к естественному (английскому). Эти системы используют понятие *расширенной сети переходов* (augmented transition network) и основываются на трансформационной грамматике. Их задача выработать глубинную структуру предложения, анализируя его поверхностную структуру.

Введем понятия расширенной и *рекурсивной сети переходов,* являющиеся основными в перечисленных выше анализаторах и используемые ими для задания грамматики.

Рекурсивная сеть переходов (РСП) есть направленный граф с помеченными вершинами (состояниями) и дугами. Выделяется начальное состояние и множество конечных состояний (рис. 8.11).

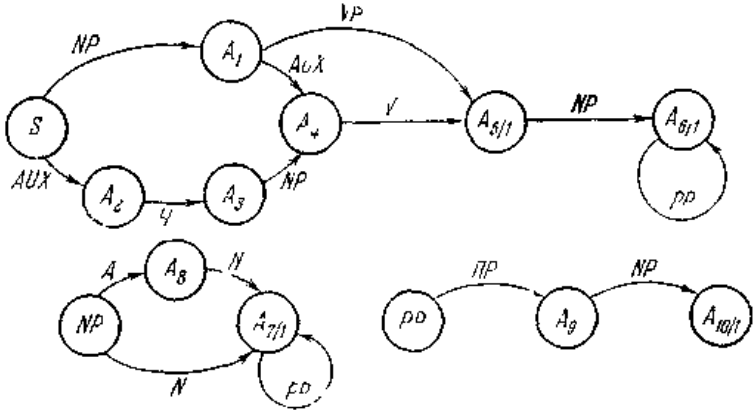


Рис. 8.11. Фрагмент грамматики, представленный в виде рекурсивной сети переходов. S — начальное состояние; A_5, A_6, A_7, A_{10}— конечные состояния; AUX — вспомогательный глагол; Ч — частица; PP — предложная группа существительного; ПР — предлог; AUX, Ч,V, A, N, ПР лексические (терминальные) категории.

Метки на дугах могут быть как терминальными символами, так и наименованиями состояний.

Дуга, помеченная наименованием состояния, интерпретируется следующим образом:

1) наименование состояния на конце дуги запоминается в стеке;

2) управление передается к состоянию, которым помечена дуга (без изменения обрабатываемого символа входной строки).

Будем в дальнейшем называть операции, перечисленные в пунктах 1 и 2, *погружением.*

Если текущее состояние является конечным, то происходит выталкивание последнего запомненного в стеке состояния и передача управления в это состояние. Критерием приемлемости входной строки является попытка вытолкнуть пустой стек, когда последний символ входного предложения обработан. Наименования состояний, появляющихся на дугах в этой модели, соответствуют наименованиям нетерминальных конструкций, которые могут быть обнаружены во входных предложениях.

На рис. 8.11 приведен пример рекурсивной сети переходов для подмножества русского языка. Конечные состояния имеют вид $A_{i/1}$. Приведенная РСП принимает, например, такие предложения, как: «Большой Джон будет строить красивый дом у белой реки», «Будет ли Михаил строить кооператив в Москве?» и т. п.

Нетрудно установить, что РСП эквивалентна недетерминированному автомату с магазинной памятью и имеет мощность контекстно-свободной грамматики. Как было уже указано, КС-грамматика не является адекватным механизмом для описания естественного языка. Увеличим возможности РСП, определив понятие расширенной сети переходов (РАСП), которая способна выполнять трансформационные преобразования, не вводя понятия трансформационной компоненты. Основное свойство, которое трансформационная грамматика добавляет к КС-грамматике — это способность передвигать, копировать и устранять фрагменты синтаксической структуры (так, что их положение в глубинной структуре отличается от положения в поверхностной структуре) и совершать эти действия в зависимости от контекста. Мы можем ввести эквивалентные свойства в РСП, добавив к каждой дуге:

1) произвольные *условия* (переход в состояние, указываемое дугой, разрешается только при выполнении этих условий);

2) произвольные *действия* (действия выполняются, если осуществляется переход к состоянию, на которое указывает дуга).

Будем называть это расширение РСП *расширенной сетью переходов* (РАСП).

РАСП строит частичное структурное описание предложения при переходе из одного состояния сети в другое. Куски этого частичного описания хранятся в регистрах (со стековой структурой), и их содержимое автоматически проталкивается (погружается), когда рекурсивное применение вызывает переход к более нижнему уровню.

Действия, связанные с дугами, изменяют содержимое этих регистров в терминах их предыдущего содержимого, содержимого других регистров, текущих входных символов и (или) результатов нижних уровней вычисления. Кроме того, регистры могут использоваться для хранения указателей (флагов), отражающих особенности прохождения через сеть. Указатели могут опрашиваться *условиями,* связанными с дугами.

Каждое конечное состояние связывается одним или несколькими условиями, при выполнении которых осуществляется переход на предыдущий уровень. Каждому из этих условий ставится в соответствие функция, результат вычисления которой возвращается как итог при переходе на верхний уровень.

Чтобы рассмотрение РАСП было более конкретным, приведем в табл. 8.4 формальное определение языка, в котором РАСП может быть представлена.

Таблица 8.4

Формальное определение языка для описания расширенных сетей переходов

```
<сеть переходов> → (<множество дуг> <множество дуг> *)
<множество дуг> → (<состояние> <дуга> *)
<дуга> → (КАТ <имя категории> <условие> <действие> *<переход>) |
         (ПОГР <состояние> <условие> <действие> * <переход>) |
         (УСЛ<произвольн.метка><условие><действие>*<переход>) |
         (ВЫТ <форма> <условие>)
<действие> → (ПРТЕК <регистр> <форма>) |
             (ПРПОГР <регистр> <форма>) |
             (ПРВЫТ <регистр> <форма>)
<переход> → (ПУ <состояние>) |
            (ПП <состояние>)
<форма> → (ЗНАЧ <регистр>) |
          Δ |
          (СВОЙСТВО <свойство>) |
          (ДЕР <фрагмент> <регистр> *) |
          (СПИСОК <форма> *) |
          (ОБЪЕДИН <форма> <форма>) |
          (ИМЯ <произвольная структура>)
```

Определение дадим в форме расширенной бэкусовской нотации. Будем обозначать вертикальной чертой | альтернативные варианты, операцию * будем использовать как индекс, присутствие которого указывает на возможность вхождения конституента произвольное

число раз. Нетерминальные символы грамматики будем представлять в виде текста из строчных букв, заключенного в угловые скобки, а терминальные (исключая *, Δ, |) заглавными буквами, не заключенными в скобки. Символом Δ будем обозначать наименование регистра, содержащего текущее слово входного предложения.

Дадим пояснения к определениям. Первая строка в таблице указывает, что сеть переходов есть множество дуг (или несколько таких множеств), заключенное в круглые скобки. Множество дуг состоит из наименования состояния, за которым следует любое число дуг. Мы будем предполагать, что РАСП представляется в виде списочной структуры. Тогда сеть переходов есть список, элементы которого — множество дуг. Множество дуг в свою очередь есть список, элементами которого являются наименование состояния и дуги.

Дуги представляются в виде списков четырех разновидностей. Условия и действия, связанные с конечными состояниями, представляются как псевдодуги. Первый элемент списка (дуга) указывает тип дуги (КАТ — задает категорию терминального наименования, которым помечена дуга, ПОГР — указывает, что данный тип дуги вызывает операцию погружения, УСЛ — задает условие, ВЫТ — указывает на необходимость выполнения операции выталкивания стека). Третий элемент определяет произвольное условие, при выполнении которого осуществляется переход к состоянию, указанному в пятом элементе. Четвертый элемент определяет действие, осуществляемое при переходе к состоянию, на которое указывает дуга.

Опишем более подробно типы дуг. Дуга типа КАТ — дуга, которой следуют, если текущее входное слово является лексической (терминальной) категорией, указанной во втором элементе дуги.

Дуга типа ПОГР используется для обработки дуг, помеченных наименованием состояния (что вызывает операцию погружения)

Дуга типа УСЛ (условие) используется для задания произвольного условия, определяющего возможность следования к состоянию, указываемому дугой.

Дуга типа ВЫТ (выталкивание) является пустой дугой и предназначена для того, чтобы иметь возможность связать с конечными состояниями определенные условия (выполнение которых необходимо для подъема на более верхний уровень) и действия (возвращаемые на верхний уровень). Представления условий и действий в виде информации, связанной с пустой дугой, дает возможность упорядочить выбор ВЫТ по отношению к другим дугам данного состояния.

Состояние, к которому осуществляется переход, указывается в пятом элементе дуги. Переходы бывают двух видов:

1) с выбором в качестве текущего обрабатываемого символа очередного символа входной строки (предложения);
2) без изменения текущего обрабатываемого символа. По аналогии с программированием первый переход будем
называть передачей управления (ПУ), а второй—переходом к подпрограмме (ПП).

Действия и *формы,* встречающиеся в сети, представляются в виде «польской нотации», нотации, в которой функция представляется как список, заключенный в скобки, первым элементом его является наименование функции, а остальными элементами — аргументы функции.

Мы определим три типа *действия,* которые присваивают значение *формы* наименованию регистра. Действия различаются по тому, на каком уровне они выполняют присваивание (ПРТЕК—присваивание на текущем уровне, ПРПОГР — присваивание на уровне погружения, т. е. уровне более глубоком, чем текущий, ПРВЫТ — присваивание на уровне выталкивания).

Формы, так же как и условия, можно записывать в виде функций языка программирования (например, LISP). Приведенные в табл. 8.4 типы форм составляют базовое множество, достаточное для того, чтобы проиллюстрировать основные свойства РАСП.

ЗНАЧ есть функция, чьим значением является содержимое указанное в форме регистра. Значением формы А является текущее слово входного предложения. В случае ПОГР значением формы Δ является уровень, получаемый при возврате из ПОГР.

СВОЙСТВО есть функция, которая определяет значение свойства, указанного в форме для текущего слова входного предложения.

Форма ДЕР (дерево) используется для построения фрагмента или полного дерева (структуры) разбора. Фрагмент задается использованием скобок, наименований символов и указателей параметров, изображаемых знаком + (см. ниже пример). Значением формы ДЕР является фрагмент дерева, в которое на место параметров подставлены значения регистров, указанных в форме. Параметры ставятся в соответствие регистрам следующим образом: содержимое первого регистра замещает первый символ +, содержимое второго регистра — второй символ и т. д. Кроме того, форма Δ (если она присутствует в ДЕР) заменяется на соответствующее ей значение. Три оставшиеся формы предназначены также для построения структур. Форма СПИСОК — создает список из значений аргументов, ОБЪЕДИНИТЬ — объединяет два списка в один, ИМЯ — производит как значение (невычисленные) аргументы формы. Отметим, что три

последних формы являются встроенными функциями языка LISP (LIST, APPEND, QUOTE).

При выполнении анализатора для конкретной грамматики для создания более гибкой системы (чем описанная) могут быть введены дополнительные форматы дуг, увеличена гибкость функции ДЕР и т. п. Вообще целесообразно **множество условий, действий и форм оставить открытым для расширений в ходе эксперимента.** Заметим, что формат дуг и действий вместе с произвольными выражениями LISP для изображения условий и форм обеспечивает модель, эквивалентную по мощности машине Тьюринга и, следовательно, полную в теоретическом смысле.

В табл. 8.5 приведен пример записи во введенном формализме фрагмента грамматики, введенной на рис. 8.11.

Таблица 8.5

Фрагмент грамматики, представленной в расширенной сети переходов

```
((S (ПОГР NP | Усл
      (ПРТЕК S Δ)
      (ПРТЕК ТИП (ИМЯ ПОВЕСТВ))
      (ПУА1))
      (КАТ AUX Усл
      (ПРТЕК AUX Δ)
      (ПРТЕК ТИП (ИМЯ ВОПР))
      (ПУА2)))
 (A1  (ПОГР VP | Усл
      (ПРТЕК AUX ø)
      (ПРТЕК VP Δ)
      (ПУА5))
      (КАТ AUX Усл
      (ПРТЕК AUX Δ)
      (ПУА4)))
 (A2  (КАТ Ч Усл
      (ПУА3)))
 (A3  (ПОГР NP | Усл
      (ПРТЕК S Δ)
      (ПУА4)))
 (A4  (КАТ V | Усл
      (ПРТЕК V Δ)
      (ПУА5)))
 (A5  (ВЫТ (ДЕР (S+++| (VP+)) тип, S, AUX, Δ) Усл)
      (ПОГР NP | Усл
      (ПРТЕК VP (ДЕР (VP (V+)Δ) V))
      (ПУА6)))
 (A6  (ВЫТ (ДЕР (S++++), тип, S, AUX, VP) Усл)
      (ПОГР PP | Усл
      (ПРТЕК VP (ОБЪЕДИН (ЗНАЧ VP (Список Δ)))
      (ПУА6))))
```

Рассмотрим действие РАСП, введенной в табл 8.5 для предложения «Будет ли Джон строить дом?»

В качестве пояснения приведем содержимое регистров в соответствующих состояниях:

S: AUX:=будет;
ТИП: = ВОПР;
A2:
A3: *S:=(NP* Джон);
A4: *V* =строить;
A5: *VP:=(VP{V* строить)(*NP* дом));
A6: (S ВОПР(*NP* Джон) будет *(VP(V* строить) *(NP* дом))).

Полученную в состоянии А6 списочную структуру, соответствующую глубинному представлению исходного предложения, изобразим на рис. 8.12.

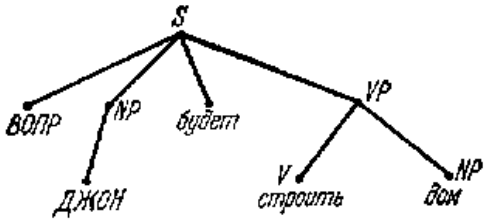


Рис. 8.12. Глубинная структура предложения.

В анализаторах НС-грамматик полученное структурное описание предложения соответствует алгоритму разбора, т. е. построение структуры происходит одновременно с применением грамматических правил. В РАСП эти процессы разделены. Это дает возможность некоторым конституентам, найденным в ходе разбора, появляться в окончательной структуре несколько раз или ни разу, и их место может отличаться от места в поверхностной структуре. Кроме того, структурное описание, присвоенное некоторой конституенте в ходе разбора, может быть впоследствии изменено. Эти свойства плюс способность вставлять проверку произвольных «условий» позволяет строить глубинную структуру в то время, когда анализатор выполняет переходы, соответствующие поверхностной структуре предложения.

Перечислим основные свойства РАСП, которые делают их привлекательными для использования в качестве модели естественного языка.

1. Наглядность. НС-грамматики являлись очень популярной моделью грамматики, несмотря на их неадекватность для управления

некоторыми свойствами естественного языка, во многом благодаря их наглядности. Теория трансформационных грамматик описывает практически все свойства естественного языка, но в том виде, в котором она существует, эта теория потеряла наглядность НС-грамматик. В трансформационной грамматике (ТГ) эффект отдельного правила связан с его взаимодействием с другими правилами, и требуется сложнейший анализ для определения эффекта и целей данного правила. РАСП, обеспечивая мощность ТГ, во многом сохраняет наглядность НС-грамматик.

2. Генеративная мощность. Как уже указано выше, РАСП имеет мощность машины Тьюринга, и при этом операции, выполняемые в РАСП, являются «естественными» для анализа языка. РАСП в отличие от ТГ (ориентированной на генерацию) с одинаковым успехом может использоваться как для генерации (формирования ответа), так и для распознавания (синтаксического анализа) предложений.

3. Эффективность представления. РАСП в отличие от НС-грамматик имеет средства для явного указания общих частей многих правил, что дает более эффективное представление. Объединение общих частей позволяет не только более компактно представлять грамматику, но и устраняет излишнюю обработку при разборе предложения (за счет уменьшения количества сопоставлений при определении применимости правил в течение разбора).

Отметим, что в РАСП возможно объединение подобных частей.

Пусть требуется представить в виде сети следующие подобные правила: *S→ABCDK, S→PBCTK*. На рис. 8.13 приведен пример объединения подобных частей этих правил.

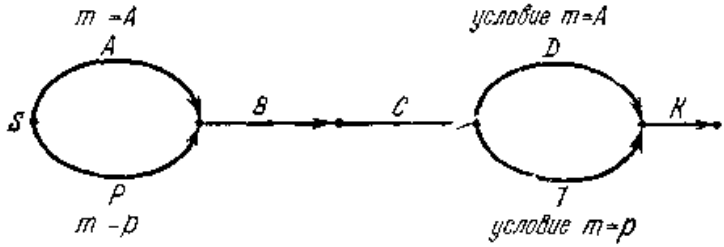


Рис. 8.13. Объединение подобных частей в расширенной сети переходов.

Для обоих правил строится единая сеть. Правила различаются благодаря действиям на дугах *A* и *P* и условиям на дугах *D* и *T*. Процесс объединения подобных частей делает грамматику более

компактной, но увеличивает время разбора (за счет внесения дополнительных действий и условий).

4. Эффективность анализа. Как уже указывалось, эффективность вытекает из объединения общих частей и способности изменять построение фрагмента дерева (без повторного просмотра входного предложения) или откладывать его построение.

5. Гибкость экспериментирования. Гибкость достигается за счет того, что множество операций оставлено открытыми и может пополняться в ходе экспериментирования. Следует отметить, что явное построение структуры предложения (с помощью действий) позволяет использовать РАСП не только для построения глубинных структур предложения, но и для других видов представления таких, как грамматики зависимостей, падежные грамматики.

Рассмотрим алгоритмы разбора, которые можно использовать при представлении грамматики в виде РАСП. Как мы уже указывали, РСП эквивалентна автомату с магазинной памятью. Поэтому многие существующие алгоритмы для НС-грамматик могут быть непосредственно сведены к РСП. Кроме того, один из наиболее эффективных алгоритмов для КС-грамматик (п.8.4 1) может быть приспособлен с небольшими модификациями для использования в РСП.

Использование алгоритма Эрли для РАСП (а не для РСП) является более сложным делом. Действительно, для переходов, зависящих от содержимого регистров, трудно определить эквивалентные конфигурации и объединить их при дальнейшей обработке. Кроме того, использование регистров и действий при построении структур усложняет задачу выбора представления для объединенных конфигураций. Поэтому прямолинейно расширить алгоритм Эрли не удается. Однако, если в РАСП делать явные различия между флаговыми регистрами (т. е. регистрами, содержащими только условия, выбранные из конечного словаря) и регистрами содержания (содержащими произвольные структуры), и ограничить условия и действия, приписанные дугам, таким образом, что:

1) действия могут ссылаться только на флаговые регистры и символы во входной строке;

2) на действия и условия наложить ограничения по времени обработки,

3) существует только один регистр содержания, изменяемый функцией ДЕР или конечным состоянием (содержимое его не анализируется),

тогда можно построить версию алгоритма Эрли с указанными ранее временными характеристиками. Однако если мы ослабим перечисленные условия, то временная граница превзойдет (Kn^3)

(например, одно условие и действие на дуге может потребовать больше, чем n^3 шагов).

Для многих применений нет необходимости получать представления всех возможных разборов входного предложения. В частности, в нашем случае более важно выбрать «наиболее вероятный» разбор в данном контексте, но сделать это наиболее быстрым образом. В таких применениях последовательный подход (с подходящим механизмом для выбора анализа, рассматриваемого первым) более предпочтителен, чем параллельный, так как он в большинстве случаев позволяет избежать необходимости следовать всем альтернативам. Повторный анализ тех же самых подстрок входной строки при альтернативных разборах можно устранить путем запоминания предыдущих результатов.

Успех последовательного подхода во многом зависит от качества механизма, осуществляющего выбор «наиболее вероятного» разбора. В РАСП некоторые из таких механизмов могут быть добавлены естественным образом. Упорядочением выбора дуг, исходящих из данного состояния, можно навязывать соответствующий порядок анализа. При создании грамматики можно подбором упорядочения дуг пытаться выделить наиболее вероятные ситуации. Кроме того, повторением в РАСП одной и той же дуги в виде нескольких дуг с разными условиями возможно сделать упорядочение этих дуг зависящим от особенностей обрабатываемого предложения, в частности, сделать зависящим от семантических свойств слов, встречающихся в предложении.

Заканчивая рассмотрение методов синтаксического анализа, укажем, что алгоритмы разбора могут рассматриваться как процедуры поиска на графе. Процедура поиска начинается с пустого дерева разбора. Каждый узел дерева поиска соответствует частному разбору предложения, а дуга соответствует добавлению нового уровня к дереву разбора. Различные методы синтаксического анализа отличаются способом, которым они добавляют новые дуги к дереву разбора. Алгоритмы «сверху—вниз» добавляют дуги на вершину дерева разбора и работают в направлении «вниз», в то время как процедуры «снизу — вверх» добавляют дуги в основание дерева и работают в направлении «вверх». Последовательные алгоритмы разбора соответствуют поиску «сначала в глубину», а параллельные — поиску «сначала в ширину». Однако хорошо известно, что большей эффективностью обладают стратегии, управляемые эвристиками.

Приведем примеры подобных эвристик:

1) избегать выполнения одного и того же частного разбора (т. е. разбора подстроки входного предложения) более одного раза;

2) избегать определенных разборов, которые не могут привести к успеху (определяются просмотром вперед, разделением предложения на части и т. п.);
3) устранять семантически нелепые анализы (проверкой части анализа на соответствие базовым данным, обычно при анализе «снизу—вверх»);
4) использовать оценочные функции и другие механизмы управления поиском наиболее перспективного пути.

8.5. Семантическая интерпретация

8.5.1. Общие сведения о семантической интерпретации

Входом для этапа семантической интерпретации (СИ) является глубинная структура обрабатываемого предложения, точнее структуры, так как на этапе синтаксического анализа не удается (в общем случае) получить однозначную интерпретацию предложения. Задачей СИ явтяется понять входное предложение. **Под словом «понять» мы подразумеваем не только выражение смысла предложения в известных нам понятиях, но и установление связи данного предложения с известными нам фактами**. Поэтому мы определяем задачи этапа семантической интерпретации следующим образом:
— получить однозначное представление входного предложения (интерпретировать его) в терминах базовых данных;
— объединить входное предложение с базовыми данными.
Этап СИ может выполняться как параллельно с процессом синтаксического анализа, так и после его завершения. В первом случае подпрограммы интерпретации добавляются к некоторым или всем правилам грамматики и выполняются во время применения грамматического правила при синтаксическом анализе. Достоинством этого подхода является то, что результат семантической интерпретации может направлять синтаксический анализ.
Второй подход имеет то преимущество, что глубинная структура, полученная на уровне синтаксического анализа, может просматриваться многократно и в произвольном порядке, что облегчает задачу СИ.
Следует отметить, что еще не существует общей теории процесса семантической интерпретации. Процессы СИ существенно зависят от ряда факторов и в первую очередь от внутреннего представления

базовых данных, определяющего общность или ограниченность вывода в системе (см. п. 8.3.3 и п. 8.3.4).

В данном параграфе мы сначала опишем общую постановку задачи СИ, а затем рассмотрим ее конкретные реализации отдельно для представления в виде семантических сетей (системы с ограниченной логикой) и в виде формул исчислении предикатов (системы с общим выводом).

В связи с распространенностью в естественных языках явления многозначности слов, словосочетаний, предложений, раскрытие многозначности при обработке текстов на ЭВМ приобретает особое значение. Раскрытие многозначности некоторых предложений невозможно без ссылок на контекст или без обращения к общим знаниям о мире. Например, предложение «Он пошел в парк с девушками» может быть понято как:

1) Он и девушки пошли в парк, или 2) Он пошел в парк, где есть девушки. Вероятно, двузначность данного предложения может быть раскрыта ссылкой на контекст. Однако только общие знания о мире могут раскрыть двузначность следующих выражений: «Гастроли балета на льду» и «Гастроли балета на Кавказе».

Значительно большее количество предложений является многозначным для машины, чем для человека, так как ЭВМ имеет более простые алгоритмы для раскрытия многозначности и меньшие знания о контексте и мире Например, приведенные выше предложения (о гастролях) будут двузначными для ЭВМ и скорее всего однозначными для всех людей. Существующие системы распознают только подмножество естественного языка, и это дает возможность раскрыть некоторые из многозначностей. Оставшиеся многозначности устраняются обычно одним из двух способов: 1) *методом семантических категорий;* 2) *ссылками на базовые данные.*

Семантические категории расширяют традиционную грамматическую классификацию слов введением подклассов. Отсутствие подобных подклассов приводит к катастрофическому возрастанию количества интерпретаций. Например, если слова A, B, C и D имеют по три значения, то предложение, содержащее все эти слова, может иметь 3×3×3×3=81 интерпретацию. Человек не строит все интерпретации, он выделяет наиболее разумные. Известно, что слово «коса» может обозначать следующие объекты: сельскохозяйственный инструмент, выступающий мыс, длинные волосы, уложенные определенным образом. С другой стороны, «зеленый» может обозначать: цвет, неспелый, неопытный. Однако, когда человек видит фразу «зеленая коса» он не рассматривает 9 интерпретаций, так как он знает, что «зеленый» в смысле «неспелый» применяется только к овощам и фруктам, а в

смысле «неопытный» только к людям. Таким образом, чтобы ввести эту информацию в ЭВМ, необходимо расширить традиционную грамматическую классификацию слов на классы. Кроме обычных классов: **существительное, прилагательное и т. д., вводят классы «одушевленный», «неодушевленный», «человеческий», «абстрактный», «физический» и т. п** Используя подобную информацию, удается выделить бессмысленные комбинации слов при выборе интерпретации.

Типичной системой, использующей такую информацию, является программа, разработанная Глазерфельдом. Анализ предложения осуществляется «снизу— вверх». В начале анализа каждому слову в предложении присваивается список «индексных категорий» из словаря (соответствуют описанным выше классам и подклассам). В течение анализа, каждой более высокой вершине дерева разбора назначается своя «индексная категория» на основании «индексных категорий» составляющих данной вершины. Назначение категории осуществляется специальной процедурой — «реклассификации».

Недостатком метода семантических категорий является тот факт, что с их помощью не могут быть раскрыты все многозначности. Например, они не могут раскрыть двузначность предложения «Он пошел в парк с девушками», так как для раскрытия двузначности необходима контекстуальная информация. Другим их недостатком является увеличение и усложнение словаря в связи с детальной классиффикацией слов. При методе семантических категорий информация в словаре будет частично дублировать информацию в базовых данных системы. Желание объединить эту информацию приводит к методу раскрытия многозначности путем *ссылки на базовые данные*. Например, в приведенном предложении семантический интерпретатор можем сформулировать вопрос к базовым данным: «Идут ли к ним девушки»? Ответ на этот вопрос может определить выбор интерпретации. В общем случае метод раскрытия многозначности заключается в том, что все имеющиеся интерпретации сопоставляются с базовыми данными и устраняются интерпретации, противоречащие базовым данным. Однако и после этого может остаться более одной интерпретации. Это наиболее трудный случай.

Можно указать несколько подходов к решению этой задачи:

1) выбрать интерпретацию, которая соответствует части базовых данных с наибольшими связями между вершинами;

2) выбрать интерпретацию, при объединении которой с базовыми данными к ним будет добавляться минимальное количество новых вершин и дуг;

3) выбрать наиболее разумную интерпретацию, используя некоторую «вероятностную» меру «разумности»;
4) запомнить все интерпретации в базовых данных и пытаться устранять многозначность за счет поступления новой информации (в частности, запрашивая необходимую информацию).

Отметим, что при обработке естественного языка процессы СИ, представления фактов, выполнения выводов и формирования ответа на вопросы могут быть существенно упрощены, если идентичные предложения, выражаемые различными поверхностными структурами, будут представлены единой концептуальной конструкцией. Одной из причин разнообразия поверхностных структур при единстве смысла является разнообразие «поверхностных глаголов», описывающих одну и ту же ситуацию. Шенк исходит из существования «глубинных» (канонических) глаголов, унифицирующих в глубинных структурах смысл многих поверхностных глаголов.

До сих пор мы, рассматривая вопросы семантической интерпретации, молчаливо предполагали, что на вход системы поступают отдельные предложения. Задача значительно усложняется, если требуется понять смысл связанного текста.

Перейдем к рассмотрению процессов СИ с учетом структуры представления базовых данных.

8.5.2. Семантическая интерпретация в системах с ограниченной логикой

При описании процесса СИ с ограниченной логикой мы будем основываться на реальной вопросно-ответной системе, воспринимающей естественный (английский) язык. Выбор реальной ВОС преследует цель показать состояние практики в решении одного из наиболее сложных этапов по обработке естественного языка — семантической интерпретации. Несмотря на то, что описываемая система воспринимает английский язык, мы там, где это возможно, будем для удобства читателей приводить примеры на русском языке. В случаях, где примеры отражают специфику английского языка, мы будем приводить английский текст, а в скобках давать его русский перевод.

В рассматриваемой системе входом для СИ является каноническая структура, полученная в результате обработки ограниченного естественного языка вариантом расширенной сети переходов (РАСП). Действия РАСП были подробно описаны в п. 8.4.3, поэтому мы сейчас остановимся только на механизме, позволяющем осуществлять перевод входного текста в каноническую структуру. Основой этой опе-

рации является включение в словарь определенной информации (табл. 8.6). В словаре каждый поверхностный глагол содержит оснозной вход, представляемый в виде L-«ГЛАГОЛ», где L— признак основного входа, а «ГЛАГОЛ» — конкретный глагол в инфинитиве. Список свойств поверхностного глагола содержит различную информацию: ссылку на каноническую форму данного поверхностного глагола, различные формы глагола, А-правила и Ф-правила (см. п.8.7). А-правила используются при анализе входного предложения и соотносят поверхностные (предложные и беспредложные — обозначаются БП) существительные, используемые с данным поверхностным глаголом, глубинным структурам (например, в смысле Филмора), связанным с соответствующим каноническим глаголом. А-правила представлены в виде списка триплетов; первый элемент в каждом триплете указывает предлог (или отсутствие его), с которым используется поверхностное существительное. Второй элемент триплета есть список семантических категорий (о назначении семантических категорий см. п. 8.5.1), к одной из которых должно относиться поверхностное существительное для того, чтобы удовлетворять глубинному падежному отношению (deep case relation), задаваемому третьим элементом триплета. Поясним структуру А-правила на примере: JOHN BOUGHT THE CAR FROM MARY (ДЖОН КУПИЛ АВТОМОБИЛЬ У МЭРИ). Как видно из табл. 8.6, поверхностному глаголу BUY (ПОКУПАТЬ) соответствует канонический глагол EXCHANGE (ОБМЕНИВАТЬ). Глагол ОБМЕНИВАТЬ определяется следующими глубинными отношениями (падежами): ПОКУПАТЕЛЬ (BUYER); ПРОДАВЕЦ (SELLER); ПОЛУЧЕННОЕ (THINGBT); ОТДАННОЕ (THINGGIVEN); МЕСТО (LOC); ВРЕМЯ (TIME).

Таблица 8.6

Фрагмент словаря

L-BUY (ПОКУПАТЬ) WORD CLASS (КЛАСС СЛОВА) CANNON-VB (КАНОН. ГЛ) INF (ИНФИНИТИВ) SG3 (ЕД. ЧИСЛО 3 ЛИЦО) PAST (ПРОШЕДШЕЕ) -EN -ING	 VERB (ГЛАГОЛ) EXCHANGE (ОБМЕНЯТЬ) BUY BUYS BOUGHT BOUGHT BUYING
P-RULES (А-Правила)	(ОК (HUMAN ORGANIZATION) BUYER) (БП (ЧЕЛОВЕК ОРГАНИЗАЦИЯ) ПОКУПАТЕЛЬ) (ОК (PHYSOBJ) THINGBT) (БП (ФИЗИЧ. ОБЪЕКТ) ПОЛУЧ.) (FROM (HUMAN ORGANIZATION) SELLER) (У (ЧЕЛОВЕЧ. ОРГАНИЗАЦИЯ) ПРОДАВЕЦ) (FOR (MONEY) THINGGIVEN) (ЗА (ДЕНЬГИ) ОТДАННОЕ) (AT (PLACE) LOC) (В (МЕСТО) МЕСТО) (IN (PLACE) LOC) (ОК (DAY TIME) TIME) (IN (DAY TIME) TIME))
G-RULES (Ф-правила)	((BUYER ACTIVE THINGBT (FROM SELLER) (FOR THINGGIVEN)) (THINGBT PASSIVE(FROM SELLER) (FOR THINGGIVEN)))
L-COST (СТОИТЬ) G-RULES	 ((THINGBT ACTIVE (BUYER) (THINGGIVEN)))

Продолжение табл. 8.6

L-PAY (ПЛАТИТЬ) G-RULES	((BUYER ACTIVE (SELLER) (THINGGIVEN) (FOR THINGBT)))
L-WHEN (КОГДА) WORD CLASS SEM	Q WORD (ВОПР. СЛОВО) (DAY DAYPART) (ДЕНЬ, ЧАСТЬ ДНЯ)
EXCHANGE SURF-VB (ПОВЕРХН. ГЛ.)	(L-BUY L-SELL L-PAY L-COST)

Из табл. 8.6 мы можем увидеть, что глагол BUY (ПОКУПАТЬ) для выражения глубинного отношения SELLER (ПРОДАВЕЦ) использует предлог FROM (У) и, кроме того требует, чтобы семантическая категория поверхностного существительного огиосилась к классу ЧЕЛОВЕЧЕСКИЙ или ОРГАНИЗАЦИЯ. Очевидно, что слово МЭРИ в рассматриваемом примере удовлетворяет обоим условиям и, следовательно, выражает отношение SELLER.

Предполагается, что в словаре каждое существительное и вопросительное слово (Q WORD) имеет специальное свойсво (SEM), указывающее семантическую категорию, к которой существительное или вопросительное слово принадлежит. Результатом разбора входного предложения с применением РАСП являются две (в случае вопроса три) вспомогательные структуры. Например, для предложения WHO BOUGHT A CAKE AT THE NEW BAKERY FROM THE BAKER? (КТО КУПИЛ ПИРОЖНОЕ В НОВОЙ БУЛОЧНОЙ У БУЛОЧНИКА?) мы получим три вспомогательные структуры:

Глагольная составляющая:

(CANON-VB EXCHANGE MODAL (TENSE PAST MOOD INTERROG CASE AFFIRM))

(КАНОН. ГЛ. ОБМЕНЯТЬ МОДАЛ. (ВРЕМЯ ПРОШ. НАКЛ. ИЗЪЯВИТ. ПАДЕЖ УТВЕРД)).

Составляющая существительных:

(OK(PHYSOBJ) (TOK L-CAKE DET INDEF NBR S))
(AT(PLACE) (TOK L-BAKERY DET DEF NBR S

MOD (AGE L-NEW)))
(FROM (HUMAN) (ТОК L-BAKER DET DEF NBR S))
(БП (ФИЗИЧ. ОБ.) (ОБОЗН. L-ПИРОЖНОЕ
АРТИКЛЬ НЕОПР. ЧИСЛО ЕДИНСТВ.))
(В(МЕСТО)(ОБОЗН. L-БУЛОЧНАЯ АРТИКЛЬ ОПРЕД.
ЧИСЛО ЕДИНСТВ. МОД.(ВОЗРАСТЬ-НОВЫЙ)))
(У (ЧЕЛОВЕЧ.) (ОБОЗН. L-БУЛОЧНИК АРТИКЛЬ
ОПРЕД. ЧИСЛО ЕДИНСТВ.)).

Составляющая вопросительного слова:
(ОК(HUMAN) (ТОК L-WHO))
(БП (ЧЕЛОВЕЧ.) (ОБОЗН. L-КТО)).

Глагольная составляющая указывает на канонический глагол, определяющий предложение, и на характеризации глагола (модальность МОДАЛ). Составляющие существительного определяют предложные и беспредложные (БП) существительные в порядке их появления во входном предложении. Каждая составляющая представляет собой триплет. Первый элемент триплета указывает синтаксическую форму (предлог или его отсутствие), в которой существительное появилось в предложении. Второй элемент определяет семантическую категорию существительного (скопированную из словаря). Третий элемент является списком, описывающим свойства данного существительного из входного предложения. Составляющая вопросительного слова представляется в виде триплета со структурой, аналогичной триплету существительного.

Окончательная стадия отображения входного предложения в каноническую структуру состоит в сопоставлении составляющих существительных и вопросительного слова с А-правилами поверхностного глагола входного предложения. Сопоставление заключается в поиске для каждой составляющей существительного А-правила, у которого первый элемент триплета совпадает с первым элементом составляющей, а вторые элементы образуют непустые пересечения. Глубинное отношение, указываемое при этом третьим элементом А-правила, связывается с третьим элементом составляющей (если оно еще не было связано с ранее сопоставляемой составляющей). Когда заканчивается процесс сопоставления составляющих существительных, переходят к сопоставлению составляющей вопросительного слова. Для вопросительного слова сопоставление делается аналогичным образом среди оставшихся несопоставленными отношений. Отличие состоит в том, что результирующей структуре назначается псевдоотношение Q (QUESTION— ВОПРОС), а подходящие отношения (в нашем примере это одно отношение— BUYER) собираются в список ARGS (АРГУМЕНТЫ).

Таким образом, каноническая структура, полученная для рассматриваемого входного предложения, имеет вид:

```
(CANNON-VB EXCHANGE MODAL (TENSE PAST
                MOOD INTERROG CASE AFFIRM)
SELLER (ТОК L-BAKER DET DEF NBR S)
(THINGBT (ТОК L-CAKE DET INDEF NBR S)
 LOC (ТОК L-BAKERY DET DEF NBR S MOD (AGE
                L-NEW))
Q (ARGS (BUYER) ТОК L-WHO))
(КАНОН. ГЛ. ОБМЕН ЯТЬ МОДАЛ. (ВРЕМЯ ПРОШ.
          НАКЛ. ИЗЪЯВИТ. ПАДЕЖ УТВЕРД.)
ПРОДАВЕЦ (ОБОЗН. L-БУЛОЧНИК АРТИКЛЬ
                ОПРЕД. ЧИСЛО ЕД)
 ПОЛУЧЕННОЕ (ОБОЗН. L-ПИРОЖНОЕ
                АРТИКЛЬ НЕОПР. ЧИСЛО ЕД)
МЕСТО (ОБОЗН. L-БУЛОЧНАЯ АРТИКЛЬ ОПРЕД.
          ЧИСЛО ЕД. МОД. (ВОЗРАСТ L-НОВЫЙ))
 ВОПРОС (АРГ. (ПОКУПАТЕЛЬ) ОБОЗН. L-КТО)).
```

Изложенный способ отображения входного предложения в каноническую структуру рассчитан на простые активные предложения без вложенных конструкций. Подход может быть расширен в направлении включения:

1) более сложных А-правил, явно выражающих упорядоченность существительных в списке составляющих;
2) более сложного алгоритма сопоставления, распознающего относительные предложения;
3) более разнообразных синтаксических форм и семантических категорий;
4) включение пассивных конструкций и вложенных предложений.

Перейдем теперь к вопросу объединения информации входного предложения с базовыми данными. Базовые данные, представленные в виде семантической сети, хранят информацию об объектах и отношениях между объектами.

На рис. 8.14. приведен пример текущего состояния семантической сети.

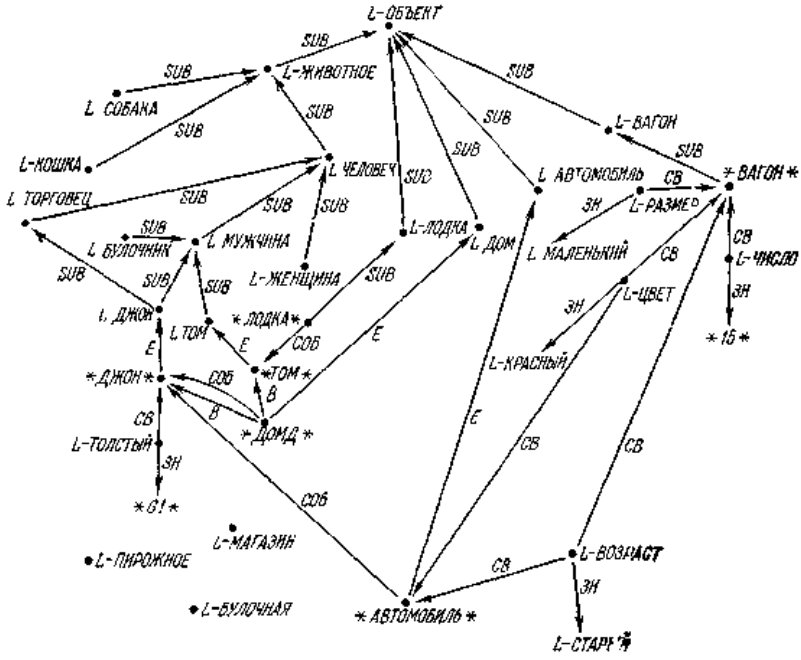


Рис. 8.14. Пример текущего состояния семантической сети.
SUB — ПОДМНОЖЕСТВО; Е — ЭЛЕМЕНТ; СОБ — СОБСТВЕННОСТЬ; В — НАХОДИТЬСЯ В; СВ — СВОЙСТВО; ЗН — ЗНАЧЕНИЕ.

В дальнейшем при описании сетей мы будем наравне с термином «вершины» использовать термин «узел». *Фактуальные вершины* будем представлять узлами, наименования которых начинаются и заканчиваются звездочкой (*), а *общие вершины*— узлами, наименованиям которых предшествует буква L. Рассмотрим узел *ДЖОН*, являющийся элементом L-ДЖОН и L-ТОРГОВЕЦ. L-ДЖОН является обозначением множества всех Джонов и подмножеством L-МУЖЧИНА (множества всех мужчин). L-ТОРГОВЕЦ есть множество всех торговцев. Следовательно, *ДЖОН* является узлом, представляющим конкретного человека, являющегося торговцем с именем Джон.

Этот Джон владеет конкретным домом *ДОМД* (что выражается отношением СОБСТВЕННОСТЬ). Джон находится в этом доме. Джон владеет старым красным автомобилем.

Для представления событий, происходящих во времени, используется понятие временной оси (рис. 8.15).

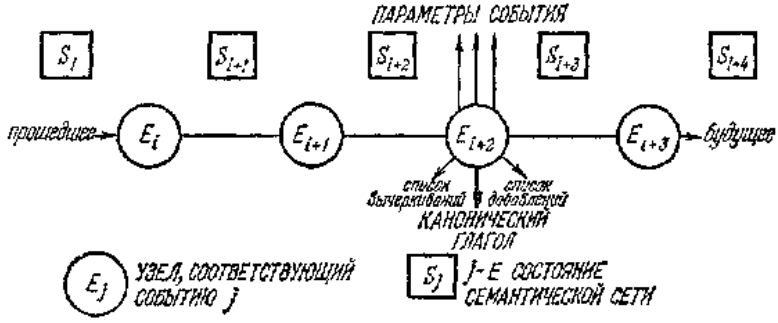


Рис. 8.15. Абстрактная временная ось.

Каждый узел на временной оси соответствует событию (предложению в естественном языке). В начальном состоянии сеть содержит только некоторую информацию из словаря. Состояние сети изменяется в соответствии с последовательностью событий, происходящих на временной оси. С каждым событием связывается список отношений (механизм связи будет определен ниже), имеющих место до совершения события и более неудовлетворяющихся после совершения события (так называемый список вычеркивания) и список отношений, которые не имели место до рассматриваемого события (или по крайней мере не было известно, что эти отношения выполняются), но которые удовлетворяются после выполнения события (список добавлений). Отметим, что появление события на временной оси изменяет не только отношения, существующие в семантической сети, но и может привести к добавлению новых узлов, если во входном предложении были объекты, отсутствовавшие в сети.

Рассмотрим в качестве примера изменения, которые вызовет в сети (см. рис. 8.14) событие: ДЖОН ОБМЕНЯЛ С ТОМОМ СВОЙ АВТОМОБИЛЬ НА ЛОДКУ ТОМА. Отношения (СОБСТВЕННОСТЬ *ДЖОН**АВТОМОБИЛЬ*) и (СОБСТВЕННОСТЬ *ТОМ** ЛОДКА*), существовавшие в сети на рис. 8.14 больше не являются истинными и должны быть заменены на отношения (СОБСТВЕННОСТЬ *ДЖОН ** ЛОДКА*) и (СОБСТВЕННОСТЬ * ТОМ* «АВТОМОБИЛЬ*). Указанные изменения в сети изображены на рис. 8.16 (состояние 2).

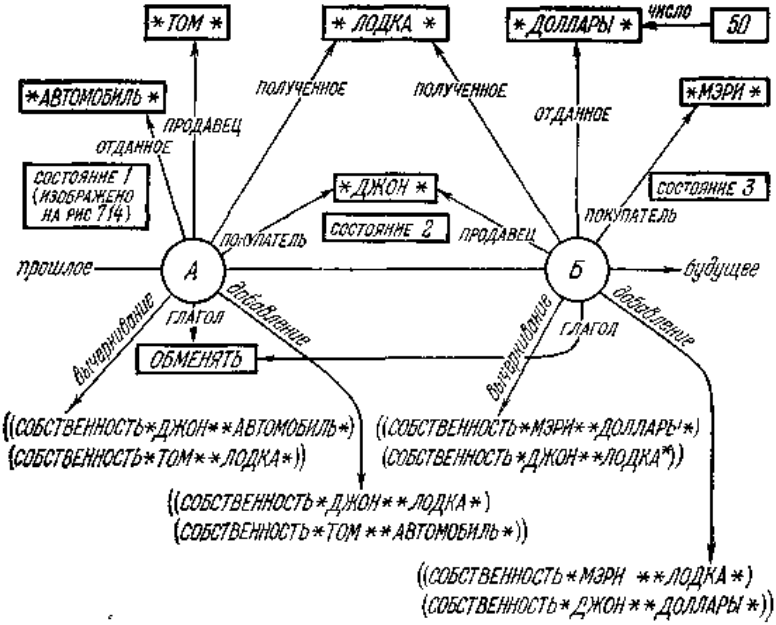


Рис. 8.16. Пример состояния временной оси при поступлении двух событий (вспомогательные узлы для простоты не показаны).

Так как события преобразуют одно состояние сети в другое, можно рассматривать их как операторы. Следуя этой трактовке, канонический глагол можно рассматривать как процедуральное событие (или оператор), которое преобразует состояние сети (являющееся неявным параметром) и множество явных параметров в новое состояние сети. Процедуральные события представляются аналогично операторам, используемым в системе STRIPS (см. п. 7.2).

Неотъемлемой частью любого оператора STRIPS является список предусловий, который используется пользователем для определения законной последовательности действий. Так как в рассматриваемой системе предполагается, что события следуют в хронологической последовательности, то исчезает необходимость в списке предусловий для определения последовательности событий (операторов). Существует однако следующая проблема. Пусть на вход поступили предложения: ДЖОН КУПИЛ ЧАСЫ В МАГАЗИНЕ, ЗАТЕМ ДЖОН КУПИЛ ПИРОЖНОЕ В НОВОЙ БУЛОЧНОЙ. Заметим, что Джон покупал часы в магазине, а в следующем событии он покупал пирожное в булочной. До того как событие по покупке пирожного

может иметь место, Джон должен перейти из магазина в булочную (предусловие для события в булочной). Таким образом, мы видим, что необходимость в удовлетворении предусловиям остается даже при заданной последовательности событий. Предусловия определяют неспецифицированные вспомогательные события, которые необходимы для выполнения специфицированных (указанных в тексте) событий.

Сущность этих неспецифицированных вспомогательных событий в значительной степени определяется предусловиями специфицированных событий. Так, в приведенном примере до того, как ДЖОН КУПИЛ ПИРОЖНОЕ В НОВОЙ БУЛОЧНОЙ необходимо выполнить некоторое событие, устраняющее отношение (В*ДЖОН ** МАГАЗИН*) и добавляющее отношение (В*ДЖОН ** БУЛОЧНАЯ *). (В общем случае событию ОБМЕНЯТЬ должно предшествовать событие, переносящее участников обмена в место обмена и т. д.)

Таким образом, процедуральное событие приводит к вызову вспомогательного события и созданию на временной оси соответствующих им двух узлов. Процедуральное событие подставляет списки добавления и вычеркивания неспецифицированного события, которое преобразует текущее состояние сети в состояние, удовлетворяющее предусловиям главного события. То, что такое событие должно существовать, следует из заданной последовательности специфицированных событий.

Заметим, что описанный способ обработки процедурального события является одним из возможных вариантов представления события в виде скелета (фрейма).

Рассмотрим на примере, как осуществляется вызов и обработка процедурального события. Пусть входное предложение имеет вид:

AT THE NEW BAKERY JOHN BOUGHT A CAKE
FROM THE BAKER
(В НОВОЙ БУЛОЧНОЙ ДЖОН КУПИЛ ПИРОЖНОЕ
У БУЛОЧНИКА).

После этапа синтаксического разбора и приведения к канонической структуре оно примет вид:

(КАНОН. ГЛ. ОБМЕНЯТЬМОДАЛ. (ВРЕМЯ ПРОШ.
НАКЛ. ИЗЪЯВИТ. ПАДЕЖ УТВЕРДИТ.)
ПРОДАВЕЦ (ОБОЗН. L-БУЛОЧНИК АРТИКЛЬ
ОПРЕД. ЧИСЛО ЕД.)
ПОКУПАТЕЛЬ (ОБОЗН. L-ДЖОН ЧИСЛО ЕД.)
ПОЛУЧЕННОЕ (ОБОЗН. L-ПИРОЖНОЕ АРТИКЛЬ
НЕОПР. ЧИСЛО ЕД.)
МЕСТО (ОБОЗН. L-БУЛОЧНАЯ АРТИКЛЬ

ОПРЕД. ЧИСЛО ЕД. МОД. (ВОЗРАСТ L-НОВЫЙ))).

Канонический глагол указывает, какая процедура должна быть вызвана (табл. 8.7).

Таблица 8.7

Представление процедурального события «обменять»

```
ПРОЦЕДУРА ОБМЕНЯТЬ (продавец, покупатель, полученное,
                                        отданное, место);
СПИСОК ВЫЧЕРКИВАНИЙ ВСПОМОГАТЕЛЬНОГО СОБЫТИЯ:
        ((В продавец *) (В покупатель *) (В полученное *)
         (В отданное *) (СОБСТВЕННОСТЬ * полученное)
         (СОБСТВЕННОСТЬ * отданное));
СПИСОК ДОБАВЛЕНИЙ ВСПОМОГАТЕЛЬНОГО СОБЫТИЯ:
        ((В продавец место) (В покупатель место)
         (В полученное место) (В отданное место)
         (СОБСТВЕННОСТЬ продавец полученное)
         (СОБСТВЕННОСТЬ покупатель отданное));
СПИСОК ВЫЧЕРКИВАНИЙ ГЛАВНОГО СОБЫТИЯ:
        ((СОБСТВЕННОСТЬ покупатель отданное)
         (СОБСТВЕННОСТЬ продавец полученное));
СПИСОК ДОБАВЛЕНИЙ ГЛАВНОГО СОБЫТИЯ:
        ((СОБСТВЕННОСТЬ покупатель полученное)
         (СОБСТВЕННОСТЬ продавец отданное)).
```

Параметры процедуры (возможно не все) в явном виде вырабатываются на стадии разбора. В приведенном примере это: продавец, покупатель, полученное, отданное, место. Остальным параметрам процедуры, не получившим на стадии разбора конкретных значений, присваивается значение NIL. Затем с каждым параметром должен быть связан некоторый узел сети. Если подходящего узла в сети не существует, то такой узел должен быть создан. Эту функцию выполняет специальная программа НИС (найти или создать), чьими аргументами являются параметры выбранной процедуры. Например, если артикль параметра является неопределенным, то НИС просто создает новый узел в сети, удовлетворяющий списку свойств параметра. Так, для А САКЕ, упомянутого во входном предложении, НИС создает новый узел, не заботясь о том, связан он или нет с некоторым понятием ПИРОЖНОЕ, существующим в сети. (Впоследствии необходимо определить различные узлы, выражающие одну и ту же сущность. Объединение этих узлов выполняется специальной функцией.) Если у параметра определенный (или специфированный) артикль, то НИС пытается найти в сети

соответствующий этому параметру узел. Если найдено больше одного узла, то выбирается тот, который упоминался последним. Если не найдено ни одного узла, то создается новый узел.

Значением свойства ОБОЗН, указанного в списке свойств параметра, является наименование узла, представляющего множество объектов. Значением параметра является элемент из этого множества. В нашем примере для параметра ПРОДАВЕЦ наименованием множества является L-БУЛОЧНИК (именующий множество всех булочников). Следовательно, значением параметра ПРОДАВЕЦ должен быть элемент из множества L-БУЛОЧНИК. Заметим, что слова типа L-БУЛОЧНИК, L-ДЖОН, L-ПИРОЖНОЕ введены и в словарь (для целей разбора), и в сеть перед началом функционирования системы. Кроме того, в сеть введены и определенные примитивные отношения между подобными словами, например, (ПОДМНОЖЕСТВО L-ДЖОН L-МУЖЧИНА). Всякий раз, когда упоминается некоторый *ДЖОН*, он становится элементом множества L-ДЖОН, являющегося в свою очередь подмножеством множества L-МУЖЧИНА.

Параметры указанного входного предложения будут обработаны следующим образом:

1) Программа НИС ищет в сети *x* такое, что для него справедливо отношение (ЭЛЕМЕНТ *x* L-БУЛОЧНИК). Не найдя такого *x,* программа создает узел * БУЛОЧНИК *, и этот узел становится значением параметра ПРОДАВЕЦ. (В течение этого процесса отношение (ЭЛЕМЕНТ * БУЛОЧНИК * L-БУЛОЧНИК) вводится в сеть).

2) Программа НИС ищет такое *x,* что (ЭЛЕМЕНТ *x* L-ДЖОН). Она находит узел «ДЖОН», который и становится значением параметра ПОКУПАТЕЛЬ.

3) В связи с тем, что у параметра ПОЛУЧЕННОЕ неопределенный артикль, НИС создает новый узел *ПИРОЖНОЕ* такой, что для него выполняется отношение (ЭЛЕМЕНТ * ПИРОЖНОЕ* L-ПИРОЖНОЕ), и он является значением параметра ПОЛУЧЕННОЕ.

4) В связи с присутствием у параметра МЕСТО модификатора (прилагательного) работа программы НИС усложняется. НИС ищет *x* такое, что (ЭЛЕМЕНТ *x* L-БУЛОЧНАЯ) и (ВОЗРАСТ НОВЫЙ *x)*. Не найдя такого узла, НИС создает его (*БУЛОЧНАЯ*) и делает значением параметра МЕСТО.

Как только значения параметров определены, система вызывает процедуру ОБМЕНЯТЬ для того, чтобы обработать поступившее событие. Важно отметить, что отношения, вводимые в сеть в процессе определения значений параметров, не вводятся на списки вычеркивания и добавления узлов временной оси.

Итак, обращение к процедуральному событию (табл. 8.7) имеет вид: ОБМЕНЯТЬ (* БУЛОЧНИК ** ДЖОН ** ПИРОЖНОЕ * NIL * БУЛОЧНАЯ *), где NIL присваивается параметру процедуры ОТДАННОЕ, отсутствующему в поступившем событии. Эффект, вызываемый обращением к процедуре, представлен на рис. 8.17.

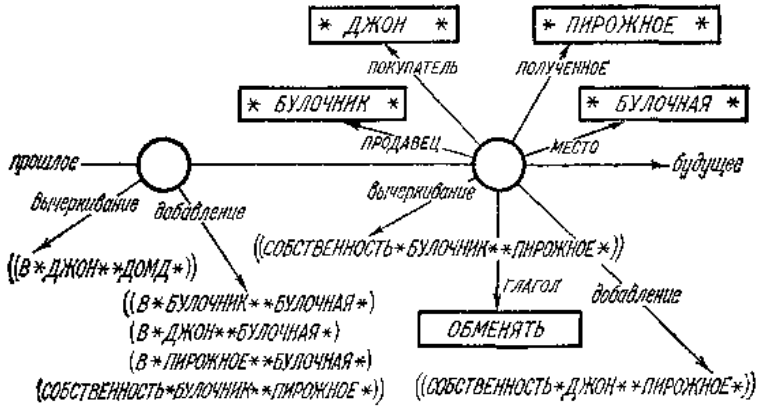


Рис. 8.17. Узлы временной оси, созданные процедуральным событием.

Рассмотрим этапы выполнения процедуры:

1) На временной оси создается узел, соответствующий вспомогательному событию. Так как природа события неизвестна, с узлом не связывается ни канонический глагол, ни параметры события.

2) Рассматривается список вычеркиваний вспомогательного события (табл. 8.7). Первым рассмотрим триплет (В ПРОДАВЕЦ *). Делается попытка сопоставить это отношение с отношениями, существующими в сети. Звездочка, стоящая на месте третьего элемента триплета, позволяет устанавливать соответствие с любым объектом сети. Параметру «продавец» при вызове процедуры сопоставлен узел *БУЛОЧНИК *. Итак, нам надо искать отношение в виде (В*БУЛОЧНИК*—). Такого отношения в сети нет, и поэтому никакие действия не предпринимаются. Следующим рассматривается отношение (В ПОКУПАТЕЛЬ*). Подстановка фактических параметров (при вызове процедуры) преобразует его в (В* ДЖОН * —). Этому отношению в сети (рис. 8.14) соответствует отношение (В*ДЖОН **ДОМД*). Следовательно, (так как мы рассматриваем список вычеркиваний) отношение (В * ДЖОН * * ДОМД *) вычеркивается из сети и устанавливается в список вычеркиваний узла, представляющего на временной оси вспомогательное событие. Остальным отношениям, перечисленным в списке вычеркиваний процедуры, не удается

сопоставить отношения сети, и поэтому они не вызывают никаких действий.

3) Рассматривается список добавлений вспомогательного события (табл. 8.7). Первое рассматриваемое отношение (В ПРОДАВЕЦ МЕСТО) после подстановки фактических параметров преобразуется в отношение (В*БУЛОЧНИК ** БУЛОЧНАЯ *). Это отношение добавляется к сети и к списку добавлений вспомогательного узла временной оси. Аналогичным образом добавляется следующие отношения (В * ДЖОН ** БУЛОЧНАЯ *), (В * ПИРОЖНОЕ ** БУЛОЧНАЯ *) и (СОБСТВЕННОСТЬ*БУЛОЧНИК ** ПИРОЖНОЕ*). Триплеты, содержащие параметр «отданное», не вызывают никаких действий, так как этот параметр неопределен поступившим событием.

4) На временной оси устанавливается узел, соответствующий главному событию. С данным узлом связываются параметры процедуры ОБМЕНЯТЬ и канонический глагол ОБМЕНЯТЬ.

5) Рассматривается список вычеркиваний главного события (табл. 8.7) и вырабатывается отношение (СОБСТВЕННОСТЬ * БУЛОЧНИК ** ПИРОЖНОЕ *), устраняемое из сети и заносимое в список вычеркиваний главного события.

6) Обрабатывается список добавлений главного события и добавляется отношение (СОБСТВЕННОСТЬ * ДЖОН * * ПИРОЖНОЕ*).

Описанной процедурой первоначальное состояние сети (рис. 8.14) преобразуется во вспомогательное состояние, в котором Джон и булочник находятся в булочной, и булочник владеет пирожным. Затем это вспомогательное состояние преобразуется в состояние, в котором Джон и булочник остаются в булочной, но владельцем пирожного становится Джон.

Итак, мы рассмотрели основные этапы семантической интерпретации в системах с ограниченной логикой при представлении базы данных в виде семантической сети. Перейдем теперь к описанию СИ при представлении базовых данных в исчислении предикатов.

8.5.3. Семантическая интерпретация в системах с общим выводом

Описание СИ систем с общим выводом дадим на базе вопросно-ответной системы. Основанием для выбора нами этой системы является следующее:

1) законченность системы (т. е. в системе реализованы все задачи диалога);

2) система разработана для отправителя (получателя);

3) система позволяет продемонстрировать альтернативные подходы в области грамматики, внутреннего представления и взаимосвязи этапов семантической интерпретации и синтаксического анализа (СА).

В связи с тем, чго в данной системе этапы СИ и СА выполняются параллельно (а не последовательно), описанию СИ необходимо предпослать краткие сведения о синтаксическом анализе

Синтаксический анализ системы основан на трансформационной грамматике (п. 8.2.3) подмножества английского языка, включающего повелительные, повествовательные и вопросительные предложения.

Задачи трансформационных правил состоят в преобразовании пассива в актив, вопросительных предложений в повествовательные, отображении множественных форм существительных и глаголов в единые формы.

Так, например, при обработке предложения с пассивной конструкцией «THE TALL BOX WAS PUSHED BY JOHN» (высокий ящик толкался Джоном), задачей трансформационной компоненты является определение необходимости применения трансформации (пассив→актив), которая переведет данное предложение в активную форму: «JOHN PUSHED THE TALL BOX» (Джон толкал высокий ящик). Затем к работе приступает базовая компонента. Анализатор базовой компоненты записывается в виде правил продукции, имеющих следующий вид:

L1: α |→β | γ * *L2;*

где *L*1 и L2— метки, α и β есть строки, → указывает операцию замещения, | — разделитель, γ есть последовательность семантических продукций, * указывает операцию «ЧТЕНИЕ» (выбора очередного слова из входной строки и размещение его на вершине стека синтаксического анализатора), ; — знак, разъединяющий продукции. *L*1, →, β, γ,* являются необязательными символами, в то время как |, α, *L*2 являются обязательными для каждой продукции. Рассмотрим, как осуществляется выполнение продукции. Предполагается, что в начале работы в верхний уровень синтаксического стека устанавливается первый (левый) символ входного предложения. Пусть управление передано продукции, помеченной меткой *L*1. Символ α правила *L*1 сопоставляется с верхними уровнями стека. Если сопоставление закончилось успешно, то:

1) часть стека, соответствующая α, заменяется на β (свободный класс переменных становится связанным в смысле исчисления предикатов);

2) выполняется последовательность γ (последовательность семантических продукций правила);

3) если * присутствует в продукции *L*1, то выполняется операция «чтения»;

4) совершается переход к продукции, с меткой *L2*.

Если сопоставление строки α правила *L*1 со стеком неудачно, то управление передается следующей продукции в последовательности. Операция сопоставления понимается в смысле сопоставления с образцом. Образец (α-строка правила) может содержать терминальную константу, класс переменных, определенных в терминах терминальных констант или в терминах булевской комбинации других классов. Образец может иметь вид $1, который сопоставляется с произвольной одиночной составляющей. Образец $ сопоставляется с любым числом произвольных составляющих.

Продукции трансформационной компоненты имеют такую же форму, как продукции базовой компоненты, за исключением того факта, что сопоставление осуществляется не со стеком, а с самим предложением (просмотром его слева направо). Любой элемент образца, взятый в кавычки, указывает на то, что сопоставление выполняется на уровне букв частного слова, а не на лексическом уровне. Таким образом могут быть выполнены проверки на множественное число и на стандартные суффиксы и префиксы.

Семантические продукции идентичны синтаксическим по форме и выполнению. Отличие состоит в том, что в семантических продукциях не используется операция * и, кроме того, семантические продукции выполняются на семантическом стеке.

В табл. 8.8 приведены примеры предложений ограниченного английского языка, используемые при обращении к отправителю. Для

Таблица 8.8

Образцы перевода предложений естественного языка в формулы исчисления предикатов

Предложения	Формулы
Повествовательные	
1) Все люди смертны	$\forall x$ (ЕСТЬ (x, человек) → ЕСТЬ (x, смертен))
2) Существуют высокие представители мужского пола, не являющиеся мальчиками	$\exists x$ (ЕСТЬ (x, высокий) ∧ ЕСТЬ (x, мужчина) ∧ ¬ЕСТЬ (x, мальчик))
3) У Джона две руки	ИМЕТЬ (Джон, рука, 2)
4) Любой зеленый объект является высоким, узким ящиком	$\forall x$ (ЕСТЬ (x, зеленый) → ЕСТЬ (x, высокий) ∧ ЕСТЬ (x, узкий) ∧ ЕСТЬ (x, ящик))
5) Высокий ящик толкался Джоном	$\exists s, x, y, z$ (РАВНО (S, ТОЛКАТЬ (ДЖОН, $x, y, z, S_{нач}$)) ∧ ЕСТЬ (x, высокий) ∧ ЕСТЬ (x, ящик) ∧ ВРЕМЯ (S, прошл.))
Вопросительные	
6) Является ли Джон мужчиной?	ЕСТЬ (Джон, мужчина)
7) Сколько пальцев имеет Джон?	$\exists x$ (ИМЕТЬ (Джон, палец, x))
8) Есть ли слева от Вас какие-нибудь ящики?	$\exists x$ (ЕСТЬ (x, ящик) ∧ СЛЕВА (x, Отправ))
9) Когда вы будете толкать ящик?	$\exists s, t, x, y, z$ (РАВНО (S, ТОЛКАТЬ (Отправ., $x, y, z, S_{нач}$)) ∧ ЕСТЬ (x, ЯЩИК) ∧ ВРЕМЯ (s, t) ∧ БУДУЩЕЕ (t))
Повелительные	
10) Остановиться 11) Повернуться кругом 12) Двигаться на 10 футов	Осуществляется обращение к подпрограммам, реализующим указанные стандартные действия.
13) Идти к большой красной призме	$\exists s, x$ (У (Отправ., x, s) ∧ ЕСТЬ (x, большой) ∧ ЕСТЬ (x, красный) ∧ ЕСТЬ (x, призма))
14) Толкать черный ящик на вершину платформы	$\exists s, x, y, z$ (ТОЛКАТЬ (x, s) ∧ ЕСТЬ (x, черный) ∧ ЕСТЬ (x, ящик) ∧ НА (x, y) ∧ ЕСТЬ (y, вершина) ∧ ПРИНАДЛЕЖИТ (y, z) ∧ ЕСТЬ (z, платформа))
15) Собрать все кубы в центре комнаты	$\forall x \exists s, y, z$ (В (x, y, s) ∧ ЕСТЬ (x, куб) ∧ ЕСТЬ (y, центр) ∧ ПРИНАДЛЕЖИТ (y, z) ∧ ЕСТЬ (z, комната))
16) Исследовать комнату Джона	$\exists s, x$ (ИССЛЕДОВАТЬ (x, s) ∧ ЕСТЬ (x, комната) ∧ ПРИНАДЛЕЖИТ (x, Джон))

Для удобства читателей предложения переведены на русский язык. В таблице 8.8 приведены формулы исчисления предикатов, соответствующие предложениям естественного языка.

Повествовательные предложения (примеры 1-5 табл. 8.8) переводятся в аксиомы системы, вопросительные предложения (примеры 6—9 табл. 8.8) переводятся в утверждения, которые должны быть доказаны вопросно-ответной системой. Простые повелительные предложения (примеры 10-12 табл. 8.8) непосредственно переводятся в наименования стандартных программ, вызывающих выполнение отправителем определенных действий. Сложные повелительные предложения (примеры 13-16 табл. 8.8) рассматриваются системой как утверждения, возможность выполнения которых должна быть определена, т. е. они аналогичны вопросам.

Большинство предикатов, введенных в примерах, очевидны, однако, для ясности рассмотрим один пример (уже упоминавшийся ранее): «Высокий ящик толкался Джоном» (THE TALL BOX WAS PUSHED BY JOHN). Синтаксическая компонента, получив данное предложение, определяет, что к предложению необходимо применить трансформацию, переводящую его в активную форму «Джон толкал высокий ящик» (JOHN PUSHED THE TALL BOX). Затем базовая компонента распознает, что глагол «толкать» использован в предложении в прошедшем времени, и устанавливает в соответствующее значение предикат ВРЕМЯ. Прилагательное «высокий» и существительное «ящик» представляются в предикате ЕСТЬ. Полученная в результате формула может быть проинтерпретирована примерно так. Существуют состояние S, объект x и координаты y и z такие, что S есть состояние, полученное из начального ($S_{нач}$) применением к нему оператора (Джон толкает объект x из координат y и z), причем x относится к классу высоких объектов и к классу ящиков. Кроме того, состояние S имело место в прошлом.

Способ представления предложений в виде формул исчисления предикатов не является единственно возможным.

Сэндуолл указывает три возможных способа. Проиллюстрируем их на примере предложения «Джон продал лодку Мэри».

1) Глагол используется как предикатный символ. После трансляции входное предложение будет иметь вид ПРОДАЛ (Джон, Лодка, Мэри), где ПРОДАЛ является трехместным отношением, а Джон, Лодка, Мэри — константы. Если мы хотим указать дополнительные сведения о происходящем событии (например, время, место), то это может быть сделано введением предиката, содержащего большее количество аргументов.

2) Главным отличием этой нотации является использование небольшого множества базовых отношений и получение на их основе более сложных отношений благодаря введению понятия «события».

Рассматриваемое предложение после трансляции может иметь вид:

(∃ еПОДЛЕЖАЩЕЕ (Джон, е) ∧ СКАЗУЕМОЕ (продал, е) ∧ ДОПОЛНЕНИЕ (лодка, е) ∧ КОСВ.

ДОПОЛН. (Мэри, е) ∧ РЕАЛЬНОСТЬ (е)),

где «е» интерпретируется как событие, в результате которого «Джон продал лодку Мэри». Первые четыре отношения используют для того, чтобы описать данное событие, а предикат РЕАЛЬНОСТЬ используется для того, чтобы определить, что указанное событие фактически имеет место. В данной нотации можно представить и более сложные предложения, например: «Джон верит, что Петр продал лодку Мэри».

(∃ е ∃ р ПОДЛЕЖАЩЕЕ (Джон, е) ∧ СКАЗУЕМОЕ (верящий, е) ∧ ДОПОЛНЕНИЕ (р, е)

∧ РЕАЛЬНОСТЬ (е) ∧ ПОДЛЕЖАЩЕЕ (Петр, р) ∧ СКАЗУЕМОЕ (продавший, р) ∧ ДОПОЛНЕНИЕ (Лодка, р) ∧ КОСВ. ДОПОЛН. (Мэри, р)),

где «р» интерпретируется как событие: «Петр продал лодку Мэри».

Предикаты, используемые для описания события, существенно зависят от конкретной системы, так, если в системе используются глубинные падежи (см. п. 8.2.3), то вместо приведенных в примере предикатов ПОДЛЕЖАЩЕЕ, СКАЗУЕМОЕ, ДОПОЛНЕНИЕ и т. д. будут использоваться предикаты, отражающие глубинные отношения.

3) Третий способ отличается тем, что отношения между словами предложения выражаются не предикатами, а функциональными символами.

ЕСТЬ (Джон, Кому Юбъект (продавший, лодку), Мэри)).

Данное выражение можно представить в виде бинарного дерева (рис. 8.18), отражающего структуру предложения.

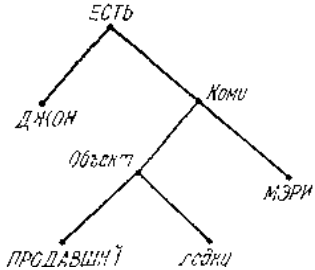


Рис 8.18. Представление структуры предложения в виде бинарного дерева.

В данном методе (так же как и в предыдущем) используются объекты (Джон, лодка, Мэри) и свойства (продавший). Отношение ЕСТЬ определено на двух аргументах: первый — объект, второй — свойства. Функции «Объект», «Кому» являются двухместными. Первый аргумент этих функций должен быть свойством, второй объектом, а результат является свойством. Подобным образом могут быть определены более сложные высказывания, содержащие кванторы, сравнительную степень прилагательных, придаточные предложения и т. п.

Первый метод широко используется. Причины его популярности вызваны тем, что для людей является естественной идентификация понятия «предикат» из традиционной грамматики с понятием «предикат» в исчислении предикатов. Однако этот метод имеет и очевидные недостатки. В нем затруднено выражение ссылок на предшествующий контекст, необходимо введение большого числа предикатов и функций при описании объектов реальной сложности.

Второй метод используется в нескольких вопросно-ответных системах. Этот метод близок к падежной грамматике Филмора (см. п. 8.2.3). Этот подход позволяет одному предложению ссылаться на другое и не требует введения большого числа предикатов.

Третий метод имеет некоторое сходство с глубинными структурами трансформационных грамматик (см. п. 8.2.3). Можно найти большое сходство между деревом на рис. 8.18 и традиционным С-маркером для предложения в естественном языке.

Второй и третий методы имеют преимущества перед первым и много общего между собой. Они вводят относительно немного функций и отношений, которые являются базовыми для отражения лингвистического и концептуального состава языка.

Итак, можно подытожить, что исчисление предикатов обладает достаточной выразительной мощностью для представления «глубинной структуры» предложений естественного языка.

Следует отметить различие в понимании термина «глубинные структуры» в лингвистике и при использовании в качестве глубинных структур формул исчисления предикатов. В лингвистике глубинная структура есть результат синтаксического анализа входного предложения до его объединения с базовыми данными, а в исчислении предикатов — после объединения.

Основным достоинством использования исчисления предикатов в качестве глубинных структур является существование для них универсальных вычислительных методов.

Для реализации конкретной системы с использованием исчисления предикатов необходимо решить две задачи:

1. Сформировать каноническое множество предикатов и функций, достаточных для описания среды, в которой будет функционировать коммуникационная система (отправитель, получатель). Множество должно быть достаточно малым, но полным.
2. Сформулировать аксиомы, выражающие свойства понятий, введенных в пункте 1.

В таблице 8.9 приведен экспериментальный список основных понятий, используемых для описания существенных свойств действий.

Таблица 8.9

Каноническое множество предикатов, используемых для описания действия

Предикаты	Вопросы
1. АГЕНТ (*x*, *y*)	Кто совершил действие?
2. ДЕЙСТВИЕ (*x*, *y*)	Что он сделал?
3. ОБЪЕКТ (*x*, *y*)	Для кого или чего он это сделал?
4. Описание окружения:	В каком контексте он это сделал?
ВРЕМЯ (*x*, *y*)	Когда он это сделал?
МЕСТО (*x*, *y*)	Где он это сделал?
ОБСТОЯТЕЛЬСТВО (*x*, *y*)	При каких обстоятельствах он это сделал?
5. Модальность:	Как он это сделал?
СРЕДСТВО (*x*, *y*)	С помощью какого инструмента или метода он это сделал?
СПОСОБ (*x*, *y*)	С помощью чего он это сделал?
6. Причинность:	Почему он это сделал?
ПРИЧИНА (*x*, *y*)	Какая причина побудила его сделать это?
ЦЕЛЬ (*x*, *y*)	С какими намерениями он это сделал?
СОСТОЯНИЕ (*x*, *y*)	В каком состоянии (настроении) он это сделал?

Для того, чтобы назначение конкретного предиката было более понятно, рядом с предикатом приведены вопросы в естественном языке. Не утверждается, что этот список в принципе полон, но для заданных целей он достаточен. Аксиомы должны описывать, по крайней мере, информацию трех видов.

1. Геометрические отношения окружающей среды.

2. Правила, описывающие ограничения на возможности отправителя по восприятию окружающей среды и манипуляций в ней.
3. Информацию, извлеченную из повествовательных предложений естественного языка, полученную в ходе диалога.

Следует отметить, что формулировка аксиом является довольно сложной задачей**. Можно предложить следующие неформальные шаги при выполнении аксиоматизации.**

1. Записать наблюдения об окружении, которое мы хотим описать. Аккуратно выразить все в корректном исчислении предикатов.

2. Проверить на непротиворечивость.

3. Проверить на избыточность, т. е. проверить, не выводимы ли некоторые аксиомы из других аксиом?.

Следует отметить, что наличие избыточности может ускорить вывод, поэтому добиваться отсутствия избыточности не всегда целесообразно.

4. Проверить на полноту, т. е. проверить, введены ли все аксиомы, необходимые для вывода. Общий метод осуществления проверки на полноту состоит в подборе проверяющих примеров (теорем, которые должны быть выводимы в предлагаемой аксиоматике) и проверке вручную, что каждая из них может быть доказана.

Проверяющие примеры могут быть «позитивные» и «негативные». Позитивные примеры — это те, которые используются при прямом выводе. Их задача — показать, что то, что должно быть выводимо, выводится. Негативные примеры должны показывать, что то, что не должно быть выводимо, выводится.

Следует отметить, что предложенные шаги имеют много общего с отладкой программы на вычислительных машинах. Однако аксиомы проще записывать, отлаживать и объединять в общую систему.

Остановимся теперь на вопросе объединения некоторого факта (повествовательное предложение), переведенного в формулу исчисления предикатов, с базовыми данными системы.

Объединить новый факт (*Ф*) с базовыми аксиомами *(А)* можно только в том случае, если этот факт не противоречит аксиомам.

Заметим, что если исходное предложение имеет после синтаксического и семантического этапа более одной интерпретации, то из них выбирается та, которая не противоречит базовым аксиомам (если такая интерпретация существует) .

Проверка на непротиворечивость состоит в определении невыполнимости множества $A \cup$ ~Ф, где A — исходное множество аксиом, ~*Ф* — отрицание объединяемого факта. Практически во всех системах дедуктивного вывода, основанных на исчислении

предикатов, для определения невыполнимости используются модификации принципа резолюции.

При определении невыполнимости возможны три исхода.

1. Множество формул $A \cup \sim\Phi$ невыполнимо (получен пустой дизъюнкт), что обозначает выводимость формулы Φ из базовых аксиом.

2. Множество формул $A \cup \sim\Phi$ выполнимо ($R_{i+1}=R(R_i(A))$), что означает невыводимость формулы Φ из аксиом.

3. Процедура резолюции не дает ответа за время, отведенное на доказательство (из-за зацикливания или из-за ограниченности времени).

В первом случае утверждение Φ, несмотря на его выводимость из A, иногда целесообразно добавить к A, так это может ускорить получение некоторых выводов.

Второй случай обозначает, что Φ невыводима в A. Известно, что *если замкнутая формула Ф теории А невыводима в А, то теория А', полученная из А добавлением* ~Ф *в качестве аксиомы, непротиворечива*.

На основании этого утверждения формула ~ Φ может быть добавлена к A.

Если произошел третий исход, то мы можем попытаться проверить выводимость ~Ф из A (т. е. установить невыполнимость $A \cup \Phi$). При этом опять же возможны три указанных исхода. Если имеет место первый случай, то это значит, что формула ~ Φ выводима из A и, следовательно, Φ противоречит A. В этом случае вопрос о раскрытии противоречия должен решать человек.

Второй случай указывает на то, что утверждение Φ должно быть добавлено к A.

Если имеет место третий случай, то это обозначает, что система не в состоянии установить (либо в принципе, либо в отведенное время), противоречит ли Φ исходному множеству A. Вероятно, наиболее разумное решение — выдать сообщение об этом факте и ждать указаний о присоединении (или неприсоединении) формулы Φ к исходному множеству аксиом.

8.6. Вывод ответа

8.6.1. Доказательство и извлечение ответа в системах с общим выводом

Входом для этапа дедуктивного вывода является некоторая формула *Ф* исчисления предикатов, полученная из вопросительного или повелительного предложения исходного языка и рассматриваемая как теорема, подлежащая доказательству. Задачей этапа является определение, следует ли данная теорема из базового множества аксиом *(А),* то есть определение невыполнимости множества $A \cup$ ~Ф. Как уже было указано в п. 8.5.3, при определении невыполнимости по методу резолюции возможны три исхода:

1) $A \cup$ ~*Ф* невыполнимо, следовательно, *Ф* выводима из *А;*
2) $A \cup$ ~*Ф* выполнимо, т. е. *Ф* не следует из *А*;
3) процедура не дает ответа.

Первый исход соответствует утвердительному ответу на поставленный вопрос. Второй исход соответствует ответу «для вывода недостаточно информации». Третий исход соответствует ответу «не знаю». В случае третьего исхода мы можем попытаться получить иной ответ на поставленный вопрос, либо увеличив время, отведенное системе на доказательство, либо выводя следование ~*Ф* из *А*, путем определения невыполнимости множества формул $A \cup$ *Ф*. В последнем случае опять возможны три исхода. Если установлена невыполнимость $A \cup$ *Ф*, т.е. ~*Ф* следует из *А*, на поставленный вопрос надо отвечать отрицательно. Выполнимость $A \cup$ *Ф*, как и прежде соответствует ответу «для вывода недостаточно информации». В связи с ограниченностью времени, отводимого на доказательство, и неразрешимостью исчисления предикатов, процедура доказательства может не дать ответа (третий исход). Часто одно из доказательств будет найдено, и ответ на вопрос получен.

Остановимся теперь на том, как извлечь из доказываемой теоремы информацию, являющуюся развернутым ответом на вопрос.

Для извлечения ответа требуется преобразовать граф опровержения с пустым дизъюнктом в корне в граф, называемый *модифицированным графом,* у которого в корне находится некоторое утверждение, могущее служить ответом. Преобразование состоит в том, что каждое предложение, возникающее из отрицания теоремы (часто называемой предположением), превращается в тавтологию (путем добавления к нему отрицания этого предложения). Затем в соответствии с первоначальной структурой графа опровержения выполняются те же

самые резолюции, что и раньше, до тех пор пока в корне не будет получено некоторое утверждение.

Так как при указанном преобразовании каждое предложение, возникающее из отрицания предположения, превращается в тавтологию, то модифицированный граф представляет собой доказательство того факта, что утверждение, расположенное в его корне, логически следует из аксиом и тавтологий. Поэтому оно также следует только из одних аксиом.

Приведем простейший пример, поясняющий принцип извлечения ответа.

П р и м е р 8.1. Пусть высказаны следующие утверждения: «Куб номер 2 находится всегда в том же месте, где куб номер 1» и «Куб номер 1 находится в месте *а*». И задан вопрос: «Где находится куб номер 2?».

Указанные предложения после перевода их в исчисление предикатов примут вид аксиом:

$\forall x$ (В(Куб1, x)→В(Куб2, x)) аксиома 1

В (Куб1, *а*) аксиома 2

и предположения: $\exists x$В(Куб2, x), которое должно быть выведено из имеющихся фактов.

В приведенном примере предикату *В* придана интерпретация «находится в определенном месте».

На рис. 8.19 приведен граф опровержения для нашего примера.

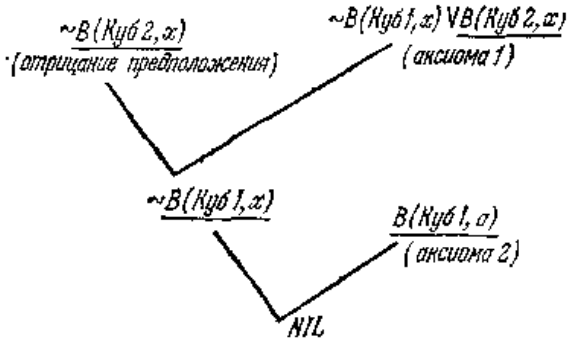


Рис. 8.19. Граф опровержения для примера 8.1.

Граф опровержения строится обычным путем.

Сначала отрицается формула, которую предстоит доказать. В рассматриваемом случае это приведет к формуле $\forall x$(~В (Куб2, x)). Затем это отрицание добавляется к множеству аксиом *(S)*, и все члены этого расширенного множества преобразуются в форму предложений.

Далее с помощью принципа резолюции показывается, что это множество неудовлетворимо.

Будем выделять литеры, подвергающиеся унификации при образовании каждой резольвенты в графе опровержения. На рис. 8.19 эти литеры подчеркнуты. Подмножество литер в предложении, подвергающееся унификации в процессе построения графа опровержения, назовем *множеством унификации*. Каждая резольвента в модифицированном графе опирается на множества унификации, которые в точности соответствуют множествам унификации исходного графа опровержения.

Для извлечения ответа из этого графа построен модифицированный граф доказательства (рис. 8.20).

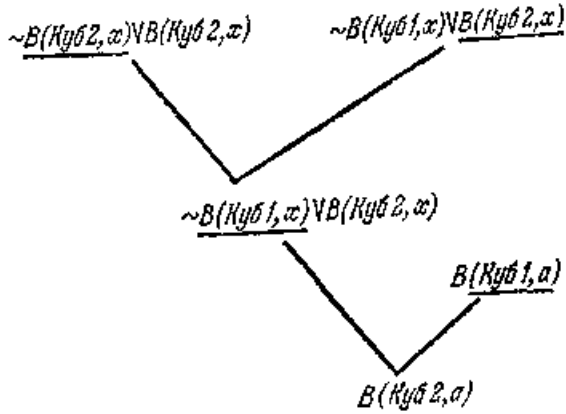


Рис. 8.20. Модифицированный граф доказательства для примера 8.1.

Смысл модификации графа состоит в том, чтобы извлечь тот частный случай переменной, относящейся к квантору существования, который служит ответом на вопрос. В нашем примере мы в качестве ответа получаем предложение B (Куб2, *a),* которое переводится в обычную форму исчисления предикатов (из формы предложений). Затем эта формула переводится в естественный язык, например, в виде предложения «Куб номер 2 находится в месте *a*».

Хотя указанный метод и прост, но у него есть несколько тонких моментов, которые мы разъясним на примерах.

Пример 8.2. Покажем, как преобразовать в тавтологию более сложные предложения, возникающие при отрицании предположения. Пусть в виде предложений дано следующее множество аксиом:

$$\sim A(x) \vee F(x) \vee G(f(x)),$$
$$\sim F(x) \vee B(x),$$
$$\sim F(x) \vee C(x),$$
$$\sim G(x) \vee B(x),$$
$$\sim G(x) \vee D(x),$$
$$A(g(x)) \vee F(h(x)).$$

Требуется доказать, исходя из этих аксиом, предположение $(\exists x \exists y((B(x) \wedge C(x)) \vee (D(y) \wedge B(y))))$.
Отрицание предположения переводит его в два дизъюнкта:

$$\sim B(x) \vee \sim C(x),$$
$$\sim B(x) \vee \sim D(x).$$

На рис. 8.21 приведен граф опровержения для примера 8.2.

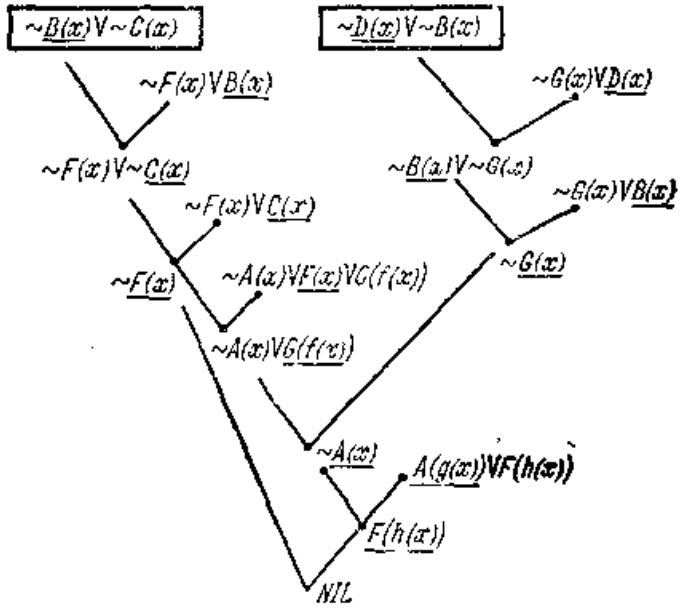


Рис. 8.21. Граф опровержения для примера 8.2.

Для преобразования графа следует превратить предложения, соответствующие отрицанию предположения, в тавтологии, добавив к ним их отрицания. Отрицания в данном случае (рис. 8.22) не являются предложениями (дизъюнктами), так как в них содержатся

конъюнкции. Однако мы будем обращаться с этими конъюнкциями как с единой литерой и действовать формально, как будто наша формула является дизъюнктом. Мы можем себе это позволить, так как ни один из элементов рассматриваемой конъюнкции не может оказаться ни в одном множестве унификации.

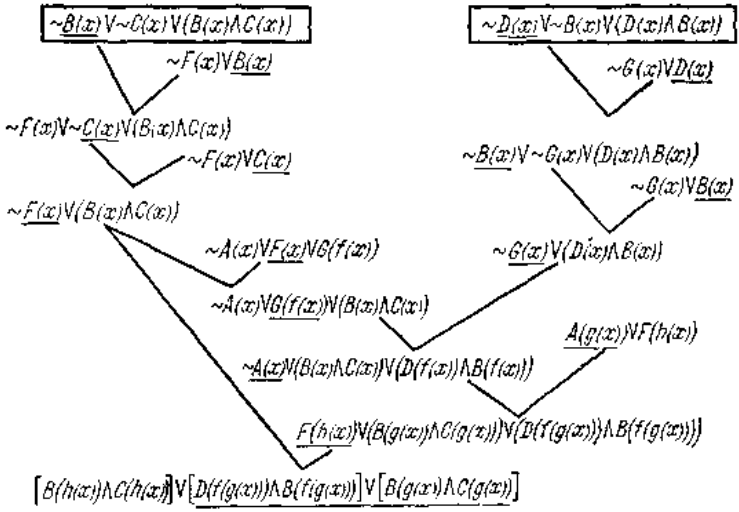


Рис. 8.22. Модифицированный граф доказательства для примера 8.2.

На рис. 8.22 приведен модифицированный граф. Мы видим, что ответное утверждение, расположенное в корневой вершине имеет форму, подобную форме предположения.

В общем случае, если предположение высказано в дизъюнктивной нормальной форме, то утверждение, получаемое в процессе извлечения ответа, представляет собой дизъюнкцию выражений, каждое из которых имеет форму либо всего предположения, либо одного или нескольких дизъюнктов этого предположения. Потому мы и говорим, что такое утверждение можно использовать в качестве «ответа» на вопрос, представляемый исходным предположением.

В случае, когда предположение, которое следует доказать, содержит переменные, относящиеся к квантору всеобщности, возникают дополнительные трудности. При отрицании такие переменные, переходят в переменные, относящиеся к квантору существования, а это приводит к необходимости введения сколемовых функций. Возникает вопрос: «как интерпретировать эти функции, если они появляются в качестве термов в ответном утверждении»?.

Если мы будем применять описанный выше метод модификации к предположениям, содержащим квантор всеобщности, то мы будем получать ответы частного вида.

Пример 8.3. Пусть задана аксиома $(\forall x \forall u(P(x, u, x) \vee P(a, u, u)))$ и требуется доказать предположение $(\exists w \forall v \exists y P(w, v, y))$.

На рис. 8.23, *а* представлен граф опровержения для примера 8.3. Граф доказательства, построенный по указанному ранее способу, приведен на рис. 8.23, *б*. Ответ, полученный в графе доказательства, имеет частный вид.

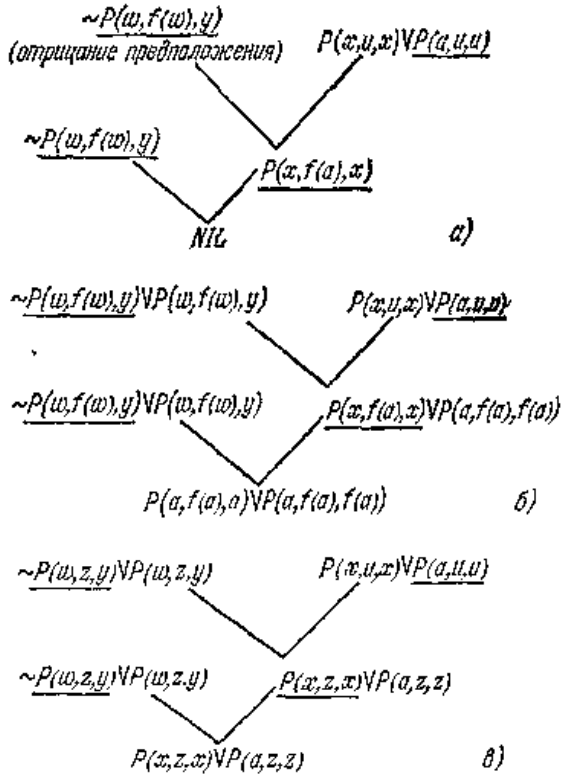


Рис. 8.23. Графы доказательств для примера 8.3.

Действительно, если из некоторого множества аксиом A мы можем доказать формулу $P(f(x))$, используя принцип резолюции, то это значит,

что $A \rightarrow (\forall x P(f(x)))$ является теоремой. Если f не встречается в A, то формула $A \rightarrow (\forall f \forall x P(f(x)))$ также доказуема и, следовательно, $A \rightarrow (\forall z P(z))$ есть теорема. Применив указанные действия к ответу на рис. 8. 23, *б* получим $(\forall z(P(a, z, a) \vee P(a, z, z)))$. Однако и этот ответ не является наиболее полным. Можно показать, что в процессе извлечения ответа всегда можно заменять сколемовые функции, возникающие при отрицании предположения, новыми переменными.

В модифицированном доказательстве в эти новые переменные не будет делаться никаких подстановок, так что они пройдут через доказательство без изменения и появятся в окончательном ответном утверждении.

На рис. 8.23, *в* приведен соответствующий граф доказательства. Ответ имеет вид $(\forall x \forall z(P(x, z, x) \vee P(a, z, z)))$.

В заключение перечислим основные этапы извлечения ответа.

1. Построение графа опровержения и выделение в нем множества унификации.
2. Подстановка новых переменных на место сколемовых функций, появляющихся в предложениях, образованных из отрицания предположения.
3. Преобразование предложений, полученных из отрицания предположения, в тавтологии.
4. Построение модифицированного графа доказательства, следуя структуре исходного графа опровержения. При образовании каждой резольвенты модифицированного графа используется множество унификации, определяемое множеством унификации графа опровержения.
5. Предложение, находящееся в корневой вершине модифицированного графа, представляет собой ответное утверждение, извлеченное в ходе описываемого процесса.

Очевидно, что ответное утверждение, построенное предложенным способом, зависит от опровержения, из которого оно было извлечено. Для одной и той же задачи может существовать несколько различных опровержений, и обычно у нас нет возможности установить, будет ли ответное утверждение, построенное на его основе, наиболее общим. Эта трудность представляет, вероятно, лишь теоретический интерес, так как ответы, получаемые работающими программами, являются вполне удовлетворительными.

Итак, мы рассмотрели способ получения ответа в системах с общим выводом, теоретически не накладывающих ограничений на тематику диалога. Однако в связи с тем, что существующие методы семантической интерпретации требуют заранее задать множество

предикатов и функций, определяющих тематику диалога, и в связи с ограниченностью времени и памяти, выделяемых на получение ответа, системы с общим выводом на практике не обладают универсальностью. Как следствие этого факта, системы с ограниченной логикой, не обладающие универсальным выводом, оказываются на практике конкурентоспособными по сравнению с системами с общим выводом. В следующем параграфе мы рассмотрим на примере системы с ограниченной логикой, введенной в п. 8.5.2, методы извлечения ответа из семантической сети.

8.6.2. Вывод ответа в системах с ограниченной логикой

Системы с ограниченной логикой не имеют универсальных процедур вывода, и их тематика задается при создании систем разработкой процедур, отвечающих на требуемые типы вопросов. Так, например, система, описанная в п. 8.5.2, отвечает на вопросы трех типов. В указанной системе базовые данные представлены в виде семантической сети, отражающей текущее состояние внутреннего мира системы, и в виде последовательности событий на временной оси, отражающей факты, сообщенные системе (рис. 8.15).

Для ответа в момент E_j на вопрос об отношениях, существовавших между узлами семантической сети в некоторый прошедший момент E_k, $k<j$, система должна вернуть сеть в состояние, соответствующее событию E_k. Возврат осуществляется путем движения по временной оси в направлении от события E_j к событию E_k. При этом, при прохождении через каждое событие E_i, $k \le i < j$—1, отношения, перечисленные в списке добавлений этого события, устраняются из семантической сети, а отношения, перечисленные в списке вычеркиваний этого события, добавляются к сети. После ответа на вопрос семантическая сеть возвращается в состояние, соответствующее событию E_j.

Приведем примеры типов вопросов, на которые умеет отвечать система. Примером первого типа вопросов является: КАКОЙ МУЖЧИНА КУПИЛ ПИРОЖНОЕ В БУЛОЧНОЙ? Ответ на этот вопрос приводит к исследованию временной оси. Для приведенного примера осуществляется поиск по временной оси в направлении прошлого до тех пор, пока не будет найдено событие, определяемое каноническим глаголом ОБМЕНЯТЬ. При этом параметр МЕСТО должен иметь значение *БУЛОЧНАЯ*.

Когда такой узел найден, проверяется, является ли значением параметра ПОЛУЧЕННОЕ элемент из множества L-ПИРОЖНОЕ. Если

условие выполняется, то проверяется, является ли значением параметра ПОКУПАТЕЛЬ элемент из множества L-МУЖЧИНА. Если и это условие выполняется, то узел, являющийся значением параметра ПОКУПАТЕЛЬ, принимается в качестве основы для ответа на вопрос. Ответ формируется либо на основании этого узла (это соответствует неполному ответу), либо на основании узла-события, являющегося родителем данного узла. Подробности формирования ответа приведены в п.8.7.

Примером вопроса второго типа является предложение: ЧТО БЫЛО СОБСТВЕННОСТЬЮ БУЛОЧНИКА ДО ТОГО, КАК ДЖОН КУПИЛ ЧТО-ТО В БУЛОЧНОЙ? В результате разбора предложение будет разделено на две части. Часть, соответствующая событию «ДЖОН КУПИЛ ЧТО-ТО В БУЛОЧНОЙ» обрабатывается аналогично вопросу первого типа, т. е. сеть возвращается в состояние, непосредственно предшествующее этому событию. Затем система ищет в полученной сети такое *x,* что для него справедливо отношение (СОБСТВЕННОСТЬ * БУЛОЧНИК * *x)*. Если такое *x* обнаруживается, то оно используется при формировании ответа на вопрос.

Примером вопросов третьего типа является предложение «ЧТО ЯВЛЯЛОСЬ СОБСТВЕННОСТЬЮ БУЛОЧНИКА ДО ТОГО, КАК ПИРОЖНОЕ СТАЛО СОБСТВЕННОСТЬЮ ДЖОНА?». Для того чтобы ответить на вопрос этого типа, сеть возвращается назад до тех пор, пока не будет получено состояние, в котором истинны отношения (СОБСТВЕННОСТЬ * ДЖОН* *x)* и (ЭЛЕМЕНТ *x* L-ПИРОЖНОЕ). Затем производится движение по временной оси в прошлое до тех пор, пока одно из этих отношений не перестанет быть истинным, а отношение (СОБСТВЕННОСТЬ х БУЛОЧНИК * *x)* не становится истинным для некоторого *x*. Если такое *x* найдено, то оно используется для ответа на вопрос. На этом мы закончим рассмотрение методов получения ответов на вопросы. Напомним, что ответы, полученные на данном этапе, выражены во внутреннем представлении системы, и поэтому нашей очередной и заключительной задачей является перевод их во внешнее представление.

8.7. Формирование ответа в ограниченном естественном языке

Задача данного этапа заключается в переводе смысла некоторого высказывания, выраженного во внутреннем языке системы, в предложение, составленное по правилам выходного языка.

Методы, используемые на данном этапе, подобны методам на этапе синтаксического анализа с той только разницей, что здесь мы на основании грамматики порождаем предложение, а не распознаем его. В частности, представление в виде расширенных сетей переходов, используемое при синтаксическом анализе, с успехом применяется в большинстве систем при формировании ответа.

Поясним детали, возникающие при формировании ответа, на примере системы, приведенной в п. 8.5.2.

В результате работы этапа дедуктивного вывода, описанного в предыдущем параграфе, происходит выбор узла в семантической сети, определяющего смысл формируемого высказывания. Управление формированием ответа осуществляется на основе словарной информации и грамматики выходного языка, представленной в РАСП. В словаре у каждого канонического глагола есть свойство, значением которого является список, содержащий соответствующие ему поверхностные глаголы. Кроме того, каждый поверхностный глагол имеет *Ф*-правила (см. табл. 8.6), соотносящие глубинные структуры синтаксическим образам поверхностного глагола.

Проиллюстрируем эти свойства на примере сети, приведенной на рис. 8.16. Пусть выбран узел, отмеченный на рисунке буквой Б. Событие основывается на каноническом глаголе ОБМЕНЯТЬ, которому в словаре соответствуют несколько поверхностных глаголов (купить, продать, заплатить, стоить). Выбор одного из этих глаголов может быть осуществлен любым желаемым образом (возможен случайный выбор). Предположим, что выбран глагол КУПИТЬ. С глаголом «купить» в словаре связано два $_k$-правила. Пусть выбрано первое правило

(BUYER ACTIVE THINGBT (FROM SELLER) (FOR THINGGIVEN))
(ПОКУПАТЕЛЬ АКТИВНЫЙ ПОЛУЧЕННОЕ (У ПРОДАВЦА) (ЗА ОТДАННОЕ)).

Это правило становится управляющим «предложением», которое должно быть разобрано генерирующей грамматикой.

Первый элемент в правиле указывает, что ему в соответствие должен быть поставлен узел, связанный с событием обменять (узел Б на рис. 8.16) глубинным отношением ПОКУПАТЕЛЬ. Этим узлом является *МЭРИ*. Итак, первым словом выходного предложения является МЭРИ. Второй элемент указывает на активный залог предложения. Пусть сформирован глагол КУПИЛ (BOUGHT). Третий элемент правила ПОЛУЧЕННОЕ указывает, что существительное должно быть сформировано на основании узла, удовлетворяющего отношению

ПОЛУЧЕННОЕ в состоянии Б. Этому узлу соответствует слово ЛОДКА (THE BOAT).

Очередным элементом является (У ПРОДАВЦА). Наличие скобок в элементе указывает, что включение в выходную строку соответствующего ему существительного возможно, но не обязательно. В нашем случае результатом включения будет У ДЖОНА (FROM JOHN). Последний элемент в правиле (FOR THINGGIVEN) будет связан с узлом *ДОЛЛАРЫ* и приведет к генерации в выходной строке конструкции ЗА 50 ДОЛЛАРОВ (FOR 50 DOLLARS). Итак Ф-правило разобрано полностью, и на выходе получено предложение: MARY BOUGHT THE BOAT (FROM JOHN) (FOR 50 DOLLARS) (МЭРИ КУПИЛА ЛОДКУ (У ДЖОНА) (ЗА 50 ДОЛЛАРОВ)). Напомним, что конструкции в скобках не являются обязательными в выходном предложении. Не вдаваясь в детали, отметим, чго, выбрав в качестве поверхностного глагола PAY (ПЛАТИТЬ), мы бы сформировали предложение MARY PAID JOHN 50 DOLLARS FOR THE BOAT (МЭРИ ЗАПЛАТИЛА ДЖОНУ 50 ДОЛЛАРОВ ЗА ЛОДКУ).

Выбрав глагол COST (СТОИТЬ), мы бы получили предложение THE BOAT COST MARY 50 DOLLARS (ЛОДКА СТОИЛА МЭРИ 50 ДОЛЛАРОВ). В разговорной речи большинство ответов на вопросы не являются полными предложениями. Грамматика, основанная на РАСП, позволяет начинать работу с любого узла в РАСП, что приводит к возможности генерировать не обязательно все предложение, а например, только группу существительного.

Рассмотрим пример на рис. 8.16. Простейшим ответом на вопрос «WHO SOLD THE BOAT?» (КТО ПРОДАЛ ЛОДКУ?) является ответ «ДЖОН».

Система работает таким образом, что если в качестве узла, на базе которого строится выходное предложение, выбран узел на временной оси, то ответ является полным предложением. Если выбран узел в сети, соответствующий существительному (например, *ВАГОН*, *ДЖОН* на рис. 8.14), то это существительное может быть использовано для получения неполного ответа. Узлу *ВАГОН* соответствует конкретный объект в реальном мире, известный как «вагон» (точнее множество из пятнадцати вагонов), с которым связаны значения: СТАРЫЙ, 15, КРАСНЫЙ, МАЛЕНЬКИЙ, являющиеся свойствами; ВОЗРАСТ, ЧИСЛО, ЦВЕТ, РАЗМЕР. Утверждение об узле *ВАГОН* во многом подобно узлу-событию. Но между этими узлами существует важное различие: **события происходят, совершаются, а утверждения являются статическими, неизменными до тех пор, пока не произойдет событие, изменяющее их истинность.**

Следует отметить, что случайная генерация модификаторов слова *ВАГОН* приводит к построению неправильных конструкций типа: КРАСНЫЙ, 15 ВАГОНОВ. Необходимо осуществить упорядочение модификаторов. Это может быть выполнено управляющей строкой словаря, руководящей генерацией группы существительного способом, аналогичным используемому при управлении генерацией предложения. Приемлемой строкой управления является (ЧИСЛО, РАЗМЕР, ВОЗРАСТ, ЦВЕТ).

В заключение отметим, что использование в качестве внутреннего представления исчисления предикатов не оказывает существенного влияния на изложенный метод формирования ответа. Справедливость этого замечания становится очевидной, если однозначно сопоставить каждой формуле исчисления предикатов некоторый граф.

8.8. Компьютеризация коммуникационной науки, ее проблемы и следствия

Развитие современной теории коммуникаций предполагает анализ и осмысление фундаментальных изменений, происходящих в науке, культуре и образовании в связи с широким внедрением компьютерных технологий и персональных компьютеров. Обращение к этой проблеме будет осуществлено лишь в той мере, в какой позволит рассмотреть новые возможности изучения знания и «знания о знании», а также выявить новые способы описания присутствия человека и социокультурной составляющей в разных формах «представления знания». Реализовать это возможно, опираясь на исследования в области когнитивной науки, где знание и информация являются главным предметом. Представление знания, как оно исследуется в когнитивной науке, не только предполагает предметное его содержание, но и определяет интерпретативную деятельность субъекта, социокультурную обусловленность его знания и поведения, а также фиксирует другие связи и отношения, представленные в традиционных эпистемологических структурах лишь опосредованно.

8.8.1. Эпистемология и когнитивная наука

Когнитивная наука (когнитология) сформировалась в 60-70-х годах XX века (Гарвард, США) в качестве дисциплины, исследующей методом компьютерного моделирования функционирование знаний в интеллектуальных системах. Когнитивную науку отличают

междисциплинарность, использование компьютерной метафоры и исследование познания. Центральным для всей проблематики когнитивной науки является обращение к компьютеру, служащему самой наглядной и самой убедительной моделью того, как формируется, структурируется и «работает» знание, а также имитируются различные когнитивные процессы (например, обучения или получения экспертного знания и т. п.). Феномен знания исследуется в аспектах его получения, хранения, переработки, коммуницированния (передачи), выясняется, какими типами знания и в какой форме обладает человек, как «представлено» знание в его голове и как он его использует.

Важную роль играет лингвистика, которая выступает для когнитивной науки как важный источник материала об устройстве когнитивных структур. По отношению к «искусственному интеллекту» (ИИ) когнитология является своего рода «теорией интеллектуальных машин и механизмов», т. е. сконструированных человеком компьютерных устройств и лишь через них — их естественных прообразов - людей познающих. Это объясняет различную природу эксперимента в психологии и когнитивной науке и определяет существование в последней компьютерной метафоры.

Традиционные проблемы гносеологии, эпистемологии, философии и методологии науки получили новое видение и интерпретацию. «Когнитивизм знаменовал появление новой парадигмы научного знания, и с ним в историю науки пришло новое понимание того, как следует изучать знание, как можно подойти к проблеме непосредственно не наблюдаемого — прежде всего к проблеме внутреннего представления мира в голове человека...» (Кубрякова Е.С., Демьянков В.З. и др. Краткий словарь когнитивных терминов. М., 1996. С. 61). Эксплицитно выраженные знания составляют лишь незначительную часть общей базы знаний человека. Согласно современным подходам, такая база есть самоорганизующаяся и саморегулируемая система. Она включает следующие компоненты:

языковые знания — грамматика (с фонетикой и фонологией), дополненная знанием композиционной и лексической семантики; знание об употреблении языка; знание принципов речевого обучения; внеязыковые знания — о контексте описываемой ситуации, об адресате коммуникации (в том числе знание о поставленных адресатом целях и планах, его представления о говорящем, об окружающей

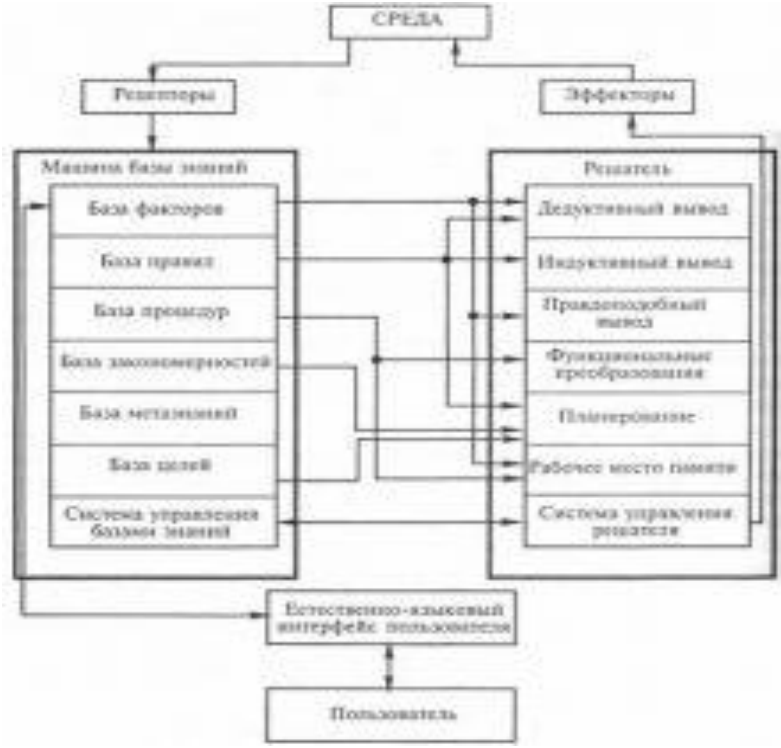
обстановке, знание своих умений); общефоновое знание, т. е. личностная картина мира (Осуга С. Обработка знаний. М., 1989. С. 9-10). При соотнесении эпистемологии и когнитивной науки необходимо различать знание и информацию, что упрощенно можно свести к формуле: информация — это знание минус человек; информация — знаковая оболочка знания. Под компьютерным представлением знания принято понимать информацию, хранимую в машине, формализованную в соответствии с определенными структурными правилами, которые компьютер может автономно использовать при решении проблем с помощью заложенных в нем алгоритмов типа логического вывода. Информационная модель знания (как записанная в компьютере, так и вербализованная в тексте) является лишь намеком на представленное знание, по которому человек способен творчески воссоздать само знание. Следует отметить принципиальное отличие той информации, которая служит для получения знаний человеком, от информации, изучаемой в теории информации. Когнитивное знание открывает человеку дополнительные возможности размышления и действия, увеличивает его свободу. Информация как управляющий сигнал уменьшает неопределенность допускаемых состояний управляемой системы. Знание — личное достояние знающих, перенимающих его друг у друга как образцы действия в процессах познания. Этого нельзя сказать об информации, которая в противоположность знанию не является достоянием конкретной личности, она равно доступна всем, хотя возможности превратить ее в знание у каждого свои, опирающиеся на личный опыт и способности.

Такое различие создается исключительно присутствием человека, способного извлечь из информации, записанной на бумаге или закодированной в компьютере, нечто, позволяющее реализовать человеческую свободу выбора. Пользователь получает представление о ряде возможных точек зрения, соответственно, возникает та самая неопределенность, которая является необходимой предпосылкой для выбора. Вечная философская тема — диалектика свободы и необходимости — специфически проявилась в исследованиях по представлению знаний. Для правильного понимания свободы важно выявить ее связь с объективной неопределенностью развития. Мысль о том, что свобода — это осознанная необходимость, была высказана Спинозой еще в XVII веке, принимавшим концепцию абсолютного и однозначного детерминизма. Свобода человека оказывается здесь иллюзорной: ни человек, ни общество в целом не имеют свободы выбора действий и поступков, а лишь осознают предопределенный ход

событий. Действительная свобода возможна лишь при наличии объективной неопределенности, когда принятое решение, понимание, осознание происходящего могут изменить ход событий, сами детерминируют социальный процесс. В этом случае нельзя спрятаться за некой безличной необходимостью, свобода объективно соотносится с этическими «параметрами» — нравственной ответственностью человека, осуществляющего выбор и принимающего решение. В когнитивных науках, таким образом, становится необходимой этическая рефлексия, основанная на принципе свободы личности.

Отличие традиционной гносеологии от разделов теории познания, имеющих дело с использованием компьютеров, состоит в том, что первая концентрируется на процедуре описания, обращаясь к высказываниям и правилам для получения знания.

«Компьютерная» теория познания делает центром своего внимания регуляцию, обращается к нормативным предложениям, использует знания для продуцирования правил. Сегодня развитие теории познания классическими гносеологическими средствами не всегда возможно, изменяются инструментарий гносеолога, требования к его профессиональной подготовке. Философия становится дисциплиной, сопричастной экспериментальной деятельности, осуществляемой при разработке программ искусственного интеллекта. Выяснилось, что именно в этой сфере возможна проверка самых тонких и абстрактных гипотез о природе человеческого разума. Сегодня здесь на первый план вышла проблема порождения знания, и это потребовало пересмотра базовых концепций ИИ. По ходу поиска обнаружилось, что идеи Локка Лейбница, Канта, Гуссерля, Хайдеггера — это концептуальные модели, которые могут быть «экспериментально» проверены в рамках программы ИИ, что позволяет по-новому решать философские споры о природе разума и познания. Так, по существу, экспериментально была доказана несостоятельность представления Локка о душе как «чистой доске» (tabula rasa), на которой только опыт записывает какое-либо содержание. Выяснилось, что возможности универсальных распознающих устройств, не имеющих тех эмпирических знаний-предпосылок, которыми обладает любой человек, ограничены. На самом деле «чистая доска», на которую записываются данные опыта, — те «образы», которые необходимо распознать иногда только по намеку, как это делает человек, — должна иметь весьма сложную структуру, включающую множество априорных знаний о мире. При разработке программ ИИ экспериментально



подтвердилась также огромная роль скрытых, неявных знаний, не выраженных в языке, но хранящих в себе жизненный опыт. **В литературе достаточно определенно высказывается мнение о том, что в настоящее время эксперименты на компьютерах, а не на людях — самый верный шаг на пути проверки гипотез о мышлении и познании, различных аналогов мыслительной деятельности.**

Такие эксперименты, разумеется, не могут рассматриваться в качестве полного доказательства предлагаемых гипотез, но принимаются как серьезный аргумент в их пользу. Программа искусственного интеллекта как своего рода «экспериментальная философия» делает фигуру эпистемолога столь же необходимой для компьютерной эпохи, как и математика-программиста. Эпистемология впервые за всю историю получает прямой выход в сферу конструктивной инженерной и технологической деятельности. Меняется характер связи эпистемологии с практикой.

Литература

1. Кононюк А.Е. Системология. Общая теория систем. К. 1 «Начала», - Киев: Освіта України, 2013.

2. Кононюк А.Е. Системология. Общая теория систем. К. 2, Ч.1 «Общая теория структур» , - Киев: Освіта України, 2014, 558 с.

3. Кононюк А.Е. Дискретно-непрерывная математика. К. 1 «Начала», - Киев: Освіта України, 2013, 576 с.

4. Кононюк А.Е. Дискретно-непрерывная математика. К. 2 «Множества (четкие)», - Киев: Освіта України, 2013, 522 с.

5. Кононюк А.Е. Дискретно-непрерывная математика. К. 3 «Множества (нечеткие)», - Киев: Освіта України, 2013, 452 с.

6. Кононюк А.Е. Дискретно-непрерывная математика. К. 4 «Отношения (четкие и нечеткие)», - Киев: Освіта України, 2013, 506 с.

7. Кононюк А.Е. Дискретно-непрерывная математика. К. 5, Ч.1 «Алгебры (четкие и нечеткие)», - Киев: Освіта України, 2011, 452 с.

8. Кононюк А.Е. Дискретно-непрерывная математика. К. 5, Ч.2 «Алгебры (четкие и нечеткие)», - Киев: Освіта України, 2011, 612 с.

9. Кононюк А.Е. Дискретно-непрерывная математика. К. 6, Ч.1 «Матрицы (четкие и нечеткие)», - Киев: Освіта України, 2011, 612 с.

10. Кононюк А.Е. Дискретно-непрерывная математика. К. 6, Ч.2 «Матрицы (четкие и нечеткие)», - Киев: Освіта України, 2011, 500 с.

11. Кононюк А.Е. Дискретно-непрерывная математика. К. 6, Ч.3 «Матрицы (четкие и нечеткие)», - Киев: Освіта України, 2012, 520 с.

12. Кононюк А.Е. Дискретно-непрерывная математика. К. 6, Ч.4 «Матрицы (четкие и нечеткие)», - Киев: Освіта України, 2012, 508 с.

13. Кононюк А.Е. Дискретно-непрерывная математика. К. 6, Ч.5 «Матрицы (четкие и нечеткие)», - Киев: Освіта України, 2012, 672 с.

14. Кононюк А.Е. Дискретно-непрерывная математика. К. 7, Ч.1 «Графы (четкие и нечеткие)», - Киев: Освіта України, 2014, 524 с.

15. Кононюк А.Е. Дискретно-непрерывная математика. К. 7, Ч.2 «Графы (четкие и нечеткие)», - Киев: Освіта України, 2014, 584 с.

16. Кононюк А.Е. Информациология. Общая теория информации. К. 1, - Киев: Освіта України, 2011, 476 с.

17. Кононюк А.Е. Информациология. Общая теория информации. К. 2, - Киев: Освіта України, 2011, 476 с.

18. Кононюк А.Е. Информациология. Общая теория информации. К. 3, - Киев: Освіта України, 2011, 412 с.

19. Кононюк А.Е. Информациология. Общая теория информации. К. 4, - Киев: Освіта України, 2011, 488 с.

20. Кононюк А.Е. Обобщенная теория моделирования. Начала. К.1.Ч.1, - Киев: "Освіта України", 2012. - 602 с.

21. Кононюк А.Е. Обобщенная теория моделирования. Начала. К.1.Ч.2, - Киев: "Освіта України", 2012. - 708 с.

22. Кононюк А.Е. Обобщенная теория моделирования. Начала. К.1.Ч.3, - Киев: "Освіта України", 2012. - 568 с.

23. Кононюк А.Е. Обобщенная теория моделирования. Числа (количественные оценки параметров моделей). К.2.- Киев:"Освіта України", 2012. - 548с.

24. Кононюк А.Е. Обобщенная теория моделирования. Величины и размерности. К.3.Ч.1, - Киев:"Освіта України", 2012. - 636 с.

25. Кононюк А.Е. Обобщенная теория моделирования. Величины (количественные характеристики моделей). К.3. Ч. 2. Физические величины (Начало), - Киев:"Освіта України", 2012. - 476 с.

26. Кононюк А.Е. Основы теории оптимизации. К.1. Киев:"Освіта України", 2011. - 692 с.

27. Кононюк А.Е. Основы теории оптимизации. Безусловная оптимизация. К.2. Ч.1. Киев:"Освіта України", 2011. - 552 с.

28. Кононюк А.Е. Основы теории оптимизации. Безусловная оптимизация. К.2. Ч.2. Киев:"Освіта України", 2011. - 616 с.

29. Кононюк А.Е. Основы теории оптимизации. Безусловная оптимизация. К.2. Ч.3. Киев:"Освіта України", 2011. - 456 с.

30. Кононюк А.Е. Основы теории оптимизации. Безусловная оптимизация. К.2. Ч.4. Киев:"Освіта України", 2011. - 512 с.

31. К. Берж. Теория графов и ее применение. М.,Иностр. Лит., 1962

32. Чемоданов Б.К., «Математические основы теории автоматического регулирования», 1977

33. Воронов А.А., «Введение в динамику сложных систем», 1980

34. Заде Л., Дезоер Ч., «Теория линейных систем», 1970