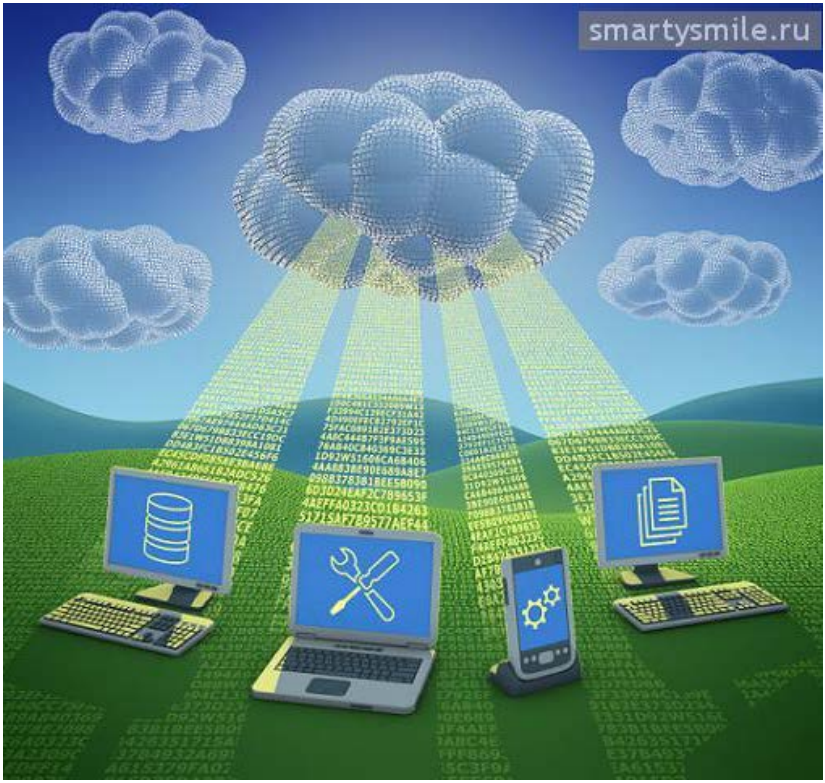
Облачные технологии

<u>Научная литература</u> – это связанные взаимным цитированием группы статей и монографий.

<u>Практическая литература</u> – литература справочного характера (т.е. сведения о предметах и способах их обработки)



Облачные технологии

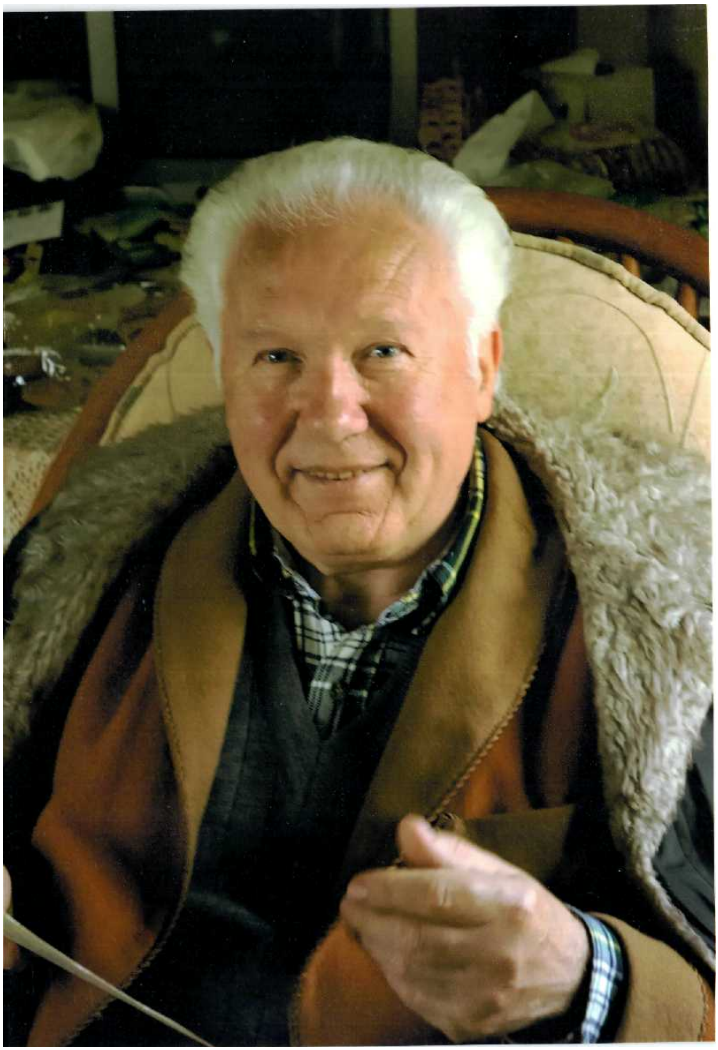
А. Е. Кононюк

Фундаментальная теория облачных технологий

Книга 2

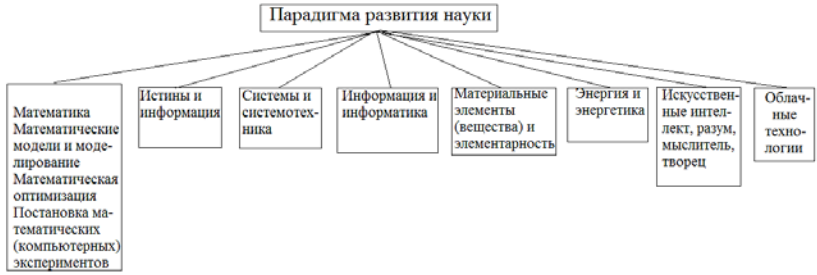
Введение в фундаментальную теорию облачных технологий

Киев
«Освіта України»
2018



Кононюк Анатолий Ефимович

Концептуальная схема
парадигмы развития науки



УДК 51 (075.8)
ББК В161.я7
К65

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

Кононюк А. Е.
К213 Фундаментальная теория облачных технологий. — В 18-и книгах. Кн.2. —К. : Освіта України. 2018.—528 с.

ISBN 978-966-373-693-8 (многотомное издание)
ISBN 978-966-373-694-15 (книга 2)

Многотомная работа посвящена систематическому изложению общих формализмов, математических моделей и алгоритмических методов, которые могут быть использованы при моделировании и исследованиях математических моделей объектов облачных технологий.

Развиваются представления и методы решения задач облачных технологий, основанные на теориях эвристического поиска и автоматизированного проектирования облачных управляющих систем, а также процедуральные методы, базирующиеся на классе проблемно-ориентированных языков, сочетающих свойства языков программирования и автоматических решателей задач отображения облачных технологий различными математическими средствами.

В работе излагаются основы теории облачных технологий такими математическими средствами как: множества, отношения, поверхности, пространства, алгебраические системы, матрицы, графы, математическая логика и др.

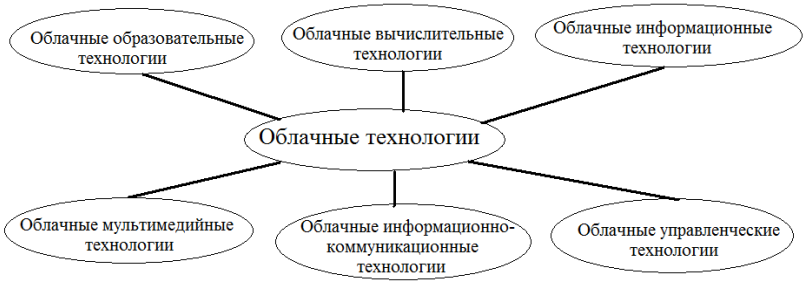
Для бакалавров, специалистов, магистров, аспирантов, докторантов всех специальностей.

УДК 51 (075.8)
ББК В161.я7
ISBN 978-966-373-693-8 (многотомное издание)
ISBN 978-966-373-694-15 (книга 2)
© Кононюк А. Е., 2018
© Освіта України, 2018

Глобальная сеть облачных технологий **(ГСОТ)**
(предметные области знаний)

1-13 - серверы ГСОТ предметных областей знаний

Представление классов облачных технологий
по научно-методическим направлениям



ОГЛАВЛЕНИЕ

1. Парадигма представления облачных технологий

1.1. Фундаментальна наука как базовая компонента парадигмы представления облачных технологий

Как принято считать (Википедия), **фундаментальная наука — область познания, подразумевающая теоретические и экспериментальные научные исследования основополагающих явлений и поиск закономерностей, руководящих ими и ответственных за форму, строение (организацию), состав, структуру и свойства, протекание процессов, обусловленных ими. Фундаментальная наука затрагивает (изучает) базовые принципы большинства гуманитарных и естественнонаучных дисциплин, — служит расширению теоретических, концептуальных представлений, в частности — детерминации идео- и формообразующей сущности предмета их изучения, — мироздания как такового во всех его проявлениях, в том числе и охватывающих сферы интеллектуальные, духовные и социальные (технологические).**

В некоторых научных кругах дают и такое определение фундаментальной науки: **Экспериментальная и/или теоретическая деятельность, направленная на получение новых знаний об основных закономерностях строения, функционирования и развития человека, общества, окружающей природной среды**.

Статус фундаментальных ЮНЕСКО присваивает исследованиям, которые способствуют открытию законов природы, пониманию взаимодействий между явлениями и объектами реальной действительности.

Существуют и другие определения понятия фундаментальной науки, но мы на них останавливаться не будем. Отметим только

неоднозначность толкования понятия фундаментальной науки как в приведенных определениях, так и в других определениях понятия фундаментальной науки. **Поэтому мы сосредоточим наше внимание на содержательном факторе фундаментальной науки.**

В **задачи фундаментальной науки не входит скорая и непременная практическая реализация (тем не менее, перспективно — эпистомологически целесообразные), в чём и состоит коренное отличие её от утилитарной теоретической или прикладной науки, являющихся таковыми и по отношению к ней**. Однако **результаты фундаментальных изысканий** находят и актуальное применение, **постоянно корректируют развитие любой дисциплины**, что вообще немыслимо без развития фундаментальных её разделов — **любые открытия и технологии непременно опираются на положения фундаментальной науки по определению**, а в случае противоречия с конвенциональными представлениями, не только стимулируют модификации таковых, но и нуждаются в фундаментальных исследованиях для полноценного понимания процессов и механизмов, лежащих в основе того или иного феномена, — дальнейшего совершенствования метода или принципа. Традиционно фундаментальные исследования соотносимы были с естествознанием, в то же время все формы научного познания опираются на системы обобщений, являющихся их основой; **таким образом и все гуманитарные науки обладают или стремятся обладать аппаратом, способным охватить и сформулировать общие фундаментальные принципы исследований и методы их истолкования. К основным** функциям **фундаментальных исследований относится — познавательная**.

Задачей **фундаментальных исследований** является получение конкретных представлений **о законах природы, которые обладают характерной общностью и стабильностью**.
К основным **признакам** фундаментальности относят:

а) концептуальную универсальность,

б) пространственно-временную общность.

Тем не менее, это не позволяет сделать вывод, что отличительной особенностью фундаментальности является отсутствие практической применимости, поскольку **в процессе решения фундаментальных**

проблем закономерно открываются новые возможности и методы решения практических задач, к которым относятся задачи, решаемые облачными технологиями.

Важное значение (роль) в теории фундаментальных наук играет **теория понятий и их образования**.

Различают точную **теорию понятий и аппромаксимационную теорию оценок понятий**. Каждая из них играет свою роль. Очевидно, что **понятия, связанные с точными числовыми характеристиками, относятся к точной теории**. Эта теория используется не только для построения и анализа абстракций, но она важна и для анализа объектов прикладных задач. ***Ведь числовые характеристики объектов всегда точны, а приближённы лишь их разнообразные оценки.***

В каждой области знаний наблюдается процесс, когда от первичного эмпирического субстрата (идею, замысел), через гипотезу, эксперимент и теоретическое его осмысление, при соответствующем их развитии и расширении, совершенствовании методологии, наука приходит к определённым постулатам, способствующим, например, поиску и формированию количественно выраженных положений, являющихся теоретической основой и для дальнейших теоретических же иссследований, и для формирования задач прикладной науки – формирования облачных технологий.

Совершенствование инструментальной базы, как теоретической, так и экспериментальной, — практической, служит (в корректных условиях реализации), совершенствованию метода. То есть любая фундаментадная дисциплина и любое прикладное направление, способны, в определённой степени, взаимно участвовать в развитии понимания и решения их самостоятельных, но и общих задач: **прикладная наука расширяет возможности исследовательского инструментария, как практического так и теоретического, фундаментальной науки, которая, в свою очередь, результатами своих исследований, предоставляет теоретический инструмент и основу для развития прикладной по соответствующей тематике**.

Часто встречаются ошибки толкования роли и значения фундаментальных наук.

Характерна ситуация, когда наблюдается непонимание самих терминов *фундаментальная наука* и *фундаментальные исследования*, — неправильное их употребление, и когда за *фундаментальностью* в контексте такого использования стоит *обстоятельность* какого-либо научного проекта. Такие исследования, в большинстве случаев, имеют отношение к *масштабным* изысканиям в пределах прикладных наук, к большим работам, подчинённым интересам тех или иных отраслей промышленности и т. п. Здесь за *фундаментальностью* стоит только атрибут *значительности*, притом никоим образом их нельзя отнести к *фундаментальным* — в том значении, о котором сказано выше. Именно такое неправильное понимание порождает деформацию представлений об истинном смысле действительно фундаментальной науки (в терминах современного науковедения), которая начинает расцениваться исключительно как «чистая наука» в самом превратном толковании, т. е. как наука оторванная от реальных практических потребностей, как обслуживающая, например, корпоративные проблемы яйцеголовых.

Достаточно быстрое развитие техники и системных методов (в отношении реализации полученного и давно «предсказанного» фундаментальной наукой) создаёт условия для иного рода неправильной классификации научных исследований, когда новое их направление, принадлежащее к области — междисциплинарных, расценивается как успех освоения технологической базы или наоборот, представляется только в виде линии развития — фундаментальных. В то время как последним эти научные исследования, действительно, обязаны своим происхождением, но имеют в большей степени отношение — к прикладным, и лишь косвенно служат развитию фундаментальной науки.

Примером тому могут служить нанотехнологии, основа которых сравнительно недавно, по срокам развития науки, была заложена, в числе многих других направлений фундаментальных исследований, — коллоидной химией, изучением дисперсных систем и поверхностных явлений. Однако это не значит, что лежащие в основе той или иной новой технологии фундаментальные исследования должны быть полностью подчинены ей, поглотив обеспечение других направлений; когда возникает опасность

перепрофилирования в отраслевые научно-исследовательских учреждений, призванных заниматься фундаментальными исследованиями достаточно широкого диапазона.

1.2. Особенности построения парадигмы представления облачных технологий

(Открытая развивающаяся панмедийная система наук (ОРПМСН) как парадигма представления облачных технологий)

В основу построения парадигмы представления облачных технологий положены **принципы системности и непрерывности**. В классическом истолковании принцип системности заключается в том, что наука представляет собой **открытую развивающуюся панмедийную систему наук**, а принцип непрерывности опирается на взаимозвязность (связность) отдельных научных дисциплин как целостную научную систему. Такая трактовка парадигмы представления облачных технологий порождает представление о полной предопределенности будущего ее развития.

1.2.1. Основные понятия объектно-ориентированного подхода построения парадигмы представления облачных технологий

Объектно-ориентированный подход основан на систематическом использовании моделей для языково-независимого функционирования систем облачных технологий, на основе их прагматики.

Последний термин нуждается в пояснении. *Прагматика* определяется целью функционирования системы облачных технологий. В формулировке цели участвуют предметы и понятия реального мира, имеющие отношение к функционированию системы облачных технологий (см. рисунок 1). При объектно-ориентированном подходе эти предметы и понятия заменяются их моделями, т.е. определенными формальными конструкциями, представляющими их в системе облачных технологий.

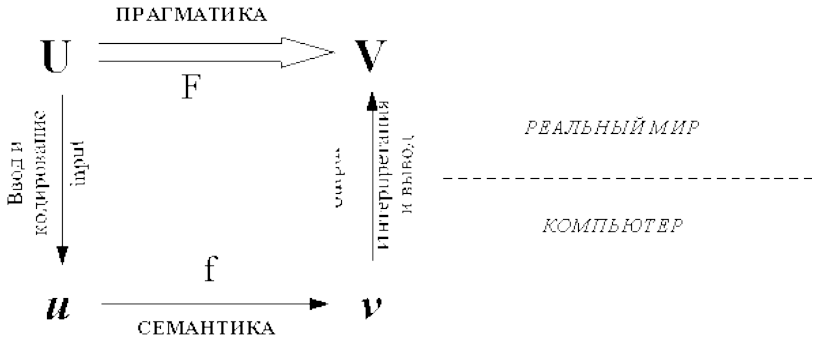


Рис. 1. Семантика (смысл процедуры облачных технологий с точки зрения выполняющего ее компьютера) и прагматика (смысл процедуры облачных технологий с точки зрения ее пользователей)

Модель содержит не все признаки и свойства представляемого ею предмета (понятия), а только те, которые существенны для исследования системы облачных технологий. Тем самым модель "беднее", а, следовательно, проще представляемого ею предмета (понятия). Но главное даже не в этом, а в том, что **модель есть формальная конструкция: формальный характер моделей позволяет определить формальные зависимости между ними и формальные операции над ними.** Это упрощает как разработку и изучение (анализ) моделей, так и их реализацию на компьютере. В частности, формальный характер моделей позволяет получить формальную модель системы облачных технологий как композицию формальных моделей ее компонент.

Таким образом, объектно-ориентированный подход помогает справиться с такими сложными проблемами, как

- уменьшение сложности задачи облачных технологий;
- повышение надежности процедур облачных технологий при выпонении задач облачных технологий;
- обеспечение возможности модификации отдельных компонент процедур облачных технологий без изменения остальных их компонент;
- обеспечение возможности повторного использования отдельных компонент процедур облачных технологий.

Систематическое применение объектно-ориентированного подхода позволяет использовать хорошо структурированные, организованные, надежные в эксплуатации, достаточно просто модифицируемые системы облачных технологий. Этим объясняется интерес исследователей и пользователей к объектно-ориентированному подходу и объектно-ориентированным языкам описания систем облачных технологий и выполняемых в них процессов. Объектно-ориентированный подход является одним из наиболее интенсивно развивающихся научных направлений.

Мы попытаемся показать целесообразность и плодотворность систематического применения объектно-ориентированного подхода на всех фазах жизненного цикла парадигмы представления облачных технологий, которая (парадигма) представлена как открытая развивающаяся панмедийная система наук (ОРПМСН).

Многие ученые научного сообщества оценивают нынешнее состояние фундаментальной науки как кризисное. Если проанализировать причины этого кризисного состояния фундаментальной науки, то, видимо, можно сделать вывод, что основная из них заключается в отсутствии общенаучных фундаментальных направлений развития науки. Как считает автор этой работы, такими фундаментальными направлениями развития науки могут быть:

- создание теории организации;
- создание теории состояния;
- создание теории устойчивости.

Такой взаимосвязанный триумвират развития науки позволит объединить идеологически разрозненные отдельные научные дисциплины в единную парадигму развития науки, которая может быть представлена в виде открытой развивающейся панмедийной системы наук.

1.2.2. Жизненный цикл функционирования парадигмы представления облачных технологий

В основе работы по созданию и использованию парадигмы представления облачных технологий лежит понятие жизненного

цикла, которое является одним из базовых понятий методологии построения парадигмы науки. Существует ряд общих методологий, которые могут быть использованы при разработке парадигмы представления облачных технологий. Главное в них - единая дисциплина функционирования во все периоды жизненного цикла парадигмы представления облачных технологий, учет проблемных задач облачных технологий и контроль их решения, применение развитых инструментальных средств поддержки процессов анализа, формирования и реализации парадигмы представления облачных технологий.

Парадигма представления облачных технологий функционирует в течении своего жизненного цикла. Совокупность фаз и периодов, которые проходит парадигма представления облачных технологий в своем развитии от момента принятия решения о необходимости возрождения посткризисного состояния облачных технологий до момента кризиса функционирования парадигмы представления облачных технологий, будем называть ***жизненным циклом парадигмы представления облачных технологий***.

В общем случае под термином **жизненный цикл парадигмы представления облачных технологий** будем понимать определенную эволюция, период времени и совокупность научных исследований, меняющих состояние парадигты представления облачных технологий от появления идеи (замысла) и начала ее разработки до окончания функционирования (кризиса облачных технологий). Жизненный цикл парадигмы представления облачных технологий разбьем на отдельные периоды. Периоды жизненного цикла парадигмы представления облачных технологий могут повторяться итерационным образом в связи с постепенным уточнением требований к парадигме и/или с необходимостью ее адаптации к тем изменениям, которые возникают в предметной области парадигмы.

Понятие ЖЦ парадигмы представления облачных технологий позволяет определить понятие жизненный цикл ***парадигмы представления облачных технологий*** - это модель создания и использования (эволюция) парадигмы, отражающая ее различные состояния, начиная с момента возникновения необходимости в данной парадигме и обмена информацией, и заканчивая моментом ее полного выхода из употребления у исследователей и пользователей.

ЖЦ парадигмы представления облачных технологий - это совокупность взаимосвязанных процессов создания и последовательного изменения состояния парадигмы представления облачных технологий, меняющих состояние парадигмы от формирования исходных требований к ней до окончания эксплуатации и утилизации комплекса средств автоматизации парадигмы представления облачных технологий.

Жизненный цикл парадигмы представления облачных технологий образуется в соответствии с принципом нисходящего формирования рекомендаций по развитию облачных технологий на каждом периоде ее функционирования и, как правило, носит итерационный характер: реализованные периоды, начиная с самых ранних, циклически повторяются в соответствии с изменениями требований и внешних условий, введением дополнительных ограничений и т.п., что приводит к изменениям в технологических решениях, выработанных на более ранних этапах.

Модель жизненного цикла парадигмы представления облачных технологий — структура, определяющая последовательность выполнения и взаимосвязи процессов, технологических процедур и технологических задач на протяжении всего жизненного цикла. Модель жизненного цикла зависит от специфики, масштаба и сложности технологических задач и специфики условий, в которых парадигма представления облачных технологий создается и функционирует.

Парадигма представления облачных технологий не предлагает конкретную модель жизненного цикла. Ее положения являются общими для любых моделей жизненного цикла, методов и процессов функционирования науки. В парадигме представления облачных технологий описывается структура процессов жизненного цикла, не конкретизируя, как реализовать или выполнить процедуры и задачи, включенные в эти процессы.

Модель жизненного цикла парадигмы представления облачных технологий включает в себя:

1. Фазы;
2. Результаты выполнения технологических работ в каждой фазе;

3. Ключевые технологические события — точки завершения технологических проблем и принятия решений.

Период — часть процесса функционирования парадигмы представления облачных технологий , ограниченная определенными временными рамками и заканчивающаяся формированием системы рекомендаций по обеспечению функционирования очередной фазы жизненного цикла, определяемого заданными для данной стадии технологическими требованиями.

В каждой фазе могут выполняться несколько технологических процессов, определенных в парадигме представления облачных технологий, и наоборот, один и тот же технологический процесс может выполняться в различных фазах. Соотношение между технологическими процессами и периодами также определяется используемой моделью жизненного цикла парадигмы представления облачных технологий.

1.2.3. Модель жизненного цикла парадигмы представления облачных технологий

Нами принята за основу **спиральная модель** (англ. *spiral model*) жизненного цикла парадигмы представления облачных технологий. При использовании этой модели парадигма представления облачных технологий создается в несколько итераций (витков спирали) методом прототипирования.

Спиральная модель характеризуется тем, что в начальных фазах ЖЦ осуществляются выработка стратегии, анализ требований и предварительное формирование технологических процедур. При этом создаются прототипы (модели), позволяющие проверить и обосновать реализуемость технологических решений. Каждый виток спирали соответствует пофазной модели создания фрагмента технологической версии парадигмы. На нём уточняются цели и характеристики базовых фундаментальных технологических направлений, определяется их качество, и планируются работы следующего витка спирали. В результате выбирается научно обоснованный вариант парадигмы представления облачных технологий или ее фрагмента, который и реализуется.

На рис. 2 представлены фазы жизненного цикла парадигмы представления облачных технологий: ***возрождение-становление-развитие-начало падения (предкризисное состояние)-совершенствование-угасание-кризис***.

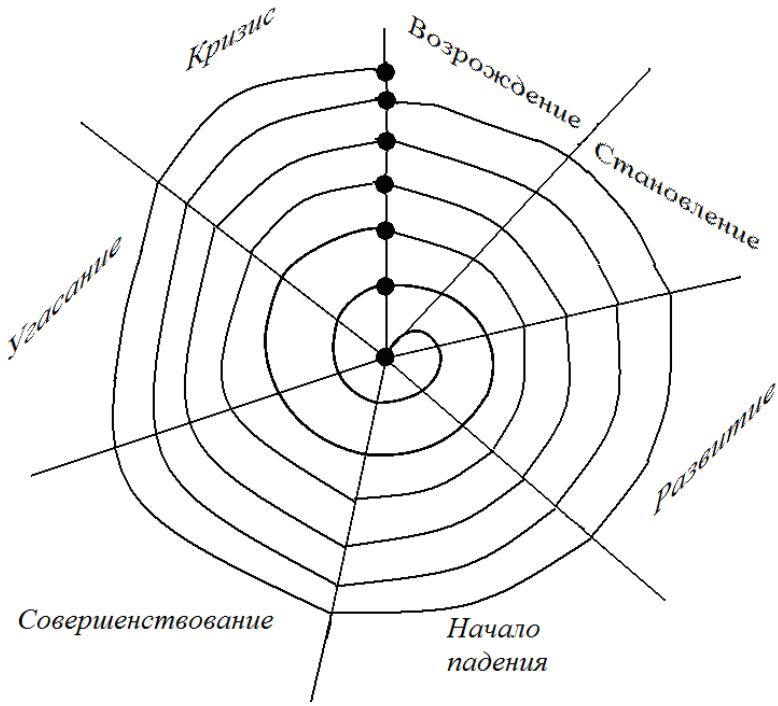


Рис. 2. Спиральная модель жизненного цикла парадигмы представления облачных технологий

Каждая итерация соответствует созданию фрагмента парадигмы представления облачных технологий, на ней уточняются цели и характеристики парадигмы, оценивается качество полученных технологических результатов и планируются работы следующей итерации.

На каждой итерации оцениваются:

- формирование технологических рекомендаций развития парадигмы;
- риск реализации сфомированных технологических рекомендаций;
- необходимость выполнения ещё одной итерации;
- степень полноты и точности понимания выработанных требований к парадигме;
- целесообразность прекращения проекта разработки парадигмы или ее фаз.

Отличительной особенностью спиральной модели является специальное внимание, уделяемое рискам, влияющим на организацию жизненного цикла, и контрольным точкам. Сформулируем наиболее распространённые (по приоритетам) риски:

1. Дефицит специалистов.
2. Нереалистичные сроки и бюджет.
3. Реализация несоответствующей функциональности.
4. Перфекционизм, ненужная оптимизация и оттачивание деталей.
5. Непрекращающийся поток изменений.
6. Нехватка информации о внешних компонентах, определяющих окружающую (технологическую) среду.
7. Недостатки в работах, выполняемых внешними (по отношению к проекту) ресурсами.
8. Недостаточная достоверность полученных технологических результатов.
9. Разрыв в квалификации специалистов разных технологических дисциплин.

В спиральной модели определён следующий общий набор контрольных точек:

1. Концепция (использования) создания парадигмы ;
2. Цели и содержание жизненного цикла парадигмы;
3. Архитектура жизненного цикла парадигмы; здесь же возможно говорить о готовности концептуальной архитектуры парадигмы;
4. Первая версия создаваемой парадигмы, пригодная для ее функционирования;

5. Проект парадигмы как готовый технологический продукт, развернутый для реального функционирования.
6. Реализация проекта парадигмы как готовый технологический продукт, развернутый для реального функционирования.

1.2.4. Основные этапы жизненного цикла парадигмы представления облачных технологий

Сгруппируем различные виды технологической деятельности, которые могут выполняться в течение жизненного цикла парадигмы представления облачных технологий, в периодические группы технологических процессов.
Каждый процесс включает ряд процедур (действий).
Каждая процедура включает ряд технологических задач.

Жизненный цикл парадигмы представления облачных технологий состоит из следующих этапов:

1. Этап описания идеи (замысла);
2. Этап построения (формирования) концепции;
3. Этап представления гипотезы;
4. Этап постановки эксперимента;
5. Этап формулировки (формирования) теории;
6. Этап разработки (выбора) методов;
7. Этап разработки процесса;
8. Этап разработки способа;
9. Этап разработки алгоритмов;
10. Этап проектирования технологий;
11. Этап эксплуатации технологий;
12. Этап начала падения эффективности технологий (предкризисное состояние);
13. Этап совершенствования;
14. Этап угасания;
15. Этап кризиса (утилизации).

Каждый этап имеет свой период существования в рамках жизненного цикла парадигмы представления облачных технологий.

1.2.4.1. Идея (замысел)

О том, как сформулировать идею своего продукта и что делать, когда его жизненный цикл подойдёт к концу.

99% новых проектов умирает. Почему умирают эти проекты? Потому что тот продукт, который они делают, оказывается никому не нужен, потому что за этот продукт никто не хочет платить. И в тот момент, когда умирает продукт — умирает сам проект. И всё потому, что ставка в проекте была сделана на один конкретный продукт, на всего лишь один способ конкретной реализации той идеи, которая на самом деле стала причиной появления этого продукта.

Отсюда можно и нужно сделать простой вывод — **работа над проектом должна начаться не с формулировки конкретного продукта, а с формулировки общей идеи, общей задачи, которую вы хотите решить.** После этого стоит переходить к продукту — и не к одному «беспроигрышному» варианту, а к перечню нескольких совершенно разных вариантов продуктов, которые могут быть реализованы в рамках этой общей идеи. Каждый конкретный продукт не взлетит с вероятностью 99%, наличие нескольких вариантов дает хотя бы надежду на то, что один из продуктов окажется востребованным, надежду на то, что вы сможете реализовать вашу общую идею, сделать успешным проект в целом.

Что же это за идея? Какими свойствами она должна обладать? Как ее сформулировать?

Идея — не может быть про то, *что* вы собираетесь сделать. «Моя идея состоит в том, что я хочу сделать мобильное приложение, которое...», «Моя идея состоит в том, что я хочу сделать сервис, который...» — это не идеи, это уже описания конкретных продуктов.

Идея — про то, *почему* вы хотите что-то сделать.

В этот момент классики обычно советуют людям формулировать «проблему, которую должен решать ваш продукт». Но «проблема» — это слишком драматичное определение. В наше время реальных

потребительских проблем не так уж и много — можно тем или иным способом практически все, что угодно, купить, практически любую услугу получить — просто с тем или иным количеством удобства или, наоборот, проблем.

Еще одна рекомендация от классиков на этом этапе сформулировать «потребность, которую вы хотите удовлетворить». Это тоже не всегда верно, потому что эти потребности людьми могут быть еще не осознаны, никто еще не бегает и не кричит: «У меня есть вот такая потребность — удовлетворите ее». Лежала ли на поверхности потребность в айфоне до того, как он появился?

Поэтому идеи больше похожи на жизненные наблюдения. Идеи могут отражать просто некий жизненный факт, например: «Многие люди любят носить вещи, которых нет у других», «Многие люди мало читают, потому что они просто не знают, что бы интересного им почитать». Или привычное и не драматичное внутреннее противоречие, например: «Большинство людей едят дома, но не любят мыть за собой посуду», «Многие люди хотели бы держать себя в хорошей физической форме, но у них не хватает терпения регулярно ходить в фитнес-клуб».

Или наличие препятствий или неудобств, например: «Людям, которые в течение дня колесят по городу, неудобно встречаться с курьерами из интернет-магазинов, которые должны приехать по указанному адресу в непонятный промежуток времени», «Есть такие товары, которые люди хотят пощупать, повертеть в руках, прежде чем их купить».

Мы специально берем простые жизненные примеры — жизненность идеи очень важна для успеха проекта. Огромное количество проектов умирает потому, что они удовлетворяют выдуманные их создателями потребности. Поэтому чем проще, жизненнее и очевиднее ваша идея, тем больше шансов на то, что созданный в рамках этой идеи продукт имеет шанс на жизнь.

При формулировке идеи очень важно подняться действительно на уровень жизненного обобщения. Многие в этот момент пытаются просто замаскировать продукт, переформулируя его в некоторое слабое подобие идеи. Например, «Люди хотят видеть все развлекательные мероприятия города на одном сайте». Почему они этого хотят? Они действительно хотят иметь сайт со списком всех

мероприятий? Или речь идет о том, что многие люди хотели бы пойти вечером развлечься, но просто зачастую не знают, куда пойти — как узнать варианты, куда им можно сходить?

Если вы до сих пор находитесь на таком уровне формулировки своей идеи, то попробуйте несколько раз подряд задать себе вопрос «Почему?». Почему вы думаете, что люди этого на самом деле хотят? Сформулировали ответ, задали себе этот же вопрос еще раз. **И так по цепочке, пока вы не подниметесь на уровень обобщения**. Этот уровень как раз и характеризуется жизненностью и, скорее всего, очевидностью. Если вы потом посмотрите на получившуюся идею и начальный продукт, от которого вы оттолкнулись в поисках идеи, то абсолютно не исключено, что вы поймете, что для реализации этой идеи можно сделать гораздо более простой, интересный и востребованный продукт, чем тот, с которым вы бегали до сих пор.

Другое главное свойство идеи, вытекающее из ее общности — это наличие совершенно разных вариантов ее реализации, возможность сделать совершенно разные продукты, которые могли бы жить в рамках одной простой идеи. Давайте возьмем простую идею: «Многие люди едят дома, но не любят мыть за собой посуду». В рамках этой идеи можно сделать автоматическую посудомойку, сервис обмена грязной посуды на чистую — выставляете вечером за дверь мешок с грязной посудой, утром получаете ее чистой, тарелки из хлеба, которые вы съедаете вместе с едой, одноразовые тарелки премиум-класса, из которых не противно есть, в отличие от существующих одноразовых тарелок.

Итак, резюмируем первые признаки хорошей идеи.

Жизненность. Жизненность позволяет надеяться на то, что продукт, который вы будете делать в рамках этой идеи, решает не выдуманную вами задачу, а помогает решать то, с чем сталкивается большинство людей в реальной жизни.

Низкая степень сомнительности в некотором диапазоне от «очевидно» до «вполне возможно». Чем от более очевидного соображения вы будете отталкиваться, тем выше вероятность того, что получившийся в итоге продукт будет действительно нужен хоть кому-то. Если ваша исходная точка уже сомнительна сама по себе, есть огромные сомнения в жизнеспособности получившегося продукта.

Достаточное количество различных способов ее реализации. Как правило, 9 из 10 продуктов умирает. Грубо говоря, опираясь просто на арифметику, можно сказать, что у вас должно быть десять вариантов продуктов, чтобы один из них выжил.
Уровень обобщения поможет вам попробовать несколько вариантов продуктов, оставаясь в рамках общей идеи, одного проекта, а не метаться от одного неудавшегося проекта к другому.

Это были основные признаки, позволяющие «опознать» идею. Кроме признаков полезно обратить внимание еще на несколько вещей, связанных с идеями.

Наличие идеи, позволяющей варьировать продукты в рамках одной идеи, помогает еще на одном этапе жизни проекта. У каждого продукта есть свой жизненный цикл — через какое-то время потребность в данном конкретном продукте исчезает, так как изменяются технологии и связанные с ними инструменты удовлетворения желаний, способы решения человеческих задач. В этот момент компании очень важно не умереть вместе со своим единственным и умирающим продуктом, а иметь возможность запустить новый продукт, приходящий на смену старому продукту, но оставаясь в рамках той же общей идеи, той же компании, решающей те же задачи, но с помощью новых инструментов.

Стоит обратить внимание еще на одну вещь, которую стоить иметь в виду, когда вы ищете свою идею. Идеи, как правило, базируются на жизненных наблюдениях. Стоит особенно наблюдать за теми явлениями, которые только входят в нашу жизнь. Не за теми, которые уже очевидны для всех, и которые, как следствие, реализует еще множество людей. А за теми, которые только становятся заметными, потому что нет ничего хуже острой конкуренции. Чем меньше конкурентов, тем лучше для вас — особенно на старте.

Наличие идеи помогает вам еще и при выделении ваших потенциальных конкурентов. Очень важно видеть всех конкурентов, конкуренты — это не только те, кто делает точно такой же продукт, как и вы. **Конкуренты — это те, кто работает в рамках той же идеи, и, как следствие, решают те же задачи, удовлетворяют те же потребности, устраняют те же самые противоречия, помогают в преодолении тех же самых неудобств — пусть даже и с помощью других продуктов.** Они все равно являются вашими конкурентами —

потому что при решении своих задач покупатель может выбрать любое подходящее решение. Например, BlaBlaCar, сервис поиска автомобильных попутчиков для поездок между городами, во Франции был вполне официально назван конкурентом местного железнодорожного сообщения.

В общем, не надо сразу бросаться делать продукт. Надо задать себе достаточно вопросов «Почему?», чтобы нащупать свою идею. Простую, жизненную и вполне себе похожую на правду — это позволит не кидаться из стороны в сторону, а планомерно перебирать разные варианты в поисках успешного. Допускаем, что в момент задавания вопросов «Почему?» окажется, что вот как раз «почему» — вдруг станет непонятно. Причина подумать о том, что, может быть, этим и не стоит заниматься вообще?

Продукт — это способ конкретной реализации нашей идеи. Мы можем пробовать разные продукты на пути реализации идеи, пока не найдем тот успешный вариант, который будет нас удовлетворять.

А что значит «удовлетворять»? Удовлетворять — значит, мы сможем с его помощью зарабатывать деньги. Продукты, сервисы, услуги, товары — это не только путь реализации абстрактной идеи, но и конкретный способ зарабатывать деньги. На этом уровне мы должны уже более внимательно думать о том, что конкретно, кому конкретно и как конкретно мы продаем. А самое главное — что люди при этом покупают?

Между фразами «что мы продаем» и «что люди покупают» нет знака равенства. Иногда возможны ситуации, когда мы вообще продаем одно, а люди покупают другое.

Замысел

Замысел - исходное представление разработчика о своем будущем проекте, его более или менее осознанный прообраз, с которого начинается творческий процесс. Планы, программы, заявки, наброски— вот наиболее распространенные формы внешней фиксации замысла. Но порою он может существовать и без такой фиксации, лишь как факт самосознания автора. Непосредственным импульсом к возникновению замысла могут служить, в зависимости от индивидуальных особенностей разработчика, самые различные

факторы: эмоциональное потрясение, случайная встреча с поразившим его воображение человеком, прочитанная книга и т. п. Иногда замысла рождается без ощутимого внешнего толчка, как бы внезапно, спонтанно. Но всегда он в конечном счете обусловлен общественной практикой, мировоззрением разработчика, подготовлен всем течением его жизни. Подобно гипотезе в науке, замысел в технологии в ходе работы над его воплощением конкретизируется, уточняется, а порою и принципиально меняется, что, однако, не умаляет его огромной эвристической значимости. Лишь с возникновением замысла в сознании автора складывается прогнозирующая установка на творческий поиск, который становится целенаправленным и внутренне организованным. Разработчик приобретает возможность прояснить для себя основные направления этого поиска, представить его результаты, проверить в процессе практической работы правильность своих планов и намерений, чтобы найти единственно возможное для себя идейно-образное решение темы.

1.2.4.2. Формирование концепции (Концепция проекта)

Итеративный подход к процессу разработки облачных технологий (характерный для MSF) требует использования гибкого способа ведения документации. *Живые документы* (*living documents*) должны изменяться по мере эволюции проекта. Такой подход существенно отличается от принципов ведения документации в известной каскадной модели, где процесс разработки начинается лишь после того, как готовы и зафиксированы все требования и спецификации.
Документация проектов MSF, также как и программный код, разрабатывается итеративно. На *фазе выработки концепции* планы имеют форму описания высокоуровневых *подходов* (*approaches*) и по мере подготовки распространяются среди членов проектной группы и других заинтересованных лиц для получения отзывов. К примеру, подход к тестированию может быть кратко сформулирован во время фазы выработки концепции, а его превращение в план тестирования происходит на более поздних фазах. После перехода к *фазе планирования* документы постепенно дорабатываются, возникающие детальные планы снова поступают на проверку всем заинтересованным сторонам, и описанный процесс повторяется итеративно. Типы планов и общее количество описывающих их документов могут варьироваться от проекта к проекту.

Необходимость проекта

Обоснование необходимости

Сформулируйте, на разрешение каких проблем и/или удовлетворение каких потребностей заинтересованных сторон направлен проект.

Видение проекта

Видение (*vision*) – это ничем не ограничиваемое представление о том, каким должно быть *решение* (*solution*). Видение проекта направлено на формирование у всех вовлеченных в проект сторон единого понимания его концепции. *Формулировка видения* (*vision statement*) должна быть достаточно краткой для запоминания, достаточно ясной для понимания и достаточно сильной для мотивирования. Хорошая формулировка видения ориентируется на пять SMART характеристик:

- *Specific* (определенность/конкретность) – видение четко указывает на то (идеальное) состояние, достижение которого является целью проекта.
- *Measurable* (измеримость) – дает проектной группе четкий критерий успешности проекта и достижения поставленных целей.
- *Achievable* (достижимость) – цели, сформулированные в видении, должны быть достижимы в рамках имеющихся ресурсов, времени и возможностей команды. Достижимость мотивирует команду на выполнение проекта.
- *Relevant* (обоснованность) – цели, сформулированные в видении, должны иметь существенное значение для заинтересованных сторон и напрямую быть связанными с их проблемами и/или потребностями.
- *Time-based* (ограниченность во времени) – видение должно четко указывать на ожидаемые временные рамки, в которые решение будет достигнуто.

Сформулируйте (максимально кратко, в одной, двух фразах) видение проекта.

Анализ выгод

Основываясь на сформулированном выше видении проекта, перечислите, какие выгоды получат заинтересованные стороны по завершении проекта (в результате внедрения и использования решения).

Концепция решения

Концепция решения (*solution concept*) предоставляет общее описание подходов, которые проектная группа предполагает использовать для разрешения проблем и/или удовлетворения потребностей заинтересованных сторон.

Цели и Задачи

Формирование концепции решения начинается с выяснения у заинтересованных сторон, описания и фиксации проектной группой **целей проекта**. Далее каждая **цель** разбивается на измеримые компоненты – **задачи.**

Во взаимодействии с заинтересованными сторонами проекта сформулируйте и утвердите **цели решения**, на достижение которых направлен проект. Определите **задачи**, из которых будет складываться достижение каждой цели.

Предположения и Ограничения

В процессе формирования концепции решения проектная группа постоянно взаимодействует с заинтересованными сторонами, собирая необходимую информацию о требованиях к функциональности будущего решения. Тем не менее, неизбежная неполнота информации приводит к тому, что относительно некоторых функциональных возможностей решения могут потребоваться *предположения* (*assumptions*). Помимо функциональных требований заинтересованные стороны могут выдвигать качественные требования, задающие *ограничения* на создаваемое решение. Также ограничения могут порождаться средой, в которой должно будет функционировать решение после внедрения.
Определите, имеются ли в проекте требования, нуждающиеся в предположениях, если да, сформулируйте их. Определите, имеются ли ограничения на будущее решение. Если да, сформулируйте их.

Анализ использования

Основой формулировки требований является анализ использования, включающий определение *пользователей* (*users*) и описание того, как пользователи будут взаимодействовать с решением.

Пользователи

В разработке решения заинтересованы множество сторон, однако непосредственная работа с ним будет выполняться пользователями, поэтому прежде чем приступать к дизайну решения, необходимо определить, кто будет с ним взаимодействовать. В процессе анализа должны быть выделены группы пользователей (например, на основе областей их деятельности, в которых будет использоваться разрабатываемое решение).
Сформируйте список групп пользователей, для которых предназначено решение.

Сценарии использования

Сценарии использования (*usage scenarios*) определяют последовательности действий, которые пользователи выполняют при

взаимодействии с решением. MSF не специфицирует явным образом способы описания сценариев использования. Один из возможных (и достаточно распространенных) вариантов – язык UML.
Для каждой выделенной на предыдущем шаге группы пользователей определите характерные способы их взаимодействия с решением и, используя необходимые диаграммы UML, опишите сценарии использования.

Требования

Требования (*requirements*) определяют, что должно делать разрабатываемое решение. **Требования могут выражаться в терминах функциональности или в виде правил и параметров, определяющих функциональность.**

Требования пользователей

Сформулируйте требования к решению с точки зрения пользователей.

Системные требования

Сформулируйте требования к решению с точки зрения среды, в которой оно должно будет функционировать после внедрения.

Рамки

Рамки (*scope*) определяют пространство параметров, в котором будет создаваться решение, детализируя функциональность, определяя, что останется за рамками решения и указывая критерии, по которым заинтересованные лица будут судить о готовности решения. Рамки создаются на основе единого видения, являются результатом компромисса между сформулированными целями и условиями реальности и отражают приоритезацию заказчиком имеющихся требований к создаваемому решению. Частью процесса определения рамок проекта является вынесение менее важной функциональности из текущего проекта в планы на будущее.
Рамки решения (solution scope) определяют функциональность решения и его возможности (включая те, что не относятся к программному обеспечению).
Возможность (*функциональность, составляющая, feature*) – это требуемый или желаемый аспект программного или аппаратного обеспечения. Например, предварительный просмотр перед печатью может быть возможностью текстового процессора; шифрование

почтовых сообщений – возможностью почтовой программы. Сопроводительные руководства пользователей, интерактивные файлы помощи, операционные руководства и обучение также могут быть составляющими решения.
Рамки проекта (project scope) определяют объем работ, который должен быть выполнен проектной группой для поставки заказчику каждого из элементов, определенного рамками решения.
Управление рамками проекта критично для его успеха. MSF предлагает определять и фиксировать рамки решения и проекта, используя *треугольник компромиссов* и *матрицу компромиссов проекта*.

Функциональность решения

Укажите здесь функциональность в терминах возможностей (features) и функций (functions), которая будет реализована в разрабатываемом решении.

За рамками решения

Укажите здесь функциональность, которая имеется или предполагается в требованиях заинтересованных сторон, но не будет реализована в решении, и опишите причины вынесения данных возможностей и функций за рамки решения (используйте треугольник компромиссов).

Критерии одобрения решения

Сформулируйте здесь критерии, в соответствии с которыми заинтересованные стороны будут принимать готовность решения.

Стратегии дизайна решения

Стратегия архитектурного дизайна

На основе разработанного списка возможностей и функций формируется *стратегия архитектурного дизайна* (*architectural design strategy*), описывающая решение в целом. Она определяет компоненты решения и их взаимодействие. Отличный способ описания решения на этом этапе – **использование иллюстрирующих диаграмм (например, UML).**
Сформируйте и опишите общий архитектурный проект решения.

Стратегия технологического дизайна

Разработка решения потребует использования определенных продуктов и технологий. *Стратегия технологического дизайна* (*technical design strategy*) описывает, какие технологии и продукты выбраны проектной группой в качестве средства реализации решения.
Аргументировано опишите, какие технологические средства будут использованы в процессе работы над решением.

1.2.4.3. Формулировка гипотезы

Гипо́теза (др.-греч. ὑπόθεσις «предположение» от ὑπό «снизу, под» + θέσις «тезис») — предположение или догадка; утверждение, предполагающее доказательство, в отличие от аксиом, постулатов, не требующих доказательств. Гипотеза считается научной, если она удовлетворяет научному методу, то есть объясняет все факты, которые гипотеза призвана объяснить; не является логически противоречивой; принципиально проверяема, то есть потенциально может быть проверена критическим экспериментом; не противоречит ранее установленным фактам; приложимо к возможно более широкому кругу явлений.

Также она может определяться как форма развитий знаний, облачных технологий, представляющая собою обоснованное предположение, выдвигаемое с целью выяснения свойств и причин исследуемых облачных технологий.

Как правило, гипотеза высказывается на основе ряда подтверждающих её наблюдений (примеров), и поэтому выглядит правдоподобно. Гипотезу впоследствии или *доказывают*, превращая её в установленный факт, или же *опровергают* (например, указывая контрпример), переводя в разряд ложных утверждений.

Недоказанная и неопровергнутая гипотеза называется **открытой проблемой**.

Это умозаключение, вывод о высокой вероятности чего-либо, построенный на основаниях (в виде ряда имеющихся наблюдений и перечня известных закономерностей).

Гипотеза в философии и других науках

Карл Поппер в философии науки дополнил позитивистский принцип верифицируемости принципом фальсифицируемости. Естественнонаучная теория не может быть окончательно подтверждена опытом. Опыт может её только опровергнуть. Любое научное знание носит лишь относительный, гипотетический характер. **Рост научного знания осуществляется благодаря выдвижению и опровержению (фальсификации) гипотез**. Научными могут быть только проверяемые (потенциально опровергаемые) утверждения.
Ученик Поппера Лакатос развил концепцию учителя. Отдельную (естественнонаучную) теорию, которая неизбежно опровергается, нельзя рассматривать как научную. **Научной может быть только «исследовательская программа» — последовательность опровергаемых и сменяющих друг друга теорий-гипотез.** Геоцентрическая механика Птолемея, гелиоцентрическая механика Галилея и Кеплера, классическая механика Ньютона и Галилея, релятивистская механика, квантовая механика, квантовая теория поля,…

Отличие Гипотезы от Теории

Часто можно встретить ситуации, когда люди случайно, по не знанию или намеренно путаются в терминах «теория» или «гипотеза». Так можно часто услышать фразу: «Это всего лишь теория...», которую относят к таким явлениям как «Глобальное потепление», «Эволюция» и другие. На самом деле существуют довольно точные критерии, которым могут отнести утверждение к одной или другой категории. Так в представленной ниже таблице показано отношение Ньютона к данным терминам:

Определения понятий «теория» и «гипотеза»

Теория	**Гипотеза**
Утверждение тогда и только тогда является теорией, когда оно удовлетворяет всем следующим критериям:	Утверждение тогда и только тогда является гипотезой, когда оно удовлетворяем одному или нескольким следующим критериям:
Т1. Это утверждение точно	Х1. Это утверждение в лучшем

является истиной, ибо оно было достоверно выведено из экспериментов.	случае хотя бы высоко вероятно является правдой.
Т2. Это утверждение экспериментально - то есть оно имеет экспериментально тестируемые последствия.	Х2. Это догадка или предположение - это то, что не основано на экспериментальных свидетельствах.
Т3. Это утверждение относится к измеримым и наблюдаемым свойствам вещи, а не к её "природе".	Х3. Это утверждение имеет отношение к "природе" вещи, а не к наблюдаемым, измеряемым её свойствам.

Ньютон считал свою «теорию универсальной гравитации» именно теорией, ибо она может быть подтверждена экспериментами. Но с другой стороны, объяснения, причины этого феномена он относил к гипотезам, ибо это уже относилось к объяснению природы явления гравитации, так как возможности для измерения или подтверждения любых утверждений о причинах возникновения гравитации экспериментально в те времена не существовало. Другими словами *гипотеза* о природе гравитации пытается ответить на вопросы: «Почему гравитация есть?» и «Что является причиной гравитации?», а *теория* гравитации отвечает на вопросы: «Существует или нет гравитация?», «Насколько сильна гравитация?» «Как измерить гравитацию?».

Бритва Оккама для проверки гипотез

- Существуют принципы, например, Бритва Оккама, которые являются не аксиомами, а **презумпциями, то есть они не запрещают более сложные объяснения явлений в принципе, а лишь рекомендуют *порядок рассмотрения* гипотез, который в большинстве случаев является наилучшим**. Альберт Эйнштейн так сформулировал принцип бритвы Оккама: «Всё следует упрощать до тех пор, пока это возможно, но не более того». Переформулированный на языке теории информации, принцип бритвы Оккама гласит, что самым точным сообщением является сообщение минимальной длины. Есть и другие.
- Среди наиболее известных примеров применения этого принципа — ответ, который дал императору Наполеону

создатель первой теории возникновения Солнечной системы математик и физик Лаплас. Наполеон спросил, почему слово «Бог», беспрерывно повторяемое Лагранжем, в его сочинении не встречается вовсе, на что Лаплас ответил: «Это потому, что я в этой гипотезе не нуждался».

Значение термина «бритва»

В философии под термином «бритва» понимается инструмент, помогающий отбрасывать (сбривать) маловероятные, неправдоподобные объяснения (гипотезы). А так как инструментом для бритья является бритва, лезвие (razor), то и на инструмент установления истины было перенесено то же название.

Примеры других «бритв»: Принцип фальсифицируемости Поппера, бритва Хэнлона, бритва Хитченса.

Научная гипотеза

Логическое предположение, для того, чтобы считаться научной гипотезой, должно удовлетворять следующим критериям:

1. Объяснять все имеющиеся в предметной области гипотезы факты.
2. Не должно иметь логических противоречий и противоречить фундаментальным положениям науки.
3. Должно быть принципиально проверяемым.
4. Не должно противоречить ранее установленным фактам, для объяснения которых оно не предназначено.
5. Должно быть приложимо к возможно более широкому классу явлений.

1.2.4.4. Общие принципы подготовки и планирования экспериментов

С момента своего появления наука ищет пути к познанию законов окружающего мира. Совершая одно открытие за другим, ученые поднимаются все выше и выше по лестнице знания, стирая границу неизвестности и выходя на новые рубежи науки. **Этот путь лежит через эксперимент**. Сознательно ограничивая бесконечное

разнообразие природы искусственными рамками научного опыта, мы превращаем его в понятную для человеческого разума картину мира.

Эксперимент как научное исследование - это форма, в которой и посредством которой наука существует и развивается. Эксперимент требует тщательной подготовки перед его проведением. В биомедицинских исследованиях планирование экспериментальной части исследования имеет особенно большое значение по причине широкой вариабельности свойств, характерной для биологических объектов. Эта особенность является основной причиной трудностей при интерпретации результатов, которые могут значительно различаться от опыта к опыту.

Статистические проблемы обосновывают необходимость выбора такой схемы эксперимента, которая минимизировала бы влияние вариабельности на выводы ученого. Поэтому **цель планирования эксперимента заключается в создании схемы, которая необходима для получения как можно большей информации при наименьших затратах для выполнения исследования.** Более точно **планирование эксперимента можно определить как процедуру выбора числа и условий проведения опытов, необходимых и достаточных для решения поставленной задачи с требуемой точностью.**

Планирование эксперимента появилось в агробиологии и связано с именем английского статистика и биолога сэра Рональда Эйлмера Фишера. В начале XX века на агробиологической станции в Ротамстеде (Великобритания) начались исследования влияния удобрений на урожайность различных сортов зерновых. Ученые вынуждены были считаться как с большой изменчивостью объектов исследования, так и с большой продолжительностью опытов (около года). В этих условиях не было иного пути, кроме разработки продуманного плана эксперимента для уменьшения негативного влияния указанных факторов на точность выводов. Применив статистические знания к биологическим проблемам, Фишер пришел к разработке собственных принципов теории статистического вывода и положил начало новой науке о планировании и анализе экспериментов.

Сам Рональд Фишер объяснял основы планирования на примере эксперимента произведенного для выяснения способности некой английской леди различать, что было налито в чашку в первую очередь - чай или молоко. Следует отметить, что для настоящих английских

леди важно, чтобы чай наливался в молоко, а не наоборот, нарушение последовательности будет признаком невежества и испортит вкус напитка.

Эксперимент проходит просто: леди пробует чай с молоком и по вкусу пытается понять, в какой очередности были налиты оба ингредиента. План, разработанный для этого исследования, характеризуется рядом свойств.

Сравнение. Во многих исследованиях точное определение результата измерения затруднительно или невозможно. Так, например, леди не сможет количественно оценить качество чая, она будет сравнивать его с эталоном правильно приготовленного напитка, вкус которого знаком ей с детства. Как правило, в научном эксперименте объект сравнивается либо с неким заранее заданным стандартом, либо с контрольным объектом.

Рандомизация. Это очень важный момент в планировании. В нашем примере рандомизация относится к тому, в каком порядке представлять чашки на дегустацию. **Рандомизация необходима для того, чтобы стало возможным применение статистических методов для анализа результатов исследования**.

Репликация. Повторяемость - это необходимый компонент постановки эксперимента. Недопустимо делать выводы о способности к определению качества чая только по одной чашке. Результат каждого отдельного измерения (дегустации) несет в себе долю неопределенности, возникшей под влиянием множества случайных факторов. Следовательно, для выявления источника вариабельности необходимо провести несколько испытаний. С этим свойством связана чувствительность эксперимента. Фишер отмечал, что пока число чашек чая не превысит некоторого минимума, невозможно сделать какие-либо однозначные выводы.

Однородность. Несмотря на необходимость повторения измерений (репликация), их число не должно быть слишком велико, чтобы не утратилась однородность. Разность температур чашек, притупление вкуса и т. п. при превышении некоторого предельного числа повторений, могут затруднить анализ результатов эксперимента.

Стратификация. Выходя за рамки примера Р. Фишера к более абстрактному описанию экспериментального плана можно дополнительно указать такое свойство как стратификация (блокировка). Стратификация - это распределение экспериментальных единиц в относительно однородные группы (блоки, слои). Процедура стратификации позволяет минимизировать эффект известных нам неслучайных источников вариабельности. Внутри каждого блока ошибку эксперимента предполагают меньшей относительно варианта со случайным отбором для эксперимента такого же количества объектов. Например, при исследовании нового лекарственного препарата мы имеем два уровня фактора - «препарат» и «плацебо», которые назначаются мужчинам и женщинам. В данном случае пол - это блокирующий фактор, по которому происходит разделение исследуемых на подгруппы.

Описанные выше характеристики экспериментального плана полностью или частично относятся к любому научному эксперименту. Однако для начала работы недостаточно одного только знания об общих свойствах исследования, необходима более тщательная подготовка. Создание подробного руководства в рамках одной статьи невозможно, поэтому здесь будет изложена наиболее общая информация об этапах планирования эксперимента.

Любое исследование начинается с постановки цели. Выбор проблемы для изучения и ее формулировка повлияют как на дизайн исследования, так и на выводы, которые будут сделаны по его результатам. **В самом простом случае формулировка проблемы должна предполагать вопросы «Кто?», «Что?», «Когда?», «Почему?» и «Как?».**

В качестве иллюстрации важности данного этапа планирования можно привести исследование, в котором проводится сбор информации о дорожно-транспортных происшествиях. В зависимости от постановки цели, работа может быть направлена на разработку нового автомобиля либо нового дорожного покрытия. Несмотря на то, что используется один и тот же набор данных, постановка задачи и выводы существенно различаются в зависимости от формулировки проблемы.

После выбора цели работы следует определить так называемые зависимые переменные. Это переменные, которые будут измеряться при проведении исследования. Например, показатели

функционирования тех или иных систем организма человека или лабораторных животных (частота сердечных сокращений, артериальное давление, содержание ферментов в крови и т. п.), а также любые другие характеристики объектов исследования, изменение которых будет для нас информативно.

Поскольку есть зависимые переменные, то должны быть еще и независимые переменные. Другое их название - **факторы**. Факторами исследователь оперирует в эксперименте. Это может быть доза исследуемого препарата, уровень стресса, степень физической нагрузки и т. д. Взаимосвязь между фактором и зависимой переменной удобно представлять с помощью кибернетической системы, часто называемой «черный ящик».

Черный ящик - это система, механизм работы которой нам неизвестен. Однако исследователь имеет информацию о том, что происходит на входе и выходе черного ящика. При этом состояние выхода функционально зависит от состояния входа. Соответственно y1, y2, ..., yp - это зависимые переменные, величина которых зависит от факторов (независимых переменных x1, x2, ..., xk). Параметры w1, w2, ..., wn представляют собой возмущающие воздействия, не поддающиеся контролю или изменяющиеся со временем.

В общем виде это можно записать так: y=f(x1, x2, ..., xk).

Каждый фактор в опыте может принимать одно из нескольких значений. Такие значения называют *уровнями фактора*. Может оказаться, что фактор способен принимать бесконечное число значений (например, доза лекарственного препарата), однако на практике выбирается несколько дискретных уровней, количество которых зависит от задач конкретного опыта.

Фиксированный набор уровней факторов определяет одно из возможных состояний черного ящика. Вместе с тем, это есть условия проведения одного из возможных опытов. Если перебрать все возможные наборы таких состояний, то мы получим полное множество различных состояний данной системы, количество которых будет числом всех возможных экспериментов. Для того, чтобы вычислить количество возможных состояний, достаточно число уровней факторов q (если для всех факторов оно одинаково) возвести в степень количества факторов k.

$$N=qk$$

Совокупность всех возможных состояний определяет сложность черного ящика. Так, система из десяти факторов на четырех уровнях может находиться более чем в миллионе разных состояний. Очевидно, что в подобных случаях невозможно провести исследование, включающее все возможные опыты. Поэтому на этапе планирования решается вопрос о том, сколько опытов и каких именно необходимо провести для решения поставленной задачи.

Следует отметить, что свойства объекта исследования имеют существенное значение для эксперимента. Во-первых, нам надо иметь информацию о степени воспроизводимости результатов опытов с данным объектом. Для этого можно провести эксперимент, а затем повторить его через неравные промежутки времени и сравнить результаты. Если разброс значений не превышает наших требований к точности эксперимента, то объект удовлетворяет требованию воспроизводимости результатов. Другое требование к объекту - его управляемость. Управляемым считается объект, на котором можно провести активный эксперимент. В свою очередь, активный эксперимент - это такой эксперимент, в процессе которого исследователь имеет возможность выбора уровней факторов, представляющих для него интерес.

На практике не существует полностью управляемых объектов. Как уже говорилось выше, на реальный объект действуют как управляемые, так и неуправляемые факторы, что приводит к вариабельности результатов между отдельными объектами. Отделить случайные изменения от закономерных, вызванных различными уровнями независимых переменных, мы можем лишь с помощью статистических методов.

Но статистические методы эффективны лишь в определенных условиях. Одно их таких условий - это **требование некоего минимального размера выборок, используемых в проведении эксперимента.** Очевидно, что чем шире диапазон изменения признаков от объекта к объекту, тем больше должна быть повторность опыта, т. е. численность экспериментальных групп.

Поскольку, неоправданно большое число испытаний сделает исследование слишком дорогим, а недостаточный объем выборки может поставить под сомнение точность выводов, определение

необходимого объема выборок играет решающую роль в планировании эксперимента. Методы вычисления минимального объема выборок подробно описаны в специальной литературе, поэтому привести их в этой работе не представляется возможным. Тем не менее, следует упомянуть, что они требуют предварительного определения средней величины исследуемого показателя и ее ошибки. Источником такой информации могут послужить публикации о похожих исследованиях. Если они еще не проводились, то возникает необходимость в выполнении предварительного «пилотного» исследования для оценки вариабельности признака.

Следующий этап в планировании экспериментов - это рандомизация. Рандомизация представляет собой процесс используемый для группировки объектов таким образом, чтобы у каждого из них была равная вероятность попасть в контрольную или опытную группу. Другими словами, выбор участников исследования должен происходить случайно, чтобы исследование не было отклонено в сторону «предпочтительного» для исследователя результата.

Рандомизация помогает предотвратить смещения, обусловленные причинами, которые не были непосредственно учтены в плане эксперимента. Для этого, например, формирование экспериментальных групп лабораторных животных производится случайным образом. Однако полная рандомизация возможна далеко не всегда. Так, в клинических исследованиях принимают участие пациенты определенной возрастной группы, с заранее заданным диагнозом и тяжестью заболевания, а, следовательно, отбор участников не является случайным. Кроме того, ограничивают рандомизацию так называемые «блочные» планы экспериментов. Эти планы подразумевают, что отбор в каждый блок выполняется в соответствии с определенными неслучайными условиями, а случайный отбор объектов исследования возможен только внутри блоков. Процесс рандомизации легко осуществить с помощью специализированного статистического программного обеспечения или специальных таблиц.

Мы приходим к гипотезам либо путем индуктивного, либо путем дедуктивного рассуждения; это зависит от стадии процесса исследования, на которой мы находимся. Если для построения теории мы пользуемся методом проб и ошибок, гипотезы можно выдвигать с помощью процесса *индуктивного обобщения*. Например, отметив, что

в США уровень участия граждан в политической жизни в разных штатах различен и при этом пропорционален уровню индустриализации, мы могли бы расширить данное наблюдение и утверждать, что это отношение между переменными может быть найдено и при сравнении разных государств. Если найдутся данные, подтверждающие эту гипотезу, у нас будет больше оснований включить индустриализацию в качестве переменной в теорию, предназначенную для объяснения политической активности. Однако до тех пор, пока у нас нет теории, позволяющей ответить на вопрос, почему индустриализация и политическая активность связаны друг с другом, мы не можем использовать факт их связи для объяснения политической активности.

Гипотезы, полученные индуктивным путем, могут играть важную роль в *поисковом исследовании*, которое полезно при построении теорий, но они оказываются бесполезными для объяснения явлений. Как только мы построили теорию, соединяющую наши переменные в логически связную систему, мы можем выводить из нее гипотезы с помощью *дедуктивного рассуждения*. Так как эти гипотезы являются прогнозами о мире, логически следующими из теории, с которой мы работаем, то выявление фактов, подтверждающих гипотезу, помогает давать объяснение событиям, поскольку обнаружение таких фактов отражает валидность теоретической системы, из которой выведены гипотезы.

Дедуктивное рассуждение – очень хорошо разработанная дисциплина, и мы не будем здесь объяснять ее правила. Однако важно отметить, что дедуктивная логика – это процесс, с помощью которого может быть эксплицирована информация, содержащаяся в наборе утверждений.

1.2.4.5. Формулирование и построение теории

ФОРМУЛИРОВАНИЕ ТЕОРИИ

Сразу становится совершенно очевидной важность объединения нормативного и эмпирического подходов. Каковы те критерии, в соответствии с которыми одна исследовательская проблема считается более интересной, чем другая? И хотя в голову приходит множество

таких критериев, начиная отличных интересов исследователя и кончая интересами общества в целом, большинство критериев все же разбивается на два основных типа. Проблема заслуживает изучения либо потому, что она отвечает некоторой конкретной потребности, т. е. ее решение послужит лучшему теоретическому познанию явления, либо потому, что она отвечает определенной социальной потребности, т. е. ее решение может помочь нам справиться с тем или иным вопросом, встающим перед обществом.

Хотя эти два типа проблем, часто называемых **фундаментальными** и **прикладными исследованиями**, не являются взаимоисключающими (если вы занимаетесь одной проблемой, это не означает, что вы *ни в коем случае* не можете одновременно заниматься и другой), тем не менее они часто находятся в состоянии конкуренции. Можно, например, исследовать гипотетические факторы, вызывающие агрессию в условиях стресса, в целях разработки сложной прогностической модели человеческого поведения, а можно вместо этого сосредоточиться на причинах возникновения взрывов и способах их предотвращения. Можно детально изучать процессы принятия решений государственными деятелями с целью понимания феномена лидерства, а можно вместо этого сосредоточить внимание на выявлении решений, способных привести к войне, и на возможностях их избежать. Поскольку для изучения всех потенциально интересных или важных исследовательских проблем имеется слишком мало научных ресурсов (финансов, времени и квалифицированных специалистов), нередко возникает конфликт между необходимостью осуществить фундаментальное исследование (практические результаты, сколь бы ни были значительными, почти всегда ощущаются лишь косвенно и в отдаленном будущем) и необходимостью использовать наши научные знания в настоящий момент непосредственно, даже если при этом будет задержано или вовсе остановлено развитие науки. Выбор должен сделать конкретный исследователь в соответствии с его (или ее) собственной системой ценностей.

Определив характер проблем, с которыми мы хотим иметь дело, и характер результатов, которых мы хотим достичь, мы должны затем более конкретно сформулировать задачу исследования. Такое решение диктуется рядом соображений. Прежде всего, необходимо выделить тот аспект проблемы, который нас более всего интересует. После того, как радостное возбуждение, сопровождающее начало поисков, несколько угаснет, и до момента, когда впереди начинают маячить

ответы на поставленные вопросы, ежедневная рутинная работа может оказаться довольно скучной. В такие периоды собственно интерес к проблеме становится важным мотивом исследования, так сказать, интеллектуальной закуской, поддерживающей наши силы до подачи на стол основного блюда. Поскольку решение любой исследовательской проблемы требует изнурительной работы, одна из величайших ошибок, которую мы можем совершить, – это выбрать задачу, вызывающую у нас мало интереса.

После того как определена интересующая нас тема исследования, необходимо тщательно проанализировать различные элементы, или компоненты, этой темы и выявить те из них, которые могут иметь значение для нашего исследования. Для установления основных факторов поведения необходимо использовать наше умение наблюдать и делать выводы, а также, в особенности, проведенные ранее исследования по сходной тематике, как наши собственные, так и выполненные другими исследователями. Попробуем пояснить сказанное на примере.

Представим себе расположенный в центре пустыни городок под названием Малая Америка. В нем нет ничего, кроме станций обслуживания и ресторанов, протянувшихся на несколько миль от въезда в город до самого горизонта. Единственное, что можно сделать в Малой Америке, – это поесть и заправить машину.

Теперь предположим, что мы захотели исследовать поведение жителей Малой Америки на президентских выборах, чтобы уметь объяснить, почему одни голосуют за кандидата демократов, другие – за кандидата республиканцев. В этом упрощенном примере объекты нашего анализа (жители Малой Америки) отличаются друг от друга (помимо способа голосования) по двум параметрам: каждый из них является либо владельцем предприятия, либо рабочим и связан либо со станцией обслуживания, либо с рестораном. Каждый из этих факторов представляет собой характеристику отдельного человека. Один житель Малой Америки может быть (1) служащим (2) ресторана, который (3) голосует за демократов, тогда как другой – (1) владельцем (2) станции обслуживания, который (3) голосует за республиканцев. Поскольку мы хотим объяснить различия в голосовании, опираясь на различия между избирателями, мы должны сосредоточить свое внимание на всех тех факторах, которые могли бы иметь отношение к их выбору. В данном случае в нашем распоряжении имеется только два таких фактора:

статус служащего или владельца и отношение к станции обслуживания или к ресторану. Будем называть их соответственно: социально-экономическим статусом (СЭС) (при этом статус владельца выше статуса служащего) и родом занятий. Есть ли какие-либо основания считать, что, зная обе характеристики конкретного избирателя, мы сможем предсказать, кому он (или она) отдаст предпочтение при голосовании?

Чтобы ответить на этот вопрос, необходимо сделать две вещи. Во-первых, необходимо *поразмышлять*. Следует спросить себя: существуют ли какие-нибудь *логические допущения* ожидать, что один из этих факторов окажет влияние на голосование? Во-вторых, следует обратиться к литературе по политологии и посмотреть, есть ли в проводившихся ранее исследованиях по данной или смежным проблемам какие-либо *эмпирические данные*, указывающие, как тот или иной фактор влияет на поведение избирателей. В действительности в данном случае почти нет оснований считать, что на голосовании сколько-нибудь заметно скажется род занятий. Конечно, между теми, кто имеет отношение к станциям обслуживания, и теми, кто имеет отношение к ресторанам, вполне могут существовать различия, однако не похоже, чтобы эти различия имели большое влияние на результат президентских выборов. Не много найдется кандидатов в президенты, которые бы предлагали программу, направленную в защиту станций обслуживания и против ресторанов (или наоборот), так что не похоже, чтобы при прочих равных эта переменная как-то способствовала объяснению поведения на выборах. Совсем иначе обстоит дело со второй переменной – социально-экономическим статусом. Поскольку принято считать, что демократическая партия является партией трудящихся, а республиканская – партией деловых кругов, и поскольку избиратели с более высоким СЭС скорее будут голосовать за республиканцев, чем избиратели с более низким СЭС, вполне допустимо, что служащие скорее будут голосовать за кандидата демократов, а владельцы предприятий – за кандидата республиканцев. И действительно, имеющиеся исследования изобилуют примерами, демонстрирующими именно такое соотношение. Таким образом, и теоретические рассуждения, и эмпирические факты говорят об одном и том же. Отсюда проблема нашего исследования может быть сформулирована следующим образом: оказывает ли СЭС избирателя – жителя Малой Америки воздействие на то, за кого данный избиратель голосует на президентских выборах?

Конечно, в реальном мире люди отличаются друг от друга столь значительно, что это не может быть описано двумя-тремя признаками, однако проблема, возникающая при постановке задачи исследования, в принципе та же самая. Поскольку мы не имеем возможности измерить все мыслимые переменные, мы должны обдуманно и обоснованно выбрать из многих тысяч характеристик людей (или организаций) те немногие характеристики, которые, как представляется, помогут объяснить интересующие нас аспекты поведения. С помощью логических рассуждений и имеющихся в литературе данных мы должны попытаться предусмотреть и выявить те факторы, которые предположительно связаны с этим поведением. Поступая таким образом, мы не предрешаем результаты исследования, как это может показаться на первый взгляд, а, скорее, вырабатываем более продуктивный подход к проблеме, позволяющий определить пути, способные привести к успешному объяснению. Такой процесс доработки исследовательской проблемы через осуществление обоснованного выбора и носит название *формулирования теории*.

Теории создаются по двум причинам. Во-первых, мы надеемся с помощью теорий так упростить действительность, чтобы можно было как-то понять ее и тем самым контролировать либо приспосабливаться к ней. Во-вторых, после того как понимание действительности достигнуто, теории могут послужить руководством для проверки его правильности. Теории логически обосновывают ожидания, или прогнозы, относительно реального мира, которые посредством соответствующих методов исследования могут сопоставляться с действительностью. Когда прогнозы подтверждаются, получают подтверждение и те рассуждения, которые лежат в их основе, соответственно возрастает наша уверенность, что мы правильно уловили ход событий. Когда наши прогнозы оказываются неверными, мы начинаем сомневаться в своем понимании и ищем способы достичь правильного понимания событий.

Теории представляют собой *множества логически связанных символов, отражающих то, что, по нашему мнению, происходит в мире*. **Теории всего лишь интеллектуальные инструменты.** Это очень важно усвоить, поскольку таким образом мы получаем возможность осознать, что **теории ни в каком абсолютном смысле не являются ни истинными, ни ложными, а только более или менее полезными.** Точно так же, как существует несколько способов изготовить молоток, существует и множество путей разработки теорий, объясняющих политическую жизнь. Таким образом, бессмысленно

ожидать, что теорию можно открыть подобно тому, как мореплаватель открывает неизвестный остров. Почему? Да потому, что теории не существуют “во внешнем мире”, так чтобы их можно было открыть. **Они – создание человеческого воображения, тяжелого труда и иногда счастливого случая.**

Если теории столь необходимы для проведения добротного исследования и в то же время их нельзя обнаружить путем простого разглядывания на протяжении многих часов груды распечаток, то как же взяться за построение теории, которая бы вела к пониманию интересующих нас аспектов жизни? Какие процессы здесь задействованы? Ответ не совсем ясен и прост, поскольку теории строятся самыми разными способами. Мы не можем предъявить набор процедур для создания конструктивной теории так, как могли бы описать изготовление стереосистемы. Однако мы можем пояснить главные идеи, лежащие в основе процесса создания теорий, и наиболее важные этапы этого процесса. Первый из них – *концептуализация проблемы*.

ЛОГИКА ПОСТРОЕНИЯ ТЕОРИИ

Начав с **события или поведения**, которое мы хотим понять, мы должны прежде всего спросить себя, **какие из имеющихся знаний о явлении могли бы помочь объяснить его**. **Понимание достигается на основе собственного опыта, случайного наблюдения или творческого размышления. Еще чаще полезным становится систематическое изучение чужих достижений в данной области**. Полезные теории начинают свое существование с досконального изучения тех событий, которые мы хотим объяснить. Без такого рода знаний мы можем оказаться не в состоянии понять, что же следует объяснять, или не будем располагать указаниями о том, где искать реальные отношения, которые можно использовать для объяснения событий.

Массовые волнения, происходившие во многих городах США в конце 60-х годов, дают пример того, что знание фактов очень важно для правильной **концептуализации проблемы исследования**. Когда волнения только начались, многие официальные лица называли их выступлениями групп бедняков, не имеющих устойчивых связей с обществом. Если бы мы приняли такую интерпретацию и попытались проанализировать эти волнения, стоящую перед нами задачу можно

было бы сформулировать следующим образом: почему в американских городах сконцентрировалось так много “отбросов общества” и каким образом были спровоцированы выступления? Многие официальные лица приводили в качестве объяснения присутствие якобы каких-то посторонних агитаторов. Однако, когда социологи провели интервью в городах, где происходили волнения, оказалось, что участниками волнений были не только “отбросы общества”. Фактически состав участников волнений почти не отличался от состава негритянского населения этих городов. В свете этого факта задача нашего исследования становится в корне отличной от той, которая диктовалась интерпретацией событий как обусловленные участием в них “отбросов общества”. В этом случае мы должны попытаться понять, что побудило обычных граждан негритянского происхождения, имеющих ту или иную работу, семью и другие общественные связи, принять участие в волнениях. Соответствующие объяснения опираются скорее на такие переменные, как реакция негритянского населения на расизм белых, чем присутствие посторонних агитаторов.

В данном примере неадекватные знания о фактах могли направить наши действия по созданию теории в совершенно неверном направлении. Вот почему столь важным является **поисковое исследование**, цель которого – установление соответствующих фактов. По этой же причине (если мы хотим строить надежные теории) необходимо искать информацию об исследуемых явлениях в литературе.

И все-таки как именно строится теория, объясняющая наблюдаемые явления, после того как все доступные факты оказываются в нашем распоряжении? Обычно мы начинаем с поиска фактов для тех моделей, которые могут объяснить наблюдаемые события.

Например, мы хотим узнать, каковы были причины политических выступлений в университетских городках. Чтобы ответить на этот вопрос, необходимо объяснить, что заставило студентов участвовать в этих выступлениях. Если бы мы сами были участниками таких выступлений или знали кого-либо из участников, у нас могли бы возникнуть некоторые соображения относительно побудительных мотивов выступлений, однако объяснение того, почему в них участвовали большие массы студентов, потребовало бы информации о гораздо большем числе людей. Чтобы дать такое объяснение, гораздо разумнее было бы собрать данные о характеристиках и мотивах

выступлений, *общих для всех* участвовавших в них студентов. Если среди участников обнаружились общие свойства, отличающие их от остальных студентов, мы можем заключить, что именно они и приводят к участию в демонстрациях. При этом особая роль таких характеристик становится частью объяснения того, почему происходят выступления.

Переход от обобщения того, что мы наблюдаем, к тому, . чего мы не наблюдаем или не можем наблюдать, называется **индукцией**.
Индукция составляет базис научной теории. Теории, построенные на основании наблюдений с помощью индукции, называются *эмпирически обоснованными*. В процессе индукции, исходя из наших знаний о некоторых ситуациях, мы делаем вывод о том, как могли бы обстоять дела в других, сходных ситуациях. Мы делаем логический скачок от того, что видели, к прогнозу относительно того, чего не видели, базируясь на предположении, что в основе всех событий реального мира лежит некая постоянная глубинная модель. В своей повседневной жизни мы все пользуемся индукцией. Если пять раз подряд мы наблюдаем, что после нажатия кнопки на стене дверь лифта открывается, мы быстро сделаем из этого вывод, что нажатие кнопки вызывает открывание двери. Здесь имеет место индуктивное обобщение – переход от нескольких случаев, которые мы наблюдали (пятикратное нажатие кнопки), к случаям, которые мы не наблюдали (нажатие кнопки большее число раз и нажатие кнопок лифтов в других зданиях). Однако для создания теории нужна не только индукция, поскольку отсылка к фактам еще не дает объяснения, если только мы не в состоянии показать, *почему* эти факты приводят к наблюдаемым результатам. Вернемся к примеру со студенческими выступлениями. Предположим, мы обнаруживаем, что их участники, как правило, были в большей степени недовольны государственной политикой, чем те, кто не принимал в них участия, и что они, кроме того, в гораздо меньшей степени верили в эффективность обычных способов достижения политических изменений. Установление этого факта служит объяснением причины выступлений только в том случае, если мы можем показать, почему подобная ситуация должна вести к выступлениям. Чтобы продемонстрировать это, вероятно, понадобится сделать ряд *предположений* о политическом поведении, а точнее, окажется необходимым сделать прогноз о том, что для изменения политики, с которой люди не согласны, они будут предпринимать определенные действия и что, не видя никаких изменений в политике

под воздействием обычных политических методов (голосование, написание писем и т. п.), они перейдут к открытым выступлениям.

В последующем эти предположения (называемые иногда *аксиомами* или *постулатами*) входят в состав нашей теории. Они описывают условия, при которых в соответствии с нашими ожиданиями полученные нами предварительные объяснения подтверждаются имеющимися данными. Делая общие утверждения о политическом поведении в определенных условиях, эти положения объясняют, почему мы ожидаем студенческих выступлений, опираясь на то, что мы знаем о студентах университетских городков. Теперь мы можем объяснить конкретный способ поведения (выступление), показав, что он логически вытекает из ряда теоретических предположений.

Поступая таким образом, мы совершаем действие, обратное тому, которое совершали при индуктивном рассуждении. Здесь мы движемся от абстрактных утверждений, касающихся общих взаимосвязей, к конкретным утверждениям, касающимся специфических типов поведения. Этот процесс рассуждения от абстрактного и общего к конкретному и специфическому известен под названием **дедукции**. **Мы все пользуемся дедуктивной логикой в повседневной жизни**. Если мы предполагаем, что работа лифтов управляется системой настенных кнопок, и оказываемся перед лифтом, мы обычно делаем вывод, что для попадания в лифт необходимо нажать имеющуюся кнопку. От обобщения мы перешли здесь к прогнозу относительно конкретного события с помощью дедукции.
Дедукция -это процесс, позволяющий нам использовать теории для объяснения событий реального мира. Если с помощью процесса дедукции мы в состоянии продемонстрировать, что некоторое наблюдаемое событие может быть логически предсказано на основе ряда предположений, входящих в нашу теорию, то тем самым теория дает объяснение наблюдаемому явлению. Теория помогает понять событие, обосновывая, почему оно именно такое, какое оно есть. **Дедукция предназначена для осуществления связи между теорией и нашими наблюдениями.**

Однако сама по себе разработка теории еще не делает ее эффективной. Обычно мы подходим к объяснению некоторого события, располагая многими теориями. В таком случае необходимо задать вопрос, какие из этих теорий больше всего помогают нам в понимании действительности. Ответ на этот вопрос требует проверки

альтернативных теорий фактами действительности. **Прежде чем обсуждать проверку теорий, необходимо понять две вещи. Во-первых, для того чтобы заниматься созданием теорий, надо выяснить, что определяет полезность теории. Во-вторых, нам следует выяснить, как компоненты теории связаны друг с другом и с эмпирическими исследованиями.** Чтобы быть полезной для объяснения наблюдений, теория должна отвечать следующим требованиям:

1. Теория должна быть *верифицируемой*. Можно ли, исходя из теории, сделать прогнозы относительно действительности, достаточно конкретные и специфические, так чтобы мы могли провести наблюдения, либо подтверждающие, либо опровергающие их? Может ли теория быть связана с действительностью систематическим образом или она представляет собой всего лишь множество абстракций?

2. Теория должна быть *логически непротиворечивой*. Является ли теория внутренне последовательной? Являются ли ее предположения совместимыми друг с другом, а входящие в нее термины – однозначными?

3. Теория должна быть *доступной*. Могут ли другие, должным образом обученные люди понять теорию так, чтобы иметь возможность использовать ее для объяснения событий и заниматься проверкой вытекающих из нее гипотез?

4. Теория должна быть *общей*. Можно ли использовать ее для объяснения разнообразных событий, происходящих в разных местах и в разное время? Можно ли, основываясь на ней, строить прогнозы, которые легко проверить при различных условиях, или она жестко привязана к одному виду наблюдений?

5. Теория должна быть *экономичной*. Достаточно ли она проста, чтобы быть легко применимой и понятной, или она столь сложна, столь переполнена условиями и исключениями, что эксплицитные ожидания о событиях реальной действительности извлекаются из нее с трудом?

Теории могут обладать этими желательными характеристиками в разной степени, и иногда при разработке конкретной теории нам приходится отдавать предпочтение одним характеристикам в ущерб другим. Мы, например, можем поступиться экономичностью в пользу

большей общности или верифицируемости. Чтобы результаты нашего труда действительно приносили пользу, мы должны, формулируя теории, иметь в виду все перечисленные выше требования.

КОМПОНЕНТЫ ТЕОРИИ

Теория состоит из множества *понятий*, связанных *утверждениями*, логически выведенными из множества *предположений*. Такова *логическая структура* теории. Именно эта структура позволяет нам использовать теорию для объяснения событий, поскольку дает возможность обосновывать, почему мы вправе логически ожидать именно того положения дел, которое имеет место.

Поиски полезной теории начинаются с решений, которые мы принимаем относительно строительных блоков теорий – **понятий**. **Понятие** – это просто *слово или символ, который обозначает некоторое* ***представление***. В понятиях нет ничего мистического. Мы пользуемся ими ежедневно, облегчая себе взаимодействие с многоплановой действительностью за счет того, что подводим под некоторую категорию встречающиеся нам объекты в соответствии с некоторыми релевантными для нас их свойствами. Четвероногих, которых мы видим, мы относим к коровам, кошкам, собакам и другим видам животных, и такая классификация сама по себе является основой для возникновения некоторых важных ожиданий (например: от собак не стоит ждать молока). Приписывая наименование тому или иному объекту, мы можем делать относительно него некоторые прогнозы, поскольку **наименование представляет собой символ определенного сочетания свойств.**

Той же самой цели служат социологические понятия. Они указывают свойства объектов (людей, политических систем, выборов), релевантные для конкретного исследования. Одного исследователя могут интересовать личностные характеристики человека, другого – идентификация данным человеком его партийной принадлежности, третьему наиболее интересен уровень политического отчуждения личности. Человек характеризуется всеми этими свойствами: личностными характеристиками, партийной принадлежностью, степенью отчуждения и многими другими, но лишь отдельные из них релевантны для каждого конкретного случая. Все исследователи имеют дело с одной и той же реальной действительностью, они лишь производят отбор, чтобы по-разному организовать свои наблюдения.

Понятия помогают решить, какие из многочисленных свойств или признаков существенны для нашего исследования.

Дело в том, что понятия, как и теории, не живут собственной жизнью. Они – инструменты, которые мы создаем для определенных целей, и их нельзя назвать истинными или ложными, можно лишь считать их более или менее полезными. Что делает понятие полезным? На этот счет имеется три основных соображения.

Во-первых, постольку мы занимаемся *эмпирическим* исследованием, понятие, для того чтобы оно приносило пользу, должно относиться к явлениям, по крайней мере потенциально *наблюдаемым*. В Средние века важную роль при объяснении событий играло понятие Божьей Воли. Однако верифицировать подобные объяснения мы не можем, поскольку не в состоянии наблюдать Божью Волю, чтобы утверждать ее наличие или отсутствие в каждом данном случае. **Чтобы иметь хотя бы какую-то научную ценность, понятие должно соотноситься с чем-то, что так или иначе измеряется с помощью наших обычных органов чувств.**

Это не означает, что все понятия должны относиться к непосредственно наблюдаемым объектам. Ряд очень существенных для наук понятий относится к свойствам, которые невозможно наблюдать непосредственно. У людей просто нет классового статуса в том смысле, в каком у них есть рыжие волосы, однако, если мы располагаем о них определенными сведениями (например, сведениями об их доходе или их специальности), мы можем *сделать вывод* о том, каков их классовый статус. Точно так же в государстве не бывает авторитарной или демократической политической системы в том же смысле, в каком его территория характеризуется наличием гор или пустынь, однако можно сделать вывод о степени демократичности данного государства, изучив определенные стороны его политической жизни (например, характер выборов и обеспечение гражданских свобод).

Возникает следующий вопрос: можем ли мы разработать набор процедур, использующих наши органы чувств для сбора информации, которая бы позволила нам определить наличие или отсутствие в реальном мире объекта, к которому относится понятие, или величину этого объекта? Если мы в состоянии сделать это для некоторого понятия, то, собственно говоря, это **понятие имеет эмпирические**

референты: **оно относится к тому, что можно прямо или косвенно наблюдать.**

Во-вторых, понятия (помимо наличия у них эмпирических референтов) должны быть ***точными*.** Они должны относиться к одному, и только к одному множеству свойств некоторого явления. **Мы должны иметь возможность точно знать, о чем идет речь, когда для описания некоторого объекта используется некоторое понятие.** Участвует ли в описании понятия социального класса раса или классовый статус полностью определяется факторами (такими, как доход к образованию), к числу которых раса не относится? Учитывается ли степень неравенства в распределении материальных благ, когда политическая система государства характеризуется как демократическая или авторитарная, или же характер политической системы полностью определяется другими факторами? **Точность очень важна, поскольку она указывает на то, за чем следует вести наблюдение, чтобы увидеть, как проявляется понятие в каждом конкретном случае. Только в том случае, если мы это увидим, мы сможем использовать понятие в объяснениях, имеющих эмпирические основания.**

Точность помогает также идентифицировать эмпирические референты и проводить разграничения среди наблюдаемых явлений. Если степень демократичности определяется лишь наличием или отсутствием всенародных выборов официальных лиц, то демократическим государством являются и бывший Советский Союз, и Соединенные Штаты. Хотим ли мы в нашем исследовании рассматривать эти два государства как примеры одного и того же типа политической системы? Если нет, мы должны усовершенствовать используемое понятие, сделать его более точным, так чтобы в нашем исследовании можно было провести различие между этими двумя государствами.

И наконец, в-третьих, полезные понятия должны быть ***теоретически значимыми***. Понятие теоретически значимо тогда, когда оно связано с достаточно большим числом других понятий данной теории, чтобы играть важную роль в объяснении наблюдаемых событий.

Давая гипотетическое объяснение студенческим выступлениям, мы использовали два понятия: *глубину политического недовольства* и *оценку эффективности влияния обычной политической деятельности на изменения в политике*. Эти два понятия были связаны друг с другом

посредством предположений, состоящих в том, что люди будут предпринимать какие-то действия для изменения политики, вызывающей у них протест, и что, поняв бесполезность других средств воздействия, они перейдут к от крытым выступлениям. Приняв во внимание эти предположения и обнаружив соответствующее сочетание взаимосвязей, о которых мы говорили, мы будем вынуждены ожидать открытых выступлений. Каждое из понятий здесь существенно для объяснения и связано как с теоретическими предположениями, так и со вторым понятием. Каждое понятие теоретически значимо, так как необходимо для объяснения.

Таким образом, становится очевидным, что **теория определяет полезность понятий, связывая их друг с другом, так чтобы их можно было использовать в формулировках объяснений. Теория связывает понятия, устанавливая между ними определенные *отношения*. Эти отношения имеют форму *утверждений*, выведенных из наших предположений.**

Утверждения обычно устанавливают между понятиями отношение одного из двух основных типов: *ковариации* или *каузации*. **Ковариационные отношения** указывают, что два или более понятий имеют тенденцию изменяться одновременно: когда увеличивается (уменьшается) одно, увеличивается (уменьшается) и другое. Ковариационные отношения не несут никакой информации о причинах одновременного изменения отношений. Например, можно предсказать, что в отношении ковариации находятся уровень политической информированности и вероятность участия в голосовании: когда возрастает одно, возрастает и второе. Однако что же при этом происходит? Люди с большей вероятностью будут голосовать, так как они лучше информированы? Или они заняты поисками информации, так как собираются принять участие в голосовании и хотят принять надежное решение? Или же и уровень информации, и вероятность голосования зависят от некоего третьего фактора, например интереса к политике или осознания общественного долга? Такое предположение не содержит никакой информации.

Каузальные (причинные) отношения указывают, что изменения в одном или нескольких понятиях приведут к изменениям в одном или нескольких других понятиях. Например, мы могли бы утверждать, что, чем сильнее у человека развита идентификация партийной принадлежности, тем выше вероятность того, что он (или она) примет

участие в голосовании. Осознание того, что ты являешься членом партии, может заставить человека принять участие в голосовании, однако высокая вероятность участия в голосовании не формирует идентификации партийной принадлежности.

Мы все в обыденной жизни привыкли рассуждать в терминах причины и следствия, однако, как правило, эти понятия используются нестрого. Часто бывает крайне трудно установить причины или последствия человеческих поступков: чем значительнее событие, тем более трудным может оказаться установление его причин. Что является причиной войны, общественного движения или образования новой политической партии?

Вследствие всех этих сложностей мы должны с осторожностью постулировать каузальные отношения и лишь в тех случаях, когда одновременно выполняются **четыре условия**. **Во-первых**, постулированные причина и следствие должны изменяться вместе, т.е. находиться в отношении ковариации. **Во-вторых**, причина должна предшествовать следствию. **В-третьих**, мы должны иметь возможность идентифицировать каузальную связь между предполагаемой причиной и следствием. (Это означает, что мы должны иметь возможность идентифицировать *процесс*, посредством которого изменения *A* вызывают изменения *B*.) **В-четвертых**, ковариация между явлениями причины и следствия не должна возникать из-за их одновременной соотнесенности каким-то третьим фактором.

Это последнее условие напоминает нам о проблеме *мнимых отношений*. Когда *A* и *B* изменяются вместе, поскольку оба они вызваны *C*, а в отсутствие *C* они совместно не изменяются, то усматриваемое нами отношение между *A* и *B* называется мнимым. Очень важно внимательно следить за предположениями, которые мы выдвигаем, пытаясь выявить возможные мнимые отношения, прежде чем включать их в теорию, посчитав результатом каузального взаимодействия. Классическим примером мнимого отношения служит следующий случай: вначале исследователь обнаруживает, что цена импортного рома и жалованье министров испытывают одновременные колебания, и затем делает вывод, что изменение цен на ром вызывает изменение министерского жалованья. Гораздо вероятнее, что и цены на ром, и жалованье министров изменяются в результате изменения общих экономических условий и общего уровня цен. Отношение

между первыми двумя переменными – это отношение ковариации, а не отношение каузации.

Важно понимать еще две особенности социальной каузации. Во-первых, одно явление может вызывать другое либо прямо, либо косвенно *A* может вызвать *B* лишь в том смысле, что будет являться причиной *C*, которое уже непосредственно будет вызывать *B*. Чтобы строить по возможности полные теории, необходимо очень внимательно следить за **косвенной каузацией**. Во-вторых, следует учитывать тот факт, что поведение человека обычно бывает обусловлено более чем одной причиной. При разработке теории необходимо избегать излишнего упрощения и отводить должное место в общественной жизни **множественной каузации**. Это означает, что любое событие может иметь несколько различных причин и что осуществление некоторого события иногда требует одновременного осуществления многих событий.

Для того чтобы справиться со всеми этими трудностями, обычно бывает полезно построить **каузальную модель** *теории*. **Каузальная модель – это диаграмма, которая в явном виде задаст все отношения, принятые в теории, и, таким образом , все следствия, вытекающие из наших посылок, становятся более наглядными**. Каждая стрелка модели изображает каузальное воздействие, а направление стрелки указывает, какая переменная в нашей теории является зависимой, а какая – независимой. И ковариационные, и каузальные отношения могут быть как *положительными*, так и *отрицательными*. Это означает, что два приятия могут изменяться либо в одном и том же направлении, либо в противоположных. Если они изменяются в одном и том же направлении, отношение считается **положительным**. Положительное отношение представлено утверждением: чем *сильнее* относительное неравноправие национальных меньшинств в обществе, тем *выше* вероятность политического насилия. **Отрицательное** отношение представлено утверждением: чем *сильнее* степень политического отчуждения, испытываемого человеком, тем *ниже* вероятность того, что он (или она) будет принимать участие в традиционной политической деятельности. В теории должно быть указано, какое именно отношение между понятиями (положительное или отрицательное) нами ожидается. Эта информация может быть добавлена к каузальной диаграмме с помощью знаков плюс (+) или минус (–), приписанных каждой стрела к указывающих на то, положительным или отрицательным мыслится данное отношение.

1.2.4.6. Разработка (выбор) метода

Ме́тод (от др.-греч. μέθοδος —путь исследования *или* познания, от μετά- + ὁδός «путь») — осознание формы внутреннего саморазвития содержания изучаемого предмета.

В отличие от области знаний или исследований, является авторским, то есть созданным конкретной персоной или группой персон, научной или практической школой. В силу своей ограниченности рамками действия и результата, методы имеют тенденцию устаревать, преобразовываясь в другие методы, развиваясь в соответствии со временем, достижениями технической и научной мысли, потребностями общества. **Совокупность однородных методов принято называть подходом**. **Развитие методов является естественным следствием развития научной мысли**.

Нау́чный ме́тод — система категорий, ценностей, регулятивных принципов, методов обоснования, образцов и т.д., которыми руководствуется в своей деятельности научное сообщество.

Метод включает в себя способы исследования феноменов, систематизацию, корректировку новых и полученных ранее знаний. Умозаключения и выводы делаются с помощью правил и принципов рассуждения на основе эмпирических (наблюдаемых и измеряемых) данных об объекте. Базой получения данных являются наблюдения и эксперименты. Для объяснения наблюдаемых фактов выдвигаются гипотезы и строятся теории, на основании которых в свою очередь строится модель изучаемого объекта.

Важной стороной научного метода, его неотъемлемой частью для любой науки, является требование объективности, исключающее субъективное толкование результатов. Не должны приниматься на веру какие-либо утверждения, даже если они исходят от авторитетных учёных. Для обеспечения независимой проверки проводится документирование наблюдений, обеспечивается доступность для других учёных всех исходных данных, методик и результатов исследований. Это позволяет не только получить дополнительное подтверждение путём воспроизведения экспериментов, но и критически оценить степень адекватности (валидности) экспериментов и результатов по отношению к проверяемой теории.

Отдельные части научного метода применялись ещё философами древней Греции. Ими были разработаны правила логики и принципы ведения спора. При этом выводам, полученным в результате рассуждений, отдавалось предпочтение по сравнению с наблюдаемой практикой. Знаменитым примером предпочтения рассуждения над практикой является утверждение, что быстроногий Ахиллес никогда не догонит черепаху.

Вершиной развития логики высказываний стала софистика. Однако целью софистов была не столько истина, сколько победа в судебных процессах, где формализм превышал любой другой подход.

Сократ создал сократический метод ведения спора. В противовес софистам, которые пытались навязать и доказать свою точку зрения, Сократ пытался наводящими вопросами заставить оппонента самостоятельно прийти к новым выводам и изменить свои первоначальные взгляды. Сократ считал свой метод искусством извлекать скрытое в каждом человеке знание с помощью наводящих вопросов. Ему приписывают высказывание о том, что в споре рождается истина.

В XX веке была сформулирована гипотетически-дедуктивная модель научного метода, состоящая в последовательном применении следующих шагов:

1. Используйте опыт: Рассмотрите проблему и попытайтесь осмыслить её. Найдите известные ранее объяснения. Если это новая для вас проблема, переходите к шагу 2.
2. Сформулируйте предположение: Если ничего из известного не подходит, попробуйте сформулировать объяснение, изложите его кому-то другому или в своих записях.
3. Сделайте выводы из предположения: Если предположение (шаг 2) истинно, какие из него следствия, выводы, прогнозы можно сделать по правилам логики?
4. Проверка: Найдите факты, противоречащие каждому из этих выводов, с тем чтобы опровергнуть гипотезу (шаг 2) (см. фальсифицируемость). Использование выводов (шаг 3) в качестве доказательств гипотезы (шаг 2) является логической ошибкой. Эта ошибка называется «подтверждение следствием» (англ. *Affirming the consequent*, греч. Επιβεβαίωση του επομένου)

Около тысячи лет назад Ибн ал-Хайсам продемонстрировал важность 1-го и 4-го шагов. Галилей в трактате «Беседы и математические обоснования двух новых наук, касающихся механики и законов падения» (1638) также показал важность 4-го шага (называемого также *эксперимент*). Шаги метода можно выполнять по порядку — 1, 2, 3, 4. Если по итогам шага 4 выводы из шага 3 выдержали проверку, можно продолжить и перейти снова к 3-му, затем 4-му, 1-му и так далее шагам. Но если итоги проверки из шага 4 показали ложность прогнозов из шага 3, следует вернуться к шагу 2 и попытаться сформулировать новую гипотезу («новый шаг 2»), на шаге 3 обосновать на основе гипотезы новые предположения («новый шаг 3»), проверить их на шаге 4 и так далее.

Следует заметить, что если следовать критерию Поппера, то при учёте полной группы событий и невозможности всеобъемлющего восприятия действительности, научный метод никогда не сможет абсолютно *верифицировать* (доказать истинность) гипотезы (шаг 2); возможно лишь опровергнуть гипотезу — доказать её ложность.

Элементы научного метода

Теории

Тео́рия (др.-греч. θεωρία «рассмотрение, исследование») — система знаний, обладающая предсказательной силой в отношении какого-либо явления. Теории формулируются, разрабатываются и проверяются в соответствии с научным методом.

Стандартный метод проверки теорий — прямая экспериментальная проверка («эксперимент — критерий истины»). Однако часто теорию нельзя проверить прямым экспериментом (например, теорию о возникновении жизни на Земле), либо такая проверка слишком сложна или затратна (макроэкономические и социальные теории), и поэтому теории часто проверяются не прямым экспериментом, а по наличию предсказательной силы — то есть если из неё следуют неизвестные/незамеченные ранее события, и при пристальном наблюдении эти события обнаруживаются, то предсказательная сила присутствует.

Гипотезы

Гипо́теза (от др.-греч. ὑπόθεσις — «основание», «предположение») — недоказанное утверждение, предположение или догадка.

Как правило, гипотеза высказывается на основе ряда подтверждающих её наблюдений (примеров) и поэтому выглядит правдоподобно. Гипотезу впоследствии или доказывают, превращая её в установленный факт, или же опровергают (например, указывая контрпример), переводя в разряд ложных утверждений.

Недоказанная и неопровергнутая гипотеза называется открытой проблемой.

Научные законы

Зако́н — вербальное и/или математически сформулированное утверждение, которое описывает соотношения, связи между различными научными понятиями, предложенное в качестве объяснения фактов и признанное на данном этапе научным сообществом согласующимся с экспериментальными данными. Непроверенное научное утверждение называют гипотезой.

Научное моделирование

Моделирование — это изучение объекта посредством моделей с переносом полученных знаний на оригинал. **Предметное моделирование** — создание моделей уменьшенных копий с определёнными свойствами, дублирующими оригинальные.
Мысленное моделирование — с использованием мысленных образов.
Знаковое или символическое — представляет собой использование формул, чертежей.
Компьютерное — компьютер является и средством, и объектом изучения, **моделью является компьютерная программа.**

Построение математической модели позволяет систематизировать существующие данные и сформулировать прогнозы, необходимые для поиска новых. Ярким примером этого является таблица Менделеева, по которой было прогнозировано существование множества ранее неизвестных элементов.

Полученные из свойств математической модели прогнозы проверяются экспериментом или сбором новых фактов.

Эксперименты

Эксперимéнт (от лат. *experimentum* — проба, опыт) в научном методе — набор действий и наблюдений, выполняемых для проверки (истинности или ложности) гипотезы или научного исследования причинных связей между феноменами. Эксперимент является краеугольным камнем эмпирического подхода к знанию. Критерий Поппера выдвигает в качестве главного отличия научной теории от псевдонаучной возможность постановки эксперимента, прежде всего такого, который может дать опровергающий эту теорию результат. Одно из главных требований к эксперименту — его воспроизводимость.

Эксперимент **делится на следующие этапы**:

- Сбор информации;
- Анализ;
- Выработка гипотезы, чтобы объяснить явление;
- Разработка теории, объясняющей феномен, основанный на предположениях, в более широком плане.

Научные исследования

Научное исследование — процесс изучения результатов наблюдений, экспериментов, концептуализации и проверки теории, связанный с получением научных знаний.

Виды исследований:

- Фундаментальное исследование, предпринятое главным образом, чтобы производить новые знания независимо от перспектив применения.
- Прикладное исследование.

Наблюдения

Наблюдение — это целенаправленный процесс восприятия предметов действительности, результаты которого фиксируются в описании. Для получения значимых результатов необходимо многократное наблюдение.

Виды наблюдений:

- непосредственное наблюдение, которое осуществляется без применения технических средств;
- опосредованное наблюдение — с использованием технических устройств.

Измерения

Измерение — это определение количественных значений свойств объекта с использованием специальных технических устройств и единиц измерения.

Истина и предубеждение

В XX веке некоторые исследователи, в частности Людвик Флек (1896—1961), отметили необходимость более тщательной оценки результатов проверки опытом, поскольку полученный результат может оказаться под влиянием наших предубеждений. Следовательно, необходимо быть более точным при описании условий и результатов проведения эксперимента. Выдающийся российский учёный, М. В. Ломоносов, придерживался мнения, что вера и наука дополняют друг друга: «Правда и вера суть две сестры родные, дщери одного Всевышнего Родителя, никогда между собою в распрю прийти не могут, разве кто из некоторого тщеславия и показания своего мудрования на них вражду всклеплет. А благоразумные и добрые люди должны рассматривать, нет ли какого способа к объяснению и отвращению мнимого между ними междоусобия.»

И сейчас среди учёных есть верующие люди, при том с довольно большим вкладом. Примером может быть директор проекта «Геном человека» Фрэнсис Колинз, написавший книгу «Доказательство Бога. Аргументы учёного», посвящённой вопросу совместимости религии и науки.

На сегодня предположение о божественном вмешательстве автоматически выводит теорию, использовавшую такое предположение, за пределы науки, потому что такое предположение является в принципе непроверяемым и неопровергаемым (то есть противоречит критерию Поппера). Научный метод подразумевает искать причины явлений исключительно в естественной области, без опоры на сверхъестественное. Академик Виталий Лазаревич Гинзбург: «Во всех известных мне случаях верующие физики и астрономы в своих научных работах ни словом не упоминают о Боге… Занимаясь конкретной научной деятельностью, верующий, по сути дела, забывает о Боге…»

Даже простая убеждённость в чём-либо на основе предыдущего опыта или знаний может изменять интерпретацию результатов наблюдения. Человек, имеющий определённое убеждение касательно некоего явления, часто склонен воспринимать факты в качестве доказательств своей веры уже только потому, что они ей прямо не противоречат. При анализе может оказаться, что предмет веры является лишь частным случаем более общих явлений (например, Корпускулярно-волновая теория считает частными случаями предшествовавшие представления о свете в форме частиц или волн) или вообще не связан с предметом наблюдения (например, концепция Теплорода в отношении температуры).

Не менее антинаучной может быть и идеологическая предубеждённость. Примером несовместимости подобной предубеждённости и научного метода является сессия ВАСХНИЛ 1948 года, в результате которой генетика в СССР оказалась под запретом до 1952 года и биологическая наука оказалась в застое почти на 20 лет. Один из основных тезисов «мичуринских» биологов во главе с Т. Д. Лысенко против генетики состоял в том, что основоположники классической теории наследственности (отнюдь не "идеалистической") Мендель, Вейсман и Морган якобы вследствие своего идеализма создали неправильную идеалистическую теорию с элементами мистики вместо правильной материалистической: «Как мы отмечали ранее, столкновение материалистического и идеалистического мировоззрений в биологической науке имело место на протяжении всей её истории… Для нас совершенно ясно, что основные положения менделизма-морганизма ложны. Они не отражают действительности живой природы и являют собой образец метафизики и идеализма…» Истинную идеологическую подоплёку морганистской генетики хорошо (невзначай для наших морганистов) вскрыл физик Э.

Шрёдингер. В своей книге «Что такое жизнь с точки зрения физики?», одобрительно излагая хромосомную вейсманистскую теорию, он пришёл к ряду философских выводов. Вот основной из них: «…личная индивидуальная душа равна вездесущей, всепостигающей, вечной душе». Это своё главное заключение Шрёдингер считает «…наибольшим из того, что может дать биолог, пытающийся одним ударом доказать и существование Бога и бессмертие души».

Критика научного метода

Ряд постпозитивистов в своих трудах во 2-й половине XX века сделали попытку применить критерии научного метода к самой науке на примере исторического материала реальных открытий. В результате появилась критика этого метода, которая, по мнению постпозитивистов, указывает на расхождение между методологией научного метода и реальным развитием научных идей. По их мнению, это свидетельствует об отсутствии полностью формализированного и достоверного метода, приводящего к более достоверному знанию, однозначной связи между принципами верификации/фальсификации и получением истинного знания.

Хотя постпозитивисты отказываются от понятия истины, тем не менее, другие методологи науки выражают надежду найти общие критерии, которые позволяли бы приблизиться к более адекватному описанию мира.

Явление парадигмы

Томас Кун считает, что научное знание развивается скачкообразно. Научная революция происходит тогда, когда учёные обнаруживают аномалии, которые невозможно объяснить при помощи старой парадигмы, в рамках которой до этого момента происходил научный прогресс. Развитие науки соответствует смене «психологических парадигм», взглядов на научную проблему, порождающих новые гипотезы и теории. Кун относит методы, которые влияют на переход от одной парадигмы к другой, в область социологии.

Утончённый фальсификационизм

Имре Лакатос, развивая на основе идей фальсификационизма Поппера свой *утончённый фальсификационизм*, пришёл к выводу, что одной из существенных проблем развития науки как системы, опирающуюся на какие-то единые методы, — является существование гипотез ad hoc. Это один из механизмов, при помощи которого преодолеваются противоречия между теорией и экспериментом. Из-за этих гипотез, которые фактически являются частью теории, временно выводятся из-под критики и становится невозможным опровержение таких теорий, так как противоречия теории и эксперимента объясняются гипотезой ad hoc и не опровергают теорию. С помощью этих гипотез становится невозможным полное опровержение ни одной теории. Возможно говорить только о временно́м сдвиге проблем: либо прогрессивном, либо регрессивном.

Догматический фальсификационист, в соответствии со своими правилами, должен отнести даже самые значительные научные теории к метафизике, где нет места рациональной дискуссии — если исходить из критериев рациональности, сводящихся к доказательствам и опровержениям, — поскольку метафизические теории не являются ни доказуемыми, ни опровержимыми. Таким образом, критерий демаркации догматического фальсификациониста оказывается в высшей степени антитеоретическим.

Знание и неявное знание

Майкл Полани считает, что научное знание можно передать через формальные языки только частично, а оставшаяся часть будет составлять личностное или *неявное знание* учёного, которое принципиально непередаваемо. Учёный, постепенно погружаясь в науку, принимает некоторые правила науки некритично. Эти некритично принятые и формально непередаваемые правила (часто включают навыки, умения и культуру) и составляют неявное знание. Ввиду того, что формализировать и передать неявное знание невозможно, невозможно и сравнение этого знания. Вследствие **чего в науке присутствует сравнение только формализованной части одной теории с формализированной частью другой теории**.

Гносеологический анархизм

Пауль Фейерабенд считает, что единственным принципом, не создающим препятствий прогрессу, является принцип «допустимо всё». Ни одна теория никогда не согласуется со всеми известными в своей области фактами. Любой факт *теоретически нагружен*, то есть зависит от теории, в рамках которой он рассматривается. Поэтому теорию нельзя сравнивать с фактами. Также теории нельзя сравнивать и друг с другом из-за того, что понятия в разных теориях имеют разное содержание.

Обоснование без применения научного метода

В истории науки есть многочисленные случаи, когда идеи, получившие впоследствии научное признание, изначально обосновывались или пояснялись без видимых рациональных оснований, не в соответствии с научным методом. Одним из наиболее ярких подобных примеров является обоснование Коперником гелиоцентрической системы. Первоначально новая теория, в которой планеты обращались вокруг Солнца строго по круговым орбитам, давала значительно больше расхождений с наблюдениями, чем господствовавшая до неё теория эпициклов Птолемея, то есть экспериментальная проверка говорила в пользу прежней теории, а не новой. Поэтому Коперник был вынужден апеллировать к простоте, внутренней красоте и гармоничности: «В центре всего, в покое, находится Солнце. В этом прекраснейшем храме кто может найти этому светильнику лучшее место, чем то, из которого он может освещать всё одновременно?»

Неспособность претендовать на абсолютную истинность

В богословии и в некоторых направлениях философии научное знание рассматривается как всегда ограниченное, условное и потому никогда не способное претендовать на абсолютную истинность. Это подтверждается процессом смены научных теорий, описанным выше. В то же время многие философские системы вообще выражают сомнения в существовании абсолютных истин, предлагая другие теории истины и знания, а успех науки в объяснении мира рассматривается большинством философов как признак её относительной истинности, что бы это ни обозначало.

1.2.4.7. Разработка процесса

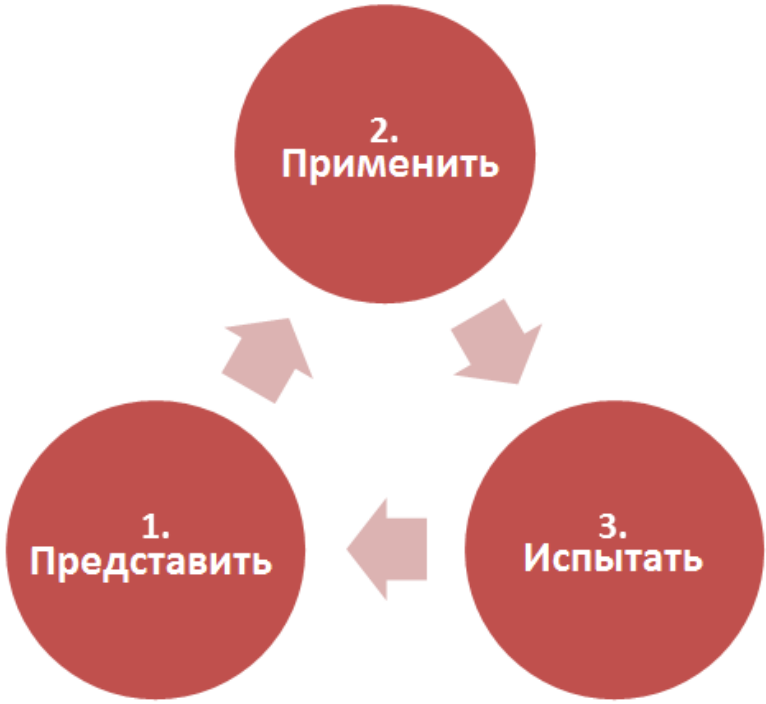
Принципы разрабтки процесса рассмотрим на примере процесса проектирования.

Проектирование будем рассматривать как применение практических и научных знаний в решении задач при использовании систематизированного процесса. Какой систематизированный подход применяют проектировщики для решения задачи?

Процесс проектирования представляет собой последовательность этапов, которой следуют проектировщики при поиске и реализации решений. Это и есть систематизированный подход. По сути это похоже на «научный метод», преподаваемый учащимся. Единого процесса проектирования, принятого во всем мире, не существует. Большинство проектировщиков используют собственные методики в процессе работы. **Процесс начинается с постановки задачи и завершается представлением готового решения, но промежуточные этапы могут быть различными.**

Процесс проектирования можно представить как рецепт бананового хлеба. Этот хлеб может быть изготовлен различными способами, но процесс его изготовления почти всегда начинается с бананов и завершается формой буханки. Один из таких «рецептов» будет описываться в настоящем блоке. Однако, это не единственно верный вариант организации процесса проектирования, а лишь один из примеров. На его примере учащиеся изучат основы процесса.

Процесс проектирования может быть представлен как замкнутый цикл из трех этапов:



В рамках этого простого цикла производится генерирование идеи (1). Ее применение (2). После внедрения идеи проектная группа производит испытания изделия или анализ результатов его внедрения путем тестирования (3). Как правило, в ходе испытаний или оценки результатов генерируются дополнительные идеи, и весь процесс повторяется. Повторяющаяся природа этого цикла позволяет утверждать, про процесс проектирования является итерационным процессом.

Итерация - это повторение действия снова и снова с целью усовершенствования процесса и последующего достижения поставленной цели.

Данный процесс мог бы продолжаться непрерывно (или до тех пор, пока проектная группа не перестанет придумывать новые идеи и искать недостатки в своем проекте). О проектировщиках говорят: «Иногда чтобы построить что либо, из процесса проектирования нужно исключить проектировщика!»

ПРИМЕНЕНИЕ ПРОЦЕССА ПРОЕКТИРОВАНИЯ:

Как обсуждалось ранее, не существует единственно верного процесса проектирования. В рамках настоящей работы, для моделирования концепции, проектирования и создания соревновательного робота учащиеся применят процесс проектирования, включающий 11 этапов:

Этап 1 - ПОНЯТЬ - Определить задачу

Этап 2 - ИССЛЕДОВАТЬ - Произвести исследования

Этап 3 – ОПРЕДЕЛИТЬ – Выделить спецификации решения

Этап 4 – ПРЕДСТАВИТЬ – Выработать решения

Этап 5 – ПРОТОТИП – Понять, работают ли концепции

Этап 6 – ВЫБРАТЬ – Выделить окончательную концепцию

Этап 7 – ДОРАБОТАТЬ – Детально проработать проект

Этап 8 – ПОКАЗАТЬ – Получить реакцию и утверждение

Этап 9 – ПРИМЕНИТЬ – Применить решение

Этап 10 – ИСПЫТАТЬ – Испытать решение на соответствие рабочих показателей

Этап 11 – ПОВТОРИТЬ

ЭТАП 1: ПОНЯТЬ

На этом этапе проектировщики определяют задачу, решением которой займутся.

Это единственный неизменный и наиболее важный этап процесса проектирования. Без понимания задачи, ее невозможно решить. Зачастую, данный этап реализуется не правильно или не полностью, что оказывает негативное воздействие на весь проект. Очень важно на данном этапе выделить задачу, а не только определить ее признаки или видимые стороны.

В ходе определения задачи необходимо помнить историю о лифте.

Иллюстрацией может служить история о небоскребе, усовершенствование которого поручили молодым инженерам. Небоскреб располагался в деловом сердце города. Его арендаторы постоянно жаловались на то, что подъем на лифте занимает слишком много времени. Владельцы здания решили исправить ситуацию и предложили различным проектным бюро представить свое видение решения проблемы.

1. Одна компания предложила добавить несколько новых лифтов. Аргументация состояла в том, что за счет добавления дополнительных лифтов длительность и количество остановок каждого из них сократится, что приведет к сокращению общего времени в пути. Стоимость проекта была абсурдной (детализация расчетов была произведена на основании устных суждений).

2. Другая компания предложила обновить здание и установить в здании несколько совершенно новых высокоскоростных лифтов, соответствующих последнему слову техники. За счет использования этих лифтов время в пути также могло бы сократиться. Расчет данного проекта был более разумным, но стоимость все равно превышала все разумные пределы.

3. Третья компания предложила обновить программное обеспечение лифта. Проектировщики предложили разработать новый алгоритм работы существующих лифтов и с его помощью повысить эффективность их работы и сократить среднее время в пути. Это

предложение было дорогостоящим, но его стоимость была в разы ниже предыдущих предложений.

Владельцы здания были почти готовы нанять третью фирму, когда поступило четвертое предложение. После детального рассмотрения, оно было незамедлительно принято. Четвертая проектная организация предложила установить в каждом лифте зеркала в полный рост. Перемещаясь в лифте, люди поправляли галстуки и проверяли макияж, не замечая времени. Реализация этого проекта почти ничего не стоила владельцам, а сам проект имел огромный успех. Проектировщики четвертой компании сфокусировали внимание на действительной проблеме, которая состояла не в слишком медленном перемещении самого лифта, а в восприятии людей.

Этап «ПОНЯТЬ» в соревновательной робототехнике

В соревновательной робототехнике существует огромного количество задач, решение которых ложится на проектную команду. Чем больше будущие проектировщики работают над проектом, тем больше новых задач возникает. Это связано с тем, что одна комплексная задача, как правило, состоит из более мелких простых задач. На ранних этапах проектирования задачи представляют собой «общие картинки», тогда как на поздних этапах они более «ориентированы на детали».

Вот некоторые простые задачи, которые проектировщик может счесть требующими решения, а также сопряженные с ними вопросы.

Какая игровая стратегия является наиболее эффективной? Как победить в матче?

Как сделать так, чтобы робот набрал наибольшее количество очков в ходе матча? Как набрать больше очков, чем противники?

Насколько быстро должен перемещаться робот?

Какими способами робот может поднимать игровые объекты?

Как сделать так, чтобы роботы мог быстро поднимать объекты?

Сколько игровых объектов должен удерживать робот?

Все эти задачи и вопросы имеют массу решений и ответов, некоторые из которых лучше других. Как проектировщик понимает, что найдено правильное решение и правильный ответ? Вот здесь в работу вступает остальная часть процесса. Но пока задача не определена, она не может быть решена!

ЭТАП 2: ИССЛЕДОВАТЬ

На этом этапе проектировщики исследуют задачу, над которой работают. Они рассматривают способы решения аналогичных задач, разработанные другими специалистами. Проектировщики также собирают информацию о внешней среде, с которой им придется иметь дело, ситуациях, в которых будет применяться их решение, а также путях его использования.

Этап «ИССЛЕДОВАТЬ» в соревновательной робототехнике

Учащиеся, вовлеченные в изучение соревновательной робототехники, должны также исследовать свою задачу. Важно детально рассмотреть аналоги поставленной игровой задачи, существующие в реальном мире. Учащиеся могут также изучить опыт предыдущих соревнований на предмет решения аналогичных игровых задач. Настоящий раздел посвящен аккумулированию данных из разных источников для обеспечения учащихся, проектирующих роботов, дополнительными ресурсами в ходе поиска успешного решения.

ЭТАП 3: ОПРЕДЕЛИТЬ

На этом этапе проектировщики выделяют решение, над которым продолжат работу, не приступая пока к его реализации. Выбор делается на основе спецификаций.

Что такое спецификация? **Спецификация - это детализированный набор требований, которым должны соответствовать материал, изделие или услуга.** В этом случае, спецификации являются требованиями, применяемыми к решению задачи, определенной в рамках 1-го этапа процесса проектирования.

Спецификации, как правило, могут иметь два источника:

1. Проектные ограничения

2. Функциональные требования

Что такое ограничение? **Ограничение - это условие, которому должно соответствовать решение задачи**. По сути своей, ограничения являются запретами. Что такое функциональное требование? **С помощью функциональных требований составляется описание того, как должно работать завершенное решение.**

Итак, спецификациями определяется, ЧТО будет реализовано с помощью решения и насколько хорошо это будет реализовано, но не КАК это будет реализовано. В соревновательной робототехнике, спецификацией определяется, ЧТО делает робот, а не КАК он это делает. На этом этапе не рекомендуется уделять слишком много внимания поиску ответа на вопрос «КАК», поскольку это может оказаться контрпродуктивным. В то же время, проектировщики должны всегда помнить о вопросе «КАК», поскольку им требуется базовое понимание возможностей для реализации. (Например, утверждать, что ноутбук сможет непрерывно работать в течение одного года на одной батарее АА, не разумно).

Этап «ОПРЕДЕЛИТЬ» в соревновательной робототехнике

В соревновательной робототехнике, проектировщикам предлагается принять участие в испытаниях или играх с созданными ими роботами. К испытанию зачастую прилагается руководство с описанием ряда ограничений и требований, которым должен соответствовать каждый робот. Это проектные ограничения. И это первый тип спецификации, которую проектировщик должен учитывать в работе. Примером данного типа являются «максимальный стартовый размер робота» и «максимальное допустимое количество электромоторов».

Некоторые спецификации могут быть спровоцированы доступными ресурсами. Первый из двух перечисленных выше наборов правил входит также в правила соревнований и известен всем проектировщикам. Второй набор ограничений не всегда очевиден, но не менее важен в рамках процесса проектирования. Некоторые из них могут являться самостоятельно назначаемыми спецификациями ограничивающего типа. Вот два примера: «проект робота должен

соответствовать выделенному бюджету» и «в конструкции робота могут быть использованы только имеющиеся у проектировщика части».

Еще одно самостоятельно назначаемое ограничение касается возможностей команды.

Одним из важнейших элементов успешного формирования проектных ограничений в соревновательной робототехнике является понимание возможностей. Желая создать исключительный проект, многие команды пытаются выйти за пределы своих возможностей.

Каждая команда должна четко осознавать собственные возможности. Возможности зачастую зависят от количества людей, ресурсов, бюджета, опыта и многого другого. В определении достижимого проекта необходимо сфокусировать внимание на его общем виде. Если проект разделен на части, каждая из них может быть доступной, но при суммировании проект окажется не по силам.

Команды могут добиться большего успеха, выбрав простой проект и полностью реализовав его, чем путем выбора сложного проекта при слабой реализации! Например, представьте себе, что две команды собирают роботов, которые должны уметь забивать футбольный мяч в ворота. Одна команда принимает решение о сборке простого плуга, который будет заталкивать мяч в ворота. Другая команда пытается собрать механизм, который будет пинать мяч, чтобы забить его в ворота. Этот механизм кажется более удачной идеей, но что если вторая команда не сможет реализовать свою задачу? В этом случае, победит более простое решение! Вторая команда должна соотнести свои возможности со сложностью задачи.

Следующая группа спецификаций основана на функциональных требованиях, предъявляемых к роботу. Сюда входят элементы, которые, по задумке проектировщика, будет выполнять робот. Некоторые из них относятся к испытанию, с которым столкнется проектировщик (например, робот может удерживать 10 игровых объектов, робот может поднимать игровые объекты вверх на 1 метр, и т.д.)

Разработка третьего типа спецификаций на ранних этапах процесса проектирования может оказаться ложной или невыполнимой задачей,

так как большая часть этих спецификаций зависят от характера проекта и его развития. Они больше относятся к подсистемам, нежели ко всему проекту.

Рейтинг спецификаций

Спецификации могут обладать разной значимостью в рамках проекта. Проектировщикам необходимо обдумать, что является наиболее важным и почему. Спецификации зачастую упорядочивают в зависимости от степени значимости. Вот пример подобного рейтинга:

W = Возможно (не так важно, но было бы полезно реализовать)

P = Предпочтительно (важно, но без этого проект может быть реализован)

D = Необходимо (критически важно для проекта, ДОЖНО быть включено в проект)

Путем расстановки этих оценок проектировщик может распределить спецификации по приоритетам. Это также послужит для проектировщика хорошим инструментом «контрольной проверки» после завершения проекта. Проектировщик может пересмотреть перечень спецификаций и проанализировать, насколько изначальный рейтинг был реализован в проекте.

На основании рейтинга проектировщик также принимает решение относительно того, что является наиболее важным. Рейтинг спецификаций, в данном случае, позволяет более четко организовать идеи, что дает возможность понять основную цель, на которой необходимо сфокусироваться. В некоторых случаях создание рейтинга представляет собой простую задачу. Например, когда к проекту применяется ОБЯЗАТЕЛЬНОЕ ограничение. Данное ограничение автоматически расценивается как «необходимое».

При создании спецификаций некоторые проектировщики размещают несколько аналогичных спецификаций на разных уровнях в рейтинге, чтобы показать разную степень их значимости. Ниже приводится пример:

Робот может удерживать 5 игровых объектов - необходимо.

Робот может удерживать 10 игровых объектов - предпочтительно.

Робот может удерживать 15 игровых объектов - возможно.

Из примера видно, что робот ДОЛЖЕН уметь удерживать 5 игровых объектов. В случае возможности реализации, он будет способен удерживать и 10. И идеальный вариант, если он сможет удерживать 15 игровых объектов. Используя правильные спецификации и расставляя их в соответствии с приоритетами, проектировщики выделяют наиболее важные требования, обязательные к соблюдению, а также цели, к достижению которых должна стремиться вся команда.

ЭТАП 4: ПРЕДСТАВИТЬ

Сформулировать, изобразить или сформулировать идею.

Теперь, когда проектировщик знает, ЧТО будет реализовано с помощью решения, он или она должны определить, КАК это будет реализовано.

Два слова: «салфеточные наброски» Эта фраза означает привычку всегда фиксировать возникающие идеи вне зависимости от места и времени. Даже если приходится рисовать ручкой на салфетке - это тоже хороший вариант.

В случае возникновения проблемы или при необходимости принятия решения, все люди делают одно и то же: обдумывают альтернативные варианты действий, даже если это происходит спонтанно. Формально, документирование этого интуитивного действия может помочь в решении сложных проектных задач.

На этом этапе также необходимо проявить изобретательность. Вот некоторые из вопросов, часто задаваемых проектировщикам: «Как вы пришли к этому?», «Откуда вы почерпнули идею?» Идея может возникнуть где угодно! Вдохновение может прийти из ниоткуда!

Ключевые слова: воображение и мысль. Для обеспечения соответствия спецификации проектировщик должен найти множество

путей решения задачи. При этом очень важно помнить, что идея может возникнуть случайно. **Мантра проектировщика - «возьми лучшее, затем изобрети остальное». Хороший проектировщик внимательно изучает окружающий мир в поиске решений, которые он сможет адаптировать и применить для решения поставленной задачи.** Инновация также очень важно на ранних этапах процесса проектирования (в первую очередь, внедряйте новое). Между использованием ранее созданных проектов и внедрением новых должен быть найден баланс.

Часто соединение двух идей или поиск компромисса между двумя различными предложениями может привести к созданию качественной концепции. **Усовершенствование и инновация на ранних этапах проектирования могут способствовать получению отличных результатов на последующих этапах работы.**

Очень важно не ограничивать и не усреднять концепции и всегда стремиться найти «правильное» решение. Правильное решение, зачастую, приходит само. В этом случае проектировщики говорят «я чувствую, что это верное решение». Правильное решение выглядит органичным и простым. К сожалению, оно не всегда легко достижимо, а его органичность не всегда очевидна.

Проектировщики должны фиксировать ВСЕ свои идеи в проектном отчете!

(Для проектировщика важным этапом является копирование салфеточных набросков в проектные отчеты для последующего использования в проекте.)

Этап «ПРЕДСТАВИТЬ» в соревновательной робототехнике

В соревновательной робототехнике, **для достижения хороших результатов необходимо разработать несколько концепций**. **Командам необходимо разработать концептуальные стратегии, концепцию целой системы, а также концепции отдельных подсистем и механизмов**. Некоторые системы являются взаимозависимыми и влияют на работу друг друга. Стратегия команды влияет на проект целой системы, что, в свою очередь, влияет на различные подсистемы, но каждая из подсистем оказывает влияние на работу целой системы.

Разработка этих концепций обычно осуществляется на встречах команды, при этом в процессе разработки участвуют все участники встречи. **Концепции фиксируются в проектных отчетах в виде схем, схематических изображений или описаний.**

Коллективный поиск решений – Техника, с помощью которой команда проявляет свою изобретательность

На этом этапе процесса проектирования требуется изобретательность и создание большого количества вариантов решения задачи. Для этого проектировщики используют инструмент, который называется **«коллективный поиск решений»**. Это упражнение, в рамках выполнения которого отдельные члены команды работают совместно с целью создания большого количества идей.

Вот несколько важных правил, касающихся коллективного поиска решений:

1. При коллективном поиске решений команды фокусируют внимание на количестве идей, а не на качестве каждой из них. Акцент делается на то, что среди множества найденных идей обязательно будут несколько гениальных!

2. Сохранение собственного мнения. Плохих идей не бывает, так как даже самая странная концепция может вдохновить проектировщика на создание исключительного решения. Сумасшедшие идеи можно усовершенствовать и проработать до состояния осуществимых идей.

3. Регистрация всех идей и мыслей. Учащиеся-проектировщики должны фиксировать все идеи, найденные в процессе коллективного поиска решений, в проектных отчетах.

ЭТАП 5: ПРОТОТИП

На данном этапе процесса проектировщики выделяют некоторые концепции, полученные в ходе предыдущего этапа, и создают на их основе версии решений. Главная цель данного этапа заключается в **изучении функциональности каждого решения, созданного на базе каждой отдельной концепции, а также формы его взаимодействия с окружающей средой**. **На данном этапе проектировщики также**

приступают к определению наиболее эффективной концепции. Создаваемые прототипы могут быть не проработаны, но должны быть достаточно функциональны для того, чтобы на их основе можно было получить представление о концепции. Ключевое слово: изучить.

Проектировщики не обязаны создавать прототипы для всех без исключения концепций, только для тех, которые они планируют реализовать в дальнейшем!

Этап «ПРОТОТИП» в соревновательной робототехнике

В соревновательной робототехнике, роботы взаимодействуют с окружающей их средой, поэтому для достижения успеха проектировщики должны изучить природу этого взаимодействия. Для этого проектировщики проводят испытания в «реальных условиях», чтобы получить представление о формах взаимодействия и внести первые корректировки уже на РАННИХ этапах проектирования. Система проектирования VEX Robotics Design System идеально подходит для быстрого создания прототипов, необходимых проектировщикам в процессе работы. Система предназначена для быстрой сборки конструкций и механизмов, проведения испытаний их работы, а также внесения незначительных изменений, без необходимости наличия производственной базы.

При создании прототипов команды должны проявлять максимум внимания. В ходе работы учащиеся должны вносить подробные примечания в проектные отчеты, записывая все, что видят, а также мысли относительно того, почему одни элементы эффективнее других, затем создавая дополнительные прототипы, чтобы проверить свои идеи. **Сбор данных является важной частью работы с прототипами**.

ЭТАП 6: ВЫБРАТЬ

На этом этапе проектировщик или проектная группа выделяет **несколько потенциальных решений задачи**. Здесь проектировщики будут использовать для определения наиболее эффективной концепции уроки, полученные в процессе работы с прототипами. Выбор не всегда является простым и очевидным. Иногда правильное решение приходит само. Зачастую определить лучшее бывает трудно. Команды сравнивают решения на соответствие спецификациям, установленным

на этапе 3, и пытаются определить одно, явно выделяющееся среди других. Проектировщики ищут наиболее простое и органичное решение.

В том случае, если выделить одно решение сложно, для принятия решения проектировщики применяют систематизированный подход.

При выборе концепций в рамках проектной группы, голосование кажется наиболее удобной методикой. Тем не менее, голосование базируется на необоснованном суждении, а необоснованные суждения не приемлемы в проектировании. Когда речь идет об обсуждении проекта, важно проанализировать суть вещей и вынести общее логическое решение. Как обсуждалось выше, важно использовать количественные доказательства аргументации. Например, утверждения «это решение лучше» не достаточно, а утверждение «на 14,8 % светлее» может служить доказательством того, почему «это решение лучше».

В некоторых случаях в процессе принятия решений участвует не вся группа, а только ее руководящая часть, либо только руководитель. В этой ситуации руководство группы несет ответственность за сравнение альтернатив и выбор плана действий. Этот метод не всегда работает хорошо, особенно если оставшаяся часть группы не принимает авторитет лидера и ставит под вопрос принимаемые им решения. Тем не менее, этот метод может быть полезен в предотвращении тупиковых ситуаций, когда консенсус не может быть достигнут. Для получения одобрения от группы руководитель должен постараться использовать метод принятия согласованного решения.

Этап «ВЫБРАТЬ» в соревновательной робототехнике

В соревновательной робототехнике, задачи, с которыми сталкиваются учащиеся, аналогичны реальным задачам, стоящим перед профессиональными проектировщиками. Учащиеся должны работать сообща, чтобы выделить концепцию, максимально соответствующую спецификациям проекта, то есть найти ответ на вопрос «Какая концепция наиболее эффективна в решении задачи?» Учащиеся должны использовать свои прототипы как основную для принятия решения. Очень важно применять количественные доказательства для аргументации преимуществ одной концепции относительно другой. Проще всего это можно сделать с помощью прототипа.

Инструмент принятия решений: Таблицы взвешенных обоснований

В процессе проектирования, на этапе выбора концепции используется инструмент, который называется **«таблица взвешенных обоснований» (WOT), или матрица решений.** Таблица взвешенных обоснований может использоваться как вспомогательное средство в осуществлении выбора между вариантами на основании рейтинга критериев. WOT является особенно эффективным инструментом, так как с его помощью проектировщики могут сравнить альтернативы на основании параметров, имеющих наибольшую значимость для финального решения.

Подсистема 1 – Сравнение ходовых частей

Критерии сравнения	Вес	Скользящий ход		Маневренный ход		Бортовой поворот	
		Балл	Взвешенный балл	Балл	Взвешенный балл	Балл	Взвешенный балл
Стоимость	5	6	30	3	15	9	45
Маневренность	15	10	150	10	150	4	60
Вес	10	5	50	2	20	8	80
Использование мотора	20	5	100	2	40	8	160
Всего:	50		330		225		345

Подсистема 2 – Сравнение захватных устройств

Критерии сравнения	Вес	Зажимные клещи		Роликовые клещи		Ковш	
		Балл	Взвешенный балл	Балл	Взвешенный балл	Балл	Взвешенный балл
Стоимость	5	5	25	5	25	9	45
Скорость захвата	20	7	140	9	180	2	40
Сила сжатия	15	10	150	10	150	1	15
Вес	10	6	60	4	40	8	80
Всего:	50		375		395		180

Подсистема 3 – Сравнение подъемных механизмов

Критерии сравнения	Вес	Лифт		Рука с 2 подвижными соединениями		Рычажный механизм	
		Балл	Взвешенный балл	Балл	Взвешенный балл	Балл	Взвешенный балл
Стоимость	5	3	30	7	15	9	45
Сложность	20	2	150	7	150	9	60
Вертикальное расстояние	15	8	50	7	20	4	80
Компоновка	10	9	100	8	40	3	160
Всего:	50		265		360		315

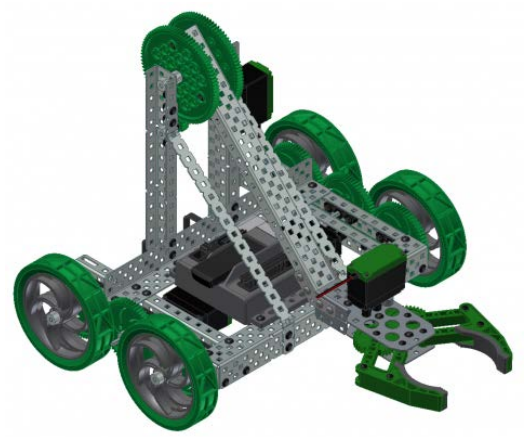
ЭТАП 7: ДОРАБОТАТЬ

На этом этапе проектировщики приступают к реализации выбранной концепции. Этот этап касается деталей. По его завершении проектные группы должны получить все необходимое для конструирования и внедрения проектного решения. Вот некоторые элементы, которые могут быть разработаны на данном этапе: модели САПР, сборочные чертежи, планы изготовления, ведомость материалов, руководства по установке, пользовательские руководства, презентации проектов, предложения и многое другое.

Работа начинается с базового проекта, который постепенно расширяется за счет добавления деталей. На практике, работа над проектом не начинается с детального представления всех его частей с последующим их сопряжением. Финальный проект является результатом постепенной проработки и детализации базовых элементов проекта.

Этап «ДОРАБОТАТЬ» в соревновательной робототехнике

В соревновательной робототехнике, создание трехмерной модели робота в Autodesk Inventor является важным этапом. Это, возможно, один из самых длительных этапов процесса проектирования, но за счет него выигрывает весь проект. Чем лучше проработан проект робота, тем проще будет справиться с его сборкой.



Самое главное в любом проекте - это детали.

Детальная проработка трехмерных моделей позволяет решать возникающие проблемы проекта до того, как они станут критическими.

На этом этапе процесса также производятся расчеты. В их числе расчеты, касающиеся оптимизации передаточного отношения, прочности материалов, массы, стоимости и многого другого. В процессе проектирования не всегда есть возможность полностью оптимизировать все аспекты проекта. В зависимости от проекта, иногда максимальной «приближенности» к цели уже достаточно. При планировании, многие проектировщики могут брать за основу только собственный опыт и интуицию, не уделяя внимания выполнению расчетов для каждой детали. Такой подход может работать в робототехническом проекте в рамках высшей школы, но не может применяться в проектировании частей костюма для космонавтов, где оптимизация очень важна. Вместо того, чтобы оптимизировать каждый элемент, убедитесь, что он способен выполнять свою задачу. Этого достаточно для того, чтобы данный элемент мог быть применен при сборке, так как все технические требования выполнены.

ЭТАП 8: ПРЕДСТАВИТЬ

Перед внедрением, детализированный проект зачастую проходит этап проектной экспертизы или утверждения. Проектная экспертиза может быть произведена различными методами. Некоторые виды экспертизы могут быть реализованы в форме беседы двух проектировщиков. Другие виды экспертизы производятся в форме собрания проектной группы, где члены группы проверяют проделанную работу на наличие ошибок. Многие виды экспертизы включают презентацию детализированного проекта перед заказчиком, менеджером или ответственным лицом, принимающим решения, с целью получения окончательного утверждения.

Этап «ПРЕДСТАВИТЬ» в соревновательной робототехнике

В соревновательной робототехнике, проектировщик робота или проектная группа представляют окончательный проект робота перед членами группы, классом, школой или руководством для получения утверждения. Иногда команда организует встречу, целью которой

является проведение экспертизы робота. Во встрече могут принимать участие спонсоры, администрация и члены сообщества.

Презентация проекта является важной часть процесса проектирования. Многие проектировщики считают, что изучение языковых дисциплин не является важным этапом в их образовании, что плохое произношение и слабое знание грамматики, а также отсутствие коммуникативных навыков, не могут повлиять на их профессиональную деятельность. Это не так. Если проектировщик не может выразить возникшую идею, эта идея не сможет быть использована для решения задачи. Способность резюмировать, представлять и защищать идеи - исключительно важный навык! Это относится к устному описанию, письменному отчету, презентационным слайдам, проектным чертежам или другим типам презентационных материалов.

Целью проектной экспертизы является не просто утверждение проекта, но также поиск проблемных аспектов проекта а также потенциальных путей усовершенствования проекта. В процессе проектирования было разработано несколько альтернативных концепций, одна из которых была выбрана. В процессе проектирования подобный выбор - обычное дело. Обоснование выбора является одним из ключевых этапов презентации проекта. *«ПОЧЕМУ выбор был сделан в пользу ЭТОГО, а не ДРУГОГО?»* Экспертная группа должна убедиться, что проектировщики провели комплексный анализ, то есть проработали все альтернативы. Проект должен быть продуман, обоснован. Нельзя просто взять первую попавшуюся идею, пришедшую кому-либо в голову, и высказать ее.

Проектировщики должны вносить в проектные отчеты подробные примечания для последующего использования в ходе проектной экспертизы, в том числе перечень последующих действий.

Общие вопросы для проектной экспертизы:

- Почему это реализовано именно так?
- Вы рассматривали другие способы реализации?
- Почему были отвергнуты другие альтернативы?
- Данный способ реализации соответствует всем требованиям и спецификациям?
- Как может быть улучшено функционирование?

- Как может быть снижена масса?
- Как может быть увеличена скорость?
- Как может быть повышена прочность?
- Как может быть сокращен размер?
- Как может быть упрощена реализация?
- Как может быть повышена производительность?
- Как может быть снижена стоимость?
- Как может быть упрощен процесс конструирования?
- Какие еще функциональные возможности могут быть добавлены?

Анализ эффективности затрат

Иногда важной частью экспертизы проекта является проведение анализа эффективности затрат. При выполнении этого вида анализа, проектировщики рассматривают проект в двух аспектах: затраты и эффективность, реализуемая за счет затрат. Проектировщик сопоставляет эти два аспекта.

При этом затраты не всегда выражены в финансовом эквиваленте. Затраты могут касаться ресурсов, необходимых для реализации, времени, количества персонала, денежных средств, свободного пространства на роботе, его массы, а также многого другого. Они также могут иметь отношение к обоснованному отказу от чего-либо с целью получения требуемой функциональности. Иными словами, если мы сконструируем руку с двумя подвижными соединениями, места для размещения приемника мячей на роботе уже не останется.

Функции, обеспечивающие высокую эффективность при низких затратах, являются наиболее выгодными и должны быть добавлены к проекту. Очень важно проводить подобный анализ на всех этапах процесса. Даже не значительные дополнения могут иметь огромное значение в последующем. Элементы, требующие значительных затрат, могут быть внедрены только в том случае, если они обладают исключительной эффективностью. Проектировщики всегда должны учитывать в работе эти важные факторы.

ЭТАП 9: ПРИМЕНИТЬ

После завершения процесса проектирования и получения утверждения проект должен быть применен. В зависимости от природы задачи, решения могут различаться. В зависимости от типа решения, его применение может также быть различным. Применение может заключаться во внедрении нового процесса, созданного проектировщиками, либо может состоять в плане производства и изготовлении некоторого физического объекта. Например, в истории о лифте, которая обсуждалась ранее, было представлено несколько уникальных решений.

Если проектировщик хочет найти способ ускорения процесса завязывания шнурков, он создает процесс. Применение решения заключается в том, чтобы представить людям новый способ ускорения процедуры завязывания шнурков. Если проектировщик хочет усовершенствовать ботинок, применение решения будет состоять в производстве и распространении новых ботинок. Области применения решений могут иметь различные формы.

Этап «ПРИМЕНИТЬ» в соревновательной робототехнике

В соревновательной робототехнике, на этом этапе учащиеся выполняют сборку. Все детали, выполненные в рамках этапа 7, используются здесь для создания полноценного функционального соревновательного робота (или подсистемы, являющейся часть большого законченного изделия!) На данном этапе может производиться закупка компонентов, обрезка частей, сборка, а также другие действия, целью которых является получение конечного изделия.

Здесь команда изготавливает пакеты для продажи, заявки на участие в категориях и другие материалы, связанные с соревнованиями, но не имеющие отношения к самому роботу. Эти элементы являются также частью применения проекта.

ЭТАП 10: ИСПЫТАТЬ

На данном этапе проектировщики проводят испытания примененного решения, чтобы оценить качество его работы. Здесь производится

анализ того, что работает и что не работает, а также что должно быть усовершенствовано. Порядок проведения испытаний и их результаты документируются. Основной задачей данного этапа является подтверждение надлежащего функционирования примененного решения и его соответствие спецификациям.

Что происходит, если проект не признан удовлетворительным? Проектная группа должна сделать его удовлетворительным! Проектная группа должна разработать план усовершенствования решения. Этот план может заключаться в повторной разработке с нуля, в возврате к чертежной доске и создании нового плана.

После применения решения, его анализа и признания проекта удовлетворительным, процесс проектирования может считаться завершенным.

Этап «ИСПЫТАТЬ» в соревновательной робототехнике

В соревновательной робототехнике, испытания могут проводиться также в ходе соревнований.

1.2.4.8. Способ как объект облачных технологий

Способ как объект облачных технологий — это способ, обладающий существенными отличиями и дающий при использовании положительный эффект процесс выполнения взаимосвязанных действий, необходимых для достижения поставленной цели.
Способ отличается от принципа действия машины или механизма тем, что в нем отсутствует причинно-следственная связь между операциями 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.**
К способам относятся:
а) способы добычи, заготовки и получения сырья и материалов;
б) **технологические процессы как совокупность действий, направленных на материальные объекты с целью их полезного преобразования** — процессы обработки и переработки сырья и полуфабрикатов в готовые продукты и изделия;

в) способы предохранения готовых веществ от вредных влияний, обеспечения их сохранности, а также маркировки, расфасовки, укладки, дозировки, упаковки продуктов и изделий;
г) способы измерения, испытания и контроля готовности, надежности, соответствия заданным параметрам искусственно созданных или существующих в природе объектов или явлений;
д) способы монтажа, сборки и установки изделий, оборудования, сооружений;
е) способы наладки, настройки, ухода, управления и регулирования, предупреждения аварийных ситуаций, обеспечивающие нормальное функционирование приборов, машин, агрегатов, поточных линий;
ж) способы уничтожения и переработки производственных или иных отходов, очистки, охраны внешней среды от загрязнений и т. п.;
з) способы воздействия на естественные природные процессы и явления с целью придания им полезного направления — способы закрепления сыпучих песков; стимулирования роста растений и животных; селекции и гибридизации и т. п.;
и) способы профилактики, диагностики и лечения заболеваний людей и животных.

Способ, так же как и устройство, характеризуется присущими только ему признаками. При характеристике I способа нельзя использовать признаки других видов объектов.

Способ характеризуется следующими признаками:

I. Приемами, операциями, т. е. различными целенаправленными действиями, совершаемыми для достижения определенной цели. Совокупность и последовательность операций составляет законченный технологический процесс.

II. Последовательностью операций. Последовательность операций в технологическом процессе может быть различной. Часто однородные технологические процессы включают одинаковые операции, но различаются их последовательностью.

III. Режимом проведения операции, т. е. конкретными параметрами (температурой, давлением, концентрацией, временем — в химии; усилием резания, стойкостью резца — в металлообработке и т. п.), которыми характеризуется операция.

IV. Соотношением материалов, используемых при проведении процессов.

V. Использованием элементов (аппаратов, механизмов, машин, контрольно-измерительных приборов и т. п.) и материалов (сырья, полупродуктов, катализаторов) при проведении процесса.

Для характеристики способов, так же как и для характеристики устройств, наиболее важной группой является первая. Сами операции и приемы и составляют технологический процесс, без совокупности приемов не может быть и признаков других групп (последовательность, режим и т. п.). Наличие в объекте новых операций обычно обусловливает для него наличие существенных отличий, так как наличие новой операции, как правило, сообщает объекту новые свойства, обеспечивающие положительный эффект.
Признаки I группы вводятся в определения понятия с помощью наиболее употребительных, устоявшихся названий операций (приемов). Обычно это глаголы действительного залога изъявительного наклонения в третьем лице и множественном числе, например: нагревают, прессуют, добавляют, протягивают, разбавляют, перемешивают, окисляют, пробивают и т. д.
Признаки II группы — последовательность операций — тесно связаны с признаками I группы, так как всегда операции осуществляются в определенной последовательности. Сама по себе последовательность операций довольно редко является новым элементом, поскольку если имеется какой-то установившийся технологический процесс, то в нем чаще меняются операции или режимы их проведения и значительно реже меняется их последовательность. Зато введение в технологический процесс новой операции всегда сопровождается введением признака последовательности. Отсюда в определении понятия появляются такие выражения: «перед штамповкой нагревают», «после добавления серной кислоты перемешивают» и т. д.
Признаки II группы обычно вводятся словами: перед, после, до, вслед, одновременно, последовательно, параллельно, затем и пр.
Признаками III группы являются режимы проведения операций, которые конкретизируют признаки I и II групп. Режимом проведения операции называется конкретная форма ее осуществления. Например, форма проведения операции нагревания предусматривает определенный интервал температур, при котором проводится эта операция, и времени, в течение которого эта температура поддерживается. Режим проведения операции прессования — значения прикладываемого усилия, температуры, при которой ведется прессование, и т. д.
Известные признаки III группы редко указываются в определении понятия, а новые признаки могут указываться в отличительных частях определений понятий..
Признаки III группы они очень важны для характеристики способа, поэтому их необходимо указывать при формулировке определения

понятия. По своей значимости признаки III группы стоят непосредственно после признаков I группы.
Признаки III группы вводятся обязательно в определении понятия с указанием интервала, например: при температуре от 20 до 50° С, при давлении от 1,5 до 2 *атм* и т. д.
Признаки IV группы — указание на соотношение материалов и веществ, применяемых в способе, — обычно используются при характеристике способа проведения химического процесса или способа получения нового вещества.
К V группе относят признаки, характеризующие использование в процессе различных элементов (и материалов). Признаки, характеризующие элементы, как самостоятельные, используются только в дополнительных пунктах формулы, а признаки, характеризующие материал, — в главных. Признак, характеризующий элемент, имеет подчиненное значение по отношению к той операции, для которой элемент (аппарат, приспособление, прибор и т. п.) применен, и служит в основном для характеристики этой операции, например: фильтруют в вакуум-фильтре, промывают в реакторе с рамной мешалкой и т. д. Само по себе использование нового (т. е. не применявшегося ранее в этом технологическом процессе) элемента аппарата, приспособления, прибора, как правило, является очевидным решением, не требующим творческого мышления (выбор целесообразного оборудования—обязанность проектанта).

1.2.4.9. Алгоритм как объект облачных технологий

1. Понятие алгоритма. Сущность алгоритмизации

Понятие алгоритма является фундаментальной категорией математики и не может быть выражено через другие, более простые понятия, а рассматривается как нечто неопределяемое. Другими словами, единого определения алгоритма не существует, есть только разные подходы, описания этого понятия, причем, в полном соответствии с той областью знаний, где он применяется. В рамках настоящей работы не предусмотрено углубление в теорию алгоритмов. Будем рассматривать понятие алгоритма и сущность процесса алгоритмизации в

приложении к решению некоторых вычислительных задач. Опишем понятие алгоритма, например, так:

Алгоритм *- это строгая, четкая последовательность математических и логических операций, приводящая к решению задачи.*

В Толковом словаре по информатике (1991г.) дано общепринятое понятие: *алгоритм* - точное предписание, определяющее вычислительный процесс, ведущий от варьируемых начальных данных к искомому результату.

Алгоритмизация процессов в широком смысле - это описание процессов на языке математических символов для получения алгоритма, отображающего элементарные акты процесса, их последовательность и взаимосвязь. Для построения алгоритма управления, например, необходимо к алгоритму, описывающему процесс функционирования системы, присоединить алгоритм определения оптимального решения или оптимальных значений параметров управления. В более узком смысле ***алгоритмизация*** - это процедура поиска, разработки и описания алгоритма решения задачи.

2. Свойства алгоритма

Описание основных свойств помогает углубить само понятие алгоритма. Итак, алгоритм должен обладать следующими свойствами:

- ***Детерминированность*** (*определенность, точность, однозначность*). Это свойство заключается в том, что при задании одних и тех же исходных данных несколько раз алгоритм будет выполняться абсолютно одинаково и всегда будет получен один и тот же результат. Свойство детерминированности проявляется также и в том, что на каждом шаге выполнения алгоритма всегда точно известно, что делать дальше, а каждое действие однозначно понятно исполнителю и не может быть истолковано неопределенно. Благодаря этому свойству выполнение алгоритма носит механический характер.
- ***Массовость*** - выражается в том, что с помощью алгоритма можно решать не одну конкретную задачу, а любую задачу из

некоторого класса однотипных задач при всех допустимых значениях исходных данных.

- ***Результативность*** (*направленность*) - означает, что выполнение алгоритма обязательно должно привести к решению поставленной задачи, либо к сообщению о том, что при заданных исходных величинах задачу решить невозможно. Алгоритмический процесс не может обрываться безрезультатно.
- ***Дискретность*** - означает, что алгоритм состоит из последовательности отдельных шагов - элементарных действий, выполнение которых не представляет сложности. Именно благодаря этому свойству алгоритм может быть реализован на ЭВМ.
- ***Конечность*** (*финитность*)- заключается в том, что последовательность элементарных действий алгоритма не может быть бесконечной, неограниченной, хотя может быть очень большой (если требуется, например, большая точность вычислений).
- ***Корректность*** - означает, что если алгоритм создан для решения определенной задачи, то для всех исходных данных он должен всегда давать правильный результат и ни для каких исходных данных не будет получен неправильный результат. Если хотя бы один из полученных результатов противоречит хотя бы одному из ранее установленных и получивших признание фактов, алгоритм нельзя признать корректным.

Если разработанная Вами последовательность действий не обладает хотя бы одним из перечисленных выше свойств, то она не может считаться алгоритмом

3. Средства описания алгоритма

Описание алгоритма вполне допустимо ***на естественном языке***, таком как русский, французский, английский, немецкий и др. Первое время это устраивало математиков. Однако постепенно выяснилось, что применение естественных языков в точных науках связано с рядом трудностей и даже может приводить к противоречиям. **В естественных языках не всегда форме конкретного предложения соответствует единственное содержание.** Предложения могут иметь расплывчатый смысл, требовать знания ситуации, контекста. Отдельные слова многозначны. Так, для обеспечения нужд науки стали

возникать ***формальные подъязыки*** (смысл каждой фразы такого подъязыка определяется только его формой). Возникла идея построения искусственных формальных языков, в результате чего возникло множество алгоритмических языков.

Алгоритмический язык **- это система обозначений формальной записи алгоритмов, предназначенных для некоторого исполнителя.** Алгоритмический язык довольно близок к обычному разговорному, но более точный, конкретный, лаконичный. **В составе алгоритмического языка - операторы, команды, служебные слова и служебные символы.** Как и любой другой язык, он имеет свой синтаксис и семантику. Вот как выглядит алгоритм Евклида на алгоритмическом языке:

```
алг ЕВКЛИД(цел A, B, H)
арг  A, B
рез  H
нач
нц     пока A ? B
если  A < B
то      X = A;  A = B;  B = X
все
A = A - B
кц
H = A
вывод H
кон
```

Существенным недостатком алгоритмических языков является то, что представленные их средствами алгоритмы недостаточно наглядны, довольно объемны и громоздки. Описания сложных математических задач или процессов управления занимают сотни страниц. Сделать описание алгоритма более наглядным помогает его графическое изображение в виде ***структурной схемы.*** На рис.1 представлена схема алгоритма Евклида.

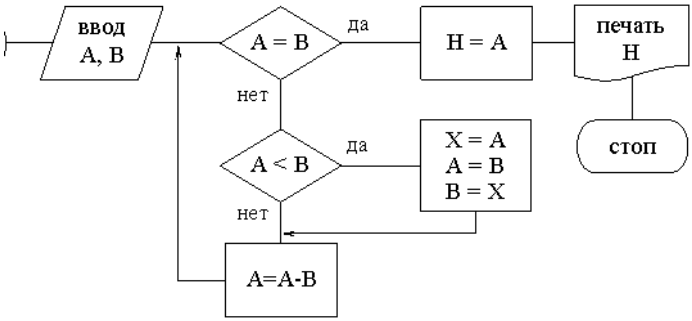


Рис. 1.

Структурная схема является ориентированным графом, у которого графические обозначения (символы, блоки) являются вершинами графа, а линии потока - ребрами. Укрупненная схема позволяет видеть функциональные связи между отдельными фрагментами, участками алгоритма, а более "мелкие", подробные схемы содержат детали, элементарные действия, составляющие содержание этих фрагментов.

Для реализации на ПК алгоритм необходимо описать на одном из ***языков программирования***. При всем своем разнообразии языки программирования (т.н. языки высокого уровня) основываются на сходстве с естественными языками, совместимы с общепринятыми математическими обозначениями и обладают еще более высокой степенью формализации, чем алгоритмические языки. Теперь алгоритм описывается в виде программы. Ниже приведено описание алгоритма Евклида средствами языка Паскаль:

```
PROGRAM  NOD (INPUT,  OUTPUT);
VAR A, B, H, X : INTEGER;
BEGIN
READ (A, B);
WHILE  A <> B  DO
BEGIN
IF  A < B
THEN        X := A;
A := B;
B := X
END;
```

```
A := A - B;
END;
H := A;
WRITE ('НОД чисел', A, 'и', B, 'равен', H)
END.
```

При вводе в ПК специальная программа-транслятор "переводит" алгоритм на ***машинный алгоритмический язык,*** в котором все данные и все действия представляются, в конечном счете, в виде двоичных чисел. После команды на запуск программа выполняется уже автоматически, а программный процессор при этом является физический моделью алгоритма выполнения программы.

4. Методы разработки алгоритмов

Составление алгоритмов решения задач - это работа творческая. Нет универсального способа, позволяющего без особого труда составлять любые алгоритмы. К сожалению, такого способа не существует, ведь жизненные ситуации и задачи так разнообразны и непредсказуемы! Если бы дело обстояло иначе, появилась бы реальная возможность автоматизировать сам процесс алгоритмизации, поручив его некоторому исполнителю - вероятно, очень высокоинтеллектуальному компьютеру.

Тем не менее, некоторые рекомендации, касающиеся методики разработки алгоритмов, можно дать.

При решении простых задач можно воспользоваться определенной схемой. Есть раздел математики, называемый вычислительной математикой, в котором накоплен многолетний (а порой и многовековой) опыт решения разных вычислительных задач. Нет необходимости разрабатывать заново те алгоритмы, которые уже созданы - надо только их изучить и практически применять при решении своих задач. Таковы, например, методы отыскания корней нелинейных уравнений, вычисления определенных интегралов, численного интегрирования дифференциальных уравнений, методы сортировки данных и многие другие.
В большинстве случаев та или иная задача может быть решена несколькими численными методами. Выбор конкретного численного

метода решения задачи обычно производится по следующим критериям:

- обеспечение оптимального времени решения задачи;
- обеспечение оптимального использования имеющихся ресурсов (памяти);
- обеспечение требуемой точности вычислений;
- минимальные стоимостные затраты;
- возможность использования стандартных подпрограмм.

При дальнейшей постановке задачи на ПК отыскивается наиболее рациональный способ решения задачи.

Однако, алгоритмы становятся все более и более сложными, соответственно растет трудность понимания того, как они работают. А еще труднее обнаружить и исправить в них ошибки или внести какие-то изменения. От 50 до 100% времени программист тратит на исправление и модификацию программ. В связи с этим индустрия программирования предлагает более систематичные подходы к программированию (а тем самым и к алгоритмизации задач в общем), т.е. предлагает методики, использование которых уменьшает вероятность ошибок в программах, упрощает их понимание и облегчает модификацию.

Структурное программирование - одна из популярных методик. **Фундаментом структурного программирования является доказанная Бемом и Джекопини *теорема о структурировании.* Эта теорема устанавливает, что как бы сложна ни была задача, блок-схема соответствующей программы (читай - "соответствующего алгоритма") всегда может быть представлена с использованием весьма ограниченного числа элементарных управляющих структур (последовательность, ветвление, цикл).**

Главная идея доказательства этой теоремы состоит в преобразовании каждой части алгоритма в одну из трех основных структур или их комбинацию так, чтобы неструктурированная часть алгоритма уменьшилась. После достаточного числа таких преобразований оставшаяся неструктурированной часть либо исчезнет, либо станет ненужной. Доказывается, что в результате получится алгоритм, эквивалентный исходному и использующий лишь упомянутые управляющие структуры.

Цель структурного программирования - выбор структуры программы путем расчленения исходной задачи на подзадачи. Программы должны иметь простую структуру. Сложные, запутанные программы, как правило, являются неработоспособными, а их тестирование требует больших затрат.

Разработка алгоритма, являясь четким логичным процессом, упрощается на каждом уровне шаг за шагом. Затем в процессе задействуется следующий метод алгоритмизации - ***метод пошагового уточнения*** *(совершенствования)*. Сначала задача рассматривается в целом, выделяются наиболее крупные ее части. Алгоритм, указывающий порядок выполнения этих частей, описывается в структурированной форме, не вдаваясь в мелкие детали. Затем от общей структуры переходят к описанию отдельных частей. Таким образом, **разработка алгоритма состоит из последовательности шагов в направлении уточнения алгоритма.**

Дальнейшим развитием, расширением структурного программирования является ***модульное программирование***, идея которого состоит в том, что алгоритм может быть представлен в виде системы, совокупности отдельных модулей. Каждый **модуль рассматривается как самостоятельная, относительно независимая программа, которая может содержать набор данных и функций, доступных только из этого модуля**.

Модульное программирование позволяет значительно ускорить процесс засчет привлечения к работе нескольких специалистов сразу, доверив каждому разработку отдельного модуля. Кроме того, модульное программирование предполагает возможность использования заранее разработанных стандартных программ (т.н. библиотек стандартных подпрограмм).

На этапе проектирования алгоритма решения сложной задачи, состоящей из нескольких подзадач, используют два подхода: нисходящий и восходящий.
При ***нисходящем проектировании*** вначале проектируются **функции управляющей программы - драйвера**. Затем более подробно представляют каждую подзадачу и разрабатывают другие модули. При нисходящем проектировании на каждом шаге функционирование модуля описывается с помощью ссылок на последующие, более подробные шаги.

При *восходящем проектировании* вначале проектируют программы низшего уровня, иногда в виде самостоятельных подпрограмм. Затем на каждом шаге разрабатываются модули более высокого уровня.

Существует также несколько общих алгоритмических методов решения очень сложных задач. Более подробно ознакомиться с некоторыми из них можно в литературе (методы частных целей, подъема и отрабатывания назад, метод ветвей и границ и др.).

5. Правила оформления схем алгоритмов

Условные обозначения и правила выполнения схем алгоритмов регламентируются требованиями Единой системы программной документации в соответствии с ГОСТ 19.701-90.

Схема алгоритма состоит из символов, краткого пояснительного текста и соединяющих линий. Символы предназначены для графического обозначения отдельных операций, суть которых выражается текстом внутри символов. Символы должны быть по возможности одного размера и располагаться в схеме равномерно, в любой ориентации, но предпочтительным является их горизонтальное расположение.

Внутри символа помещается минимальное количество текста, необходимого для понимания функции данного символа. Если такой текст требует значительного увеличения размера символа, то для размещения текста следует использовать символ "комментарий". Пунктирная линия символа "комментарий" связывается с соответствующим символом или может обводить группу символов (рис. 2).

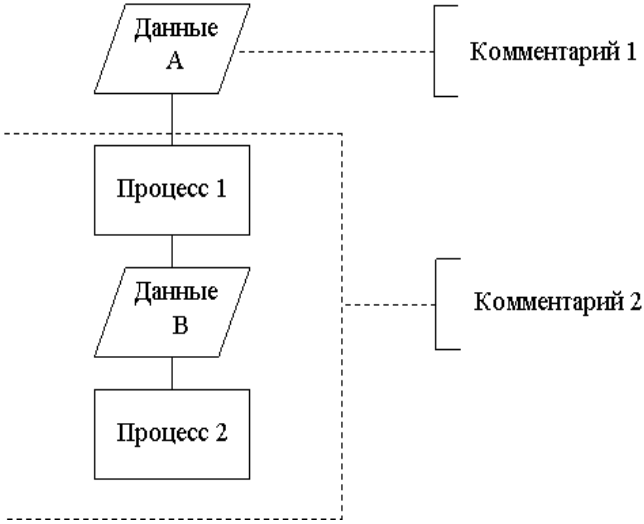


Рис. 2.

Символы в схеме соединяются линиями, которые указывают потоки управления. Направление потока слева направо и сверху вниз считается стандартным. Направление потока, отличное от стандартного, должно быть отмечено стрелкой на конце линии (при вхождении потока в символ или в другую линию потока). Линии должны быть направлены к центру символа. Следует избегать пересечения линий, если потоки в данном месте не входят друг в друга. При необходимости линии в схемах следует разрывать во избежание лишних пересечений или слишком длинных линий, а также, если схема состоит из нескольких страниц. Соединитель в начале разрыва является внешним, а в конце разрыва - внутренним. В комментариях к соединителям могут быть приведены ссылки к страницам (рис.3).

Рис. 3.

Как правило, каждый символ имеет один вход и один выход. Исключение составляют символы:

- "терминатор" (у операции "начало" нет входа, у операции "конец" нет выхода),
- "решение" (один вход и несколько выходов),
- "подготовка" (организация цикла).

Символ "решение" является *логическим.* Каждый выход из символа "решение" должен сопровождаться значением условия, приведенного внутри (рис.4).

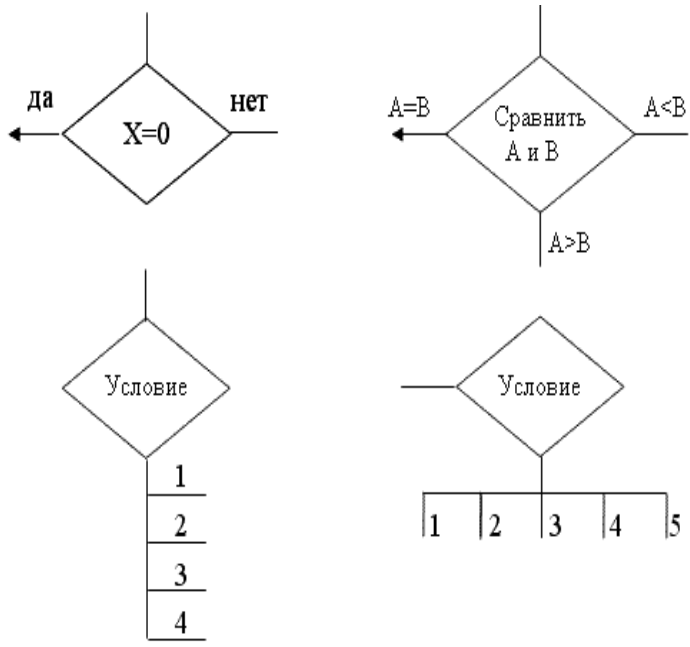


Рис. 4

Представление алгоритма решения задачи в виде схемы является наиболее наглядным, позволяет проследить процесс прохождения данных, связи между отдельными участками программы. Однако, схема должна быть удобочитаемой, т.е. не должна быть чересчур

мелкой, подробной, "перегруженной", чтобы не потерять своей наглядности.

В случае описания решения очень большой, сложной задачи рекомендуется выполнять схему с несколькими ***уровнями детализации***, число которых зависит от размеров и сложности задачи. Уровень детализации должен быть таким, чтобы различные части и взаимосвязь между ними были понятны в целом. При этом весь алгоритм разбивается на смысловые фрагменты, связь между которыми следует указать на более крупной схеме. Сами же фрагменты удобнее выполнять на отдельных страницах, желательно каждый фрагмент размещать на одной странице, не перегружая чрезмерно ссылками и комментариями.

6. Типовые структуры алгоритмов

Из многообразия всевозможных алгоритмов выделяются три основных типовых структуры:

линейная,

разветвляющаяся,

циклическая.

1. Линейная структура

Линейным называется алгоритм, в котором всегда выполняются все действия строго последовательно.

Как правило, алгоритмы линейной структуры состоят из трех частей: ввод исходных данных, вычисления результатов по формулам, вывод значений результатов. Это самые простые алгоритмы.

ПРИМЕР 1. Найти сторону и диагональ квадрата, если известна его площадь.

Математическая постановки задачи. Введем математические обозначения величин: *a* - сторона квадрата, *d* - его диагональ, s - площадь.

Исходными данными задачи является только величина s (она должна быть известна, т.е. введена в начале алгоритма). Величины *a, d* являются результатом решения задачи.

Теперь надо вспомнить формулы, связывающие эти величины:

$s = a\,2$ - формула площади квадрата,

$a\,2 + a\,2 = d\,2$ - теорема Пифагора.

Выбор метода решения задачи. Метод очень простой - прямое вычисление сначала стороны квадрата, а затем его диагонали по формулам:

$$a = \sqrt{s}\,, \qquad d = a\sqrt{2}\,.$$

Разработка алгоритма. Схема алгоритма решения задачи приведена на рис.5. Каждое элементарное действие обозначено в схеме отдельным символом - блоком.

В дальнейшем условимся понятие "символ" графической схемы называть более привычным словом "блок".

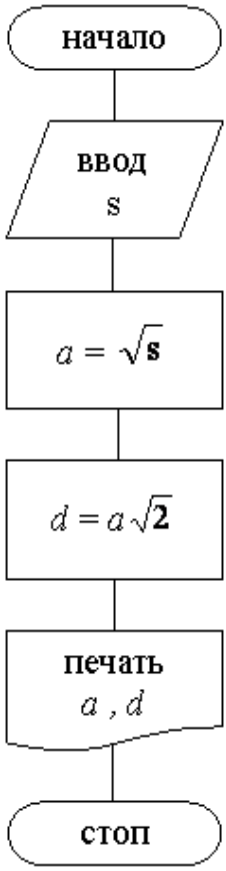


Рис. 5

Порядок выполнения блоков, как уже было сказано, определяют линии потока. По схеме видно, что при вводе совершенно различных исходных данных (по смыслу задачи, конечно, s>0) все действия этого алгоритма будут выполняться всегда, и порядок их выполнения никогда не изменяется.

Для отладки линейного алгоритма достаточно сравнить результаты его исполнения с результатами ручного решения задачи.

2. Разветвляющаяся структура

Разветвляющимсяназывается алгоритм, при выполнении которого каждый раз последовательность действий может быть разная, т.е. каждый раз выбирается один из нескольких путей прохождения схемы алгоритма. Конкретный путь прохождения алгоритма называется ветвью алгоритма. Схема подобного алгоритма обязательно содержит хотя бы один блок (символ) "решение", который и обеспечивает *разветвление* вычислительного процесса.

ПРИМЕР 2. Вычислить значение функции *y* по формуле

$$y = \frac{a^2 + 3}{x - 1}.$$

Математическая постановка задачи. Из условия задачи ясно, что все величины имеют математические обозначения, известна формула для вычислений.

Исходными данными задачи являются переменные *a, x.*

Результат задачи - переменная *y*.

Остается выяснить ограничения для исходных данных. Область определения переменных *a, x* - вся числовая ось. Но есть одно значение переменной *x*, при котором функция *y* не может быть вычислена (при *x*=1 знаменатель обращается в нуль, делить на нуль нельзя).

Выбор метода решения задачи. Прежде чем вычислять функцию *y*, выясним, не обращается ли в нуль знаменатель дроби.

Разработка алгоритма. Схема алгоритма решения задачи приведена на рис.6.

В блоке 3 проверка:

- если знаменатель равен ну-лю, то на экран выдается сообщение, что при заданных исходных данных функция не

существует (блок 4), и решение задачи заканчивается (это одна ветка алгоритма);

- если знаменатель не равен нулю, вычисляем и печатаем значение функции *y* (блоки 6, 7, 8 - это вторая ветка алгоритма).

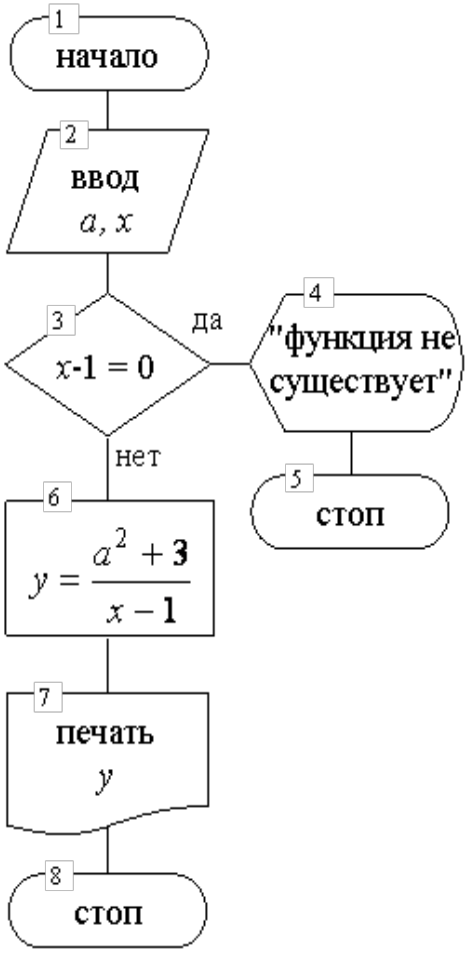


Рис. 6.

Особенность отладки разветвляющихся алгоритмов состоит в следующем: для проверки правильности всех ветвей алгоритма тест должен включать несколько наборов исходных данных - по числу ветвей алгоритма.

Одним из методов проверки правильности алгоритма является его ***трассировка*** (*trace* - след). Она заключается в тщательном, скрупулезном выполнении алгоритма вручную на примере конкретных исходных данных из всей области их определения. В ходе такой проверки должны быть установлены по крайней мере два факта:

- при выборе исходных данных из одной и той же области определения ход выполнения алгоритма всегда один и тот же;
- при выборе исходных данных из разных областей определения ход выполнения алгоритма разный.

Такие утверждения дают право предполагать, что алгоритм составлен правильно. При трассировке схемы удобно записывать пути ее прохождения для последующего анализа - будем последовательно записывать номера блоков, которые выполняются фактически.

Выполним трассировку схемы (рис.6). Выберем два разных набора исходных данных:

a=1, x=1. Путь: блоки 1, 2, 3, 4, 5.

a=1, x=2. Путь: блоки 1, 2, 3, 6, 7, 8.

От значения переменной a путь прохождения схемы не зависит, поэтому мы взяли одно и то же число. Выбор разных значений переменной x привел к получению разных путей прохождения схемы. Если взять любое другое значение (например, x=3), то мы получим второй вариант пути. Итак, схема алгоритма имеет 2 разные ветки.

Конечно, эти рассуждения могут показаться элементарными и ненужными, но умение выполнять трассировку схемы поможет выявить ошибки при разработке схемы алгоритма решения большой и сложной задачи.

ПРИМЕР 3. Найти наибольшее из трех чисел.

Математическая постановки задачи. Введем математические обозначения величин: пусть *A, B, C* - заданные числа (это исходные данные задачи, их значения будем вводить), *M* - наибольшее (максимальное) из чисел *A, B, C* (это искомая величина, результат решения задачи, его значение будем печатать).

Выбор метода решения задачи.

Первый вариант. Сначала сравним первые два числа, а затем большее из них сравним с третьим числом. То число, которое окажется большим после второго сравнения, и есть максимальное.

Разработка алгоритма.

Схема алгоритма решения задачи приведена на рис.7.

В блоке 3 происходит сравнение чисел *A* и *B*. Если *A* окажется больше *B*, то управление передается по ветке "да" и в блоке 4 число *A* сравнивается с числом *C*.

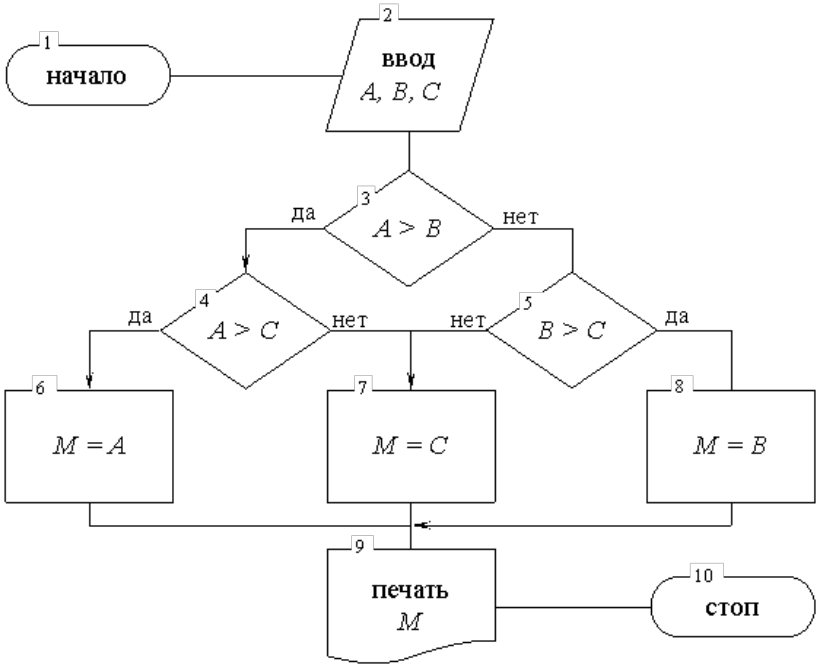


Рис. 7.

Для полной трассировки схемы подберем 4 набора исходных данных, т.к. на схеме прослеживается 4 разных ветки. Проверим их:

A=1, *B*=2, *C*=3. Путь: 1, 2, 3, 5, 7, 9, 10.
A=1, *B*=3, *C*=2. Путь: 1, 2, 3, 5, 8, 9, 10.
A=2, *B*=1, *C*=3. Путь: 1, 2, 3, 4, 7, 9, 10.
A=3, *B*=2, *C*=1. Путь: 1, 2, 3, 4, 6, 9, 10.

Действительно, все пути прохождения схемы разные, а какой-либо другой набор исходных данных даст один из вышеуказанных путей.

Второй вариант. Как правило, любая задача имеет не один способ решения. Даже для этой, на первый взгляд, простой задачи можно предложить еще один вариант. Предположим, что первое число и есть максимальное (присвоим его значение числу *M*). Затем проверим, не окажется ли следующее число больше числа *M*. Если это так, то

заменим значение *M* на это число. Такую же проверку проведем и для третьего числа.

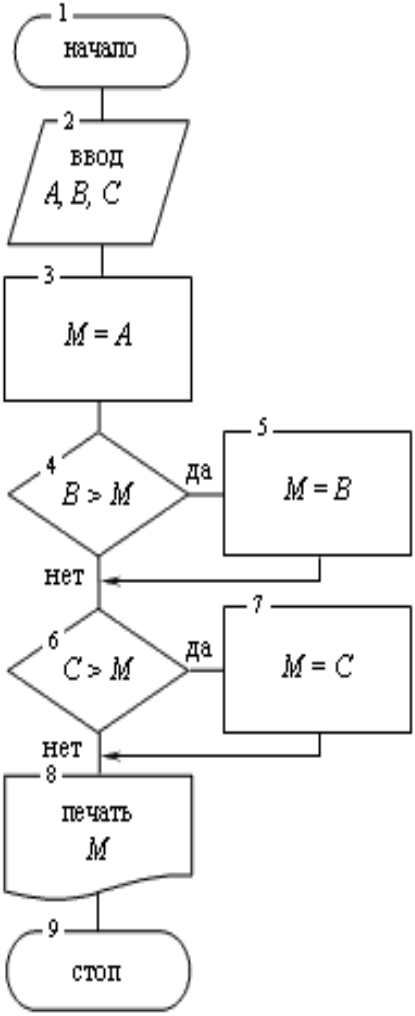


Рис. 8

Разработка алгоритма.

Схема алгоритма решения задачи приведена на рис.8. В блоках 4, 5 происходит возможное уточнение *M* по результату сравнения со вторым числом *B*. В блоках 6, 7 происходит возможное уточнение *M* по результату сравнения с третьим числом *C*.

Схема имеет 4 разные ветки. Попробуйте самостоятельно подобрать 4 разных набора исходных данных для полной трассировки схемы.

Идея, лежащая в основе этого алгоритма, позволяет найти максимальное из любого количества чисел. Для этого достаточно действия 4 и 5 повторять для каждого числа. Как организовать повторение действий? Надо построить цикл. Переходим к рассмотрению следующей структуры алгоритма - циклической.

Циклическая структура

Циклическим называется алгоритм, который содержит участок, выполняющийся многократно, каждый раз с новыми значениями переменных, изменяющихся по одним и тем же законам.
По способу организации циклы делятся на два основных вида:

- циклы с известным заранее числом повторений (*классические*);
- циклы с неизвестным числом повторений (*итерационные*).

Классический цикл организуется с помощью специальной переменной, которая называется параметром цикла.
Параметр цикла - это числовая переменная, которая управляет работой цикла. Она изменяется по закону арифметической прогрессии, что обеспечивает повторение цикла нужное количество раз. Для этого заранее должны быть известны:

начальное значение параметра (обозначим его $t_{нач}$);

конечное значение параметра (обозначим его $t_{кон}$);

- шаг изменения параметра (обозначим его Δt).

Зная эти 3 величины, можно вычислить количество повторений цикла по формуле:

$$k = \left[\frac{t_{нач} - t_{кон}}{\Delta t} \right] + 1 .$$

В этой формуле квадратные скобки обозначают, что после деления берется только целая часть числа (дробная часть всегда отбрасывается, а не округляется), т.к. количество повторений цикла - это целая величина.
Классический цикл имеет 4 части:

- *подготовка цикла* - параметру цикла присваивается начальное значение;
- *тело цикла* - основные действия, которые повторяются каждый раз, на каждом витке цикла;
- *изменение параметра* цикла на величину шага;
- *условие выхода* из цикла (или, напротив - *условие повторения* цикла) - проверка параметра на конечное значение.

От взаимного расположения этих частей зависит вид типовой схемы одиночного цикла. На рис.9 представлен цикл типа "до" (с условием выхода из него), а на рис.10 - цикл типа "пока" (с условием повтора). Оба типа абсолютно равнозначны. На этапе алгоритмизации задачи безразлично, какой из них Вы выберете. Разница между ними обнаруживается только на этапе программирования, т.к. каждому типу цикла соответствуют разные циклические операторы.

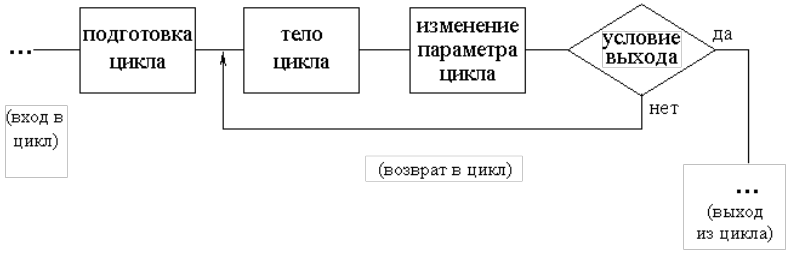


Рис. 9.

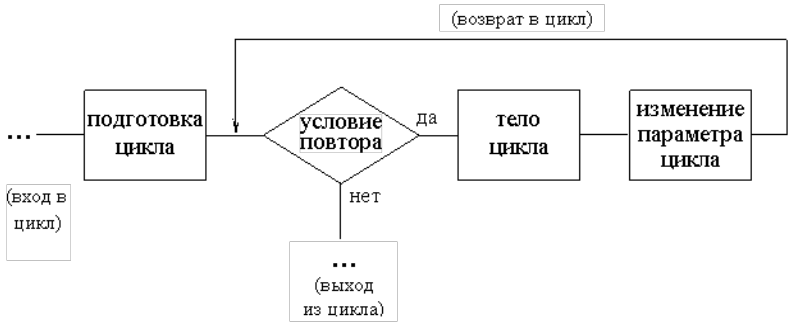


Рис. 10.

Итерационный цикл отличается другой организацией.
ПРИМЕР 4. Построить таблицу значений функции $y = 2sin(x2\text{-}1)$ на заданном интервале изменения аргумента.

Математическая постановка задачи. Введем недостающие обозначения:

(a,b) - интервал;

h - шаг изменения аргумента функции.

Все три величины являются исходными данными задачи. Результат задачи - значения функции, вычисляемые на каждом витке цикла.

Выбор метода решения задачи. На каждом витке цикла будем выбирать точку (значение аргумента), вычислять в ней функцию и печатать оба значения в таблицу.

Разработка алгоритма. В качестве параметра цикла удобно использовать сам аргумент функции, т.к. он изменяется на интервале *(a,b)* с постоянным шагом *h* (т.е. по закону арифметической прогрессии). Тогда начальным значением параметра будет число *a*, а конечным значением - число *b*. В теле цикла - 2 действия: вычисление функции и печать. Схема алгоритма решения задачи приведена на рис. 12.
Определим части цикла на схеме:

- подготовка цикла - блок 3,
- изменение параметра - блок 6,
- условие выхода из цикла - блок 7.
- тело цикла легче всего определять в последнюю очередь (это повторяющиеся в цикле действия, которые не касаются организации цикла). В этом алгоритме тело цикла составляют блоки 4, 5.

Знать принципы организации циклического алгоритма и уметь определять части цикла на схеме (даже "незнакомой" задачи) очень полезно. Это значительно облегчит Вам в дальнейшем процесс программирования алгоритма. Как правило, части цикла, касающиеся его организации (связанные с параметром цикла), описываются во всех языках программирования специальными операторами организации цикла. А действия, составляющие тело цикла, выделяются в отдельную группу операторов. Так что Вы сможете избежать ошибок программирования, если сейчас научитесь определять тело цикла на схеме алгоритма.

Для обозначения циклического процесса есть специальный блок - это символ "подготовка".

Внутри блока указываются параметр цикла, его начальное и конечное значения и шаг изменения в виде

$$t = t_{нач}, t_{кон}, \Delta t .$$

Тело цикла обычно располагается ниже и более четко просматривается на схеме.

На рис.11 приведена типовая схема организации одиночного цикла с помощью блока "подготовка".

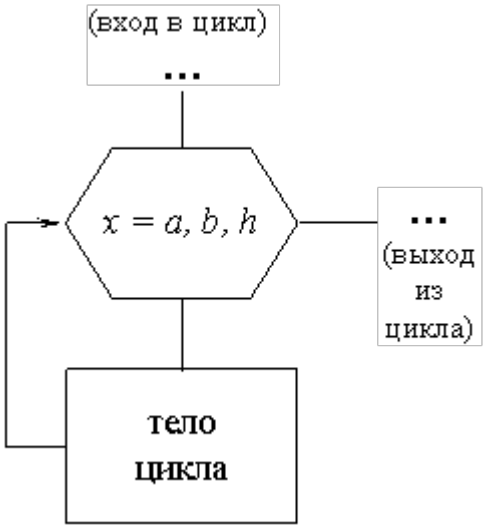


Рис. 11

На рис.13 приведена схема решения задачи табулирования функции с помощью блока "подготовка". Сравните с предыдущей, "развернутой" схемой (рис.12).

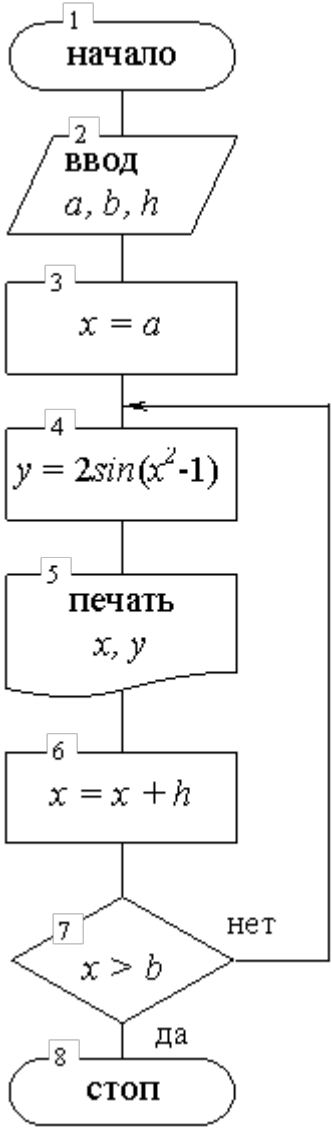

1
начало
2
ввод
a, b, h
3
x = a
4
y = 2sin(x²-1)
5
печать
x, y
6
x = x + h
7
x > b
нет
да
8
стоп

Рис. 12

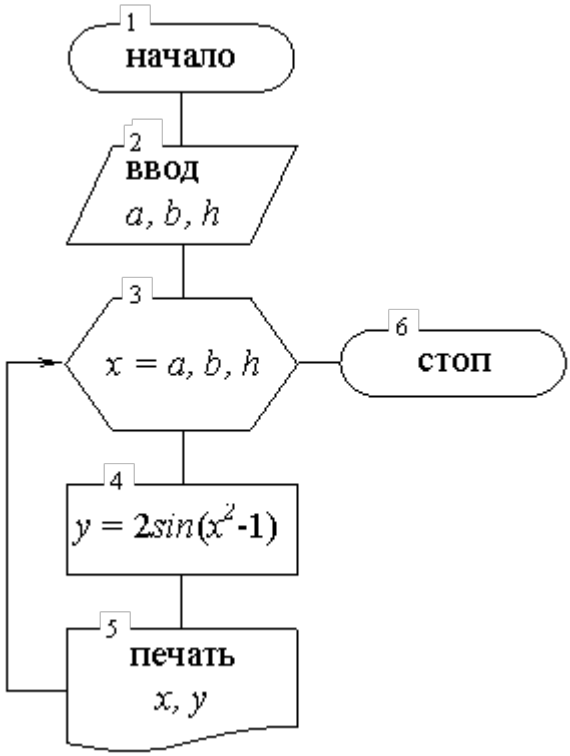


Рис. 13

Тело цикла на этой схеме составляют блоки 4, 5.

Количество повторений цикла (оно определяет количество строк в таблице значений функции) вычислим по формуле

$$k = \left[\frac{b-a}{h}\right] + 1$$

после задания конкретных значений исходных данных.

Конечно, использование символа "подготовка" значительно сокращает размер циклической схемы и упрощает ее восприятие, но говорить о

преимуществе того или иного варианта схемы не стоит, иногда при программировании более полезным оказывается именно "развернутый" вариант схемы. Все зависит от того, какими операторами описания цикла будет реализована конкретная схема.

Конечно, отнести конкретный алгоритм к какой-либо из них полностью удается нечасто, т.к. вычислительные задачи очень разные и по сути, и по методам решения. Однако, любой алгоритм, каким бы сложным он ни был, можно разбить на отдельные части, фрагменты, каждый из которых и является алгоритмом одной из перечисленных типовых структур. Каждая типовая структура имеет свои принципы построения, их необходимо знать и соблюдать при разработке своего алгоритма.

1.3. Классификация облачных технологических систем

Общепринятая классификация облачных технологических систем (ОТС) отсутствует. Мы будем **различають** облачные технологические системы **по назначению, предметной области, методам представления знаний, динамичности и сложности:**

По **назначению** классификацию облачных технологических систем можно провести следующим образом:

- диагностика состояния систем, в том числе мониторинг (непрерывное отслеживание текущего состояния);
- прогнозирование развития систем на основе моделирования прошлого и настоящего;
- планирование и разработка мероприятий в организационном и технологическом управлении;
- проектирование или выработка четких предписаний по построению объектов, удовлетворяющих поставленным требованиям;
- автоматическое управление (регулирование);
- обучение пользователей и др.

По **предметной области** наибольшее количество **облачных технологических систем используется в военном деле, геологии, инженерном деле, информатике, космической технике, математике, медицине, метеорологии, промышленности, сельском хозяйстве, управлении процессами, физике, филологии, химии, электронике, юриспруденции.**

Классификация облачных технологических систем по **методам представления знаний** делит их на традиционные и гибридные.
Традиционные облачные технологические системы используют, в основном, эмпирические модели представления знаний и исчисление предикатов первого порядка.
Гибридные облачные технологические системы используют все доступные методы, в том числе оптимизационные алгоритмы и концепции баз данных, знаний, услуг (сервисов).

По **степени сложности** облачные технологические системы делят на поверхностные и глубинные.
Поверхностные облачные технологические системы представляют знания в виде правил «ЕСЛИ-ТО». Условием выводимости решения является безобрывность цепочки правил.
Глубинные облачные технологические системы обладают способностью при обрыве цепочки правил определять (на основе метазнаний) какие действия следует предпринять для продолжения решения задачи. Кроме того, к сложным относятся предметные области в которых текст записи одного правила на естественном языке занимает более 1/3 страницы.

Классификация облачных технологических систем по **динамичности** делит облачные технологические системы на статические и динамические. Предметная область называется статической, если описывающие ее исходные данные не изменяются во времени. Статичность области означает неизменность описывающих ее исходных данных. При этом производные данные (выводимые из исходных) могут и появляться заново, и изменяться (не изменяя, однако, исходных данных).

Если исходные данные, описывающие предметную область, изменяются за время решения задачи, то предметную область называют динамической. В архитектуру динамической облачной

технологической системы, по сравнению со статической, вводятся два компонента:

- подсистема моделирования внешнего мира;
- подсистема связи с внешним окружением.

Последняя осуществляет связи с внешним миром через систему датчиков и контроллеров. Кроме того, традиционные компоненты статической облачной технологической системы (база знаний и механизм логического вывода) претерпевают существенные изменения, чтобы отразить временную логику происходящих в реальном мире событий.

1.3.1. Схема классификации

Класс " облачные технологические системы " объединяет несколько тысяч различных программных комплексов, которые можно классифицировать по различным критериям. Полезными могут оказаться следующие классификации.

Схема классификации облачных технологических систем

1.3.2. Классификация по решаемой задаче

Интерпретация данных. Это одна из традиционных задач для облачных технологических систем. Под интерпретацией понимается определение смысла данных, результаты которого должны быть согласованными и корректными. Обычно предусматривается многовариантный анализ данных.

Пример 1:

- обнаружение и идентификация различных типов океанских судов — SIAP;

- определение основных свойств личности по результатам психодиагностического тестирования в системах АВТАНТЕСТ и МИКРОЛЮШЕР и др.

Диагностика. Под диагностикой понимается обнаружение неисправности в некоторой системе. Неисправность — это отклонение от нормы. Такая трактовка позволяет с единых теоретических позиций рассматривать и неисправность оборудования в технических системах, и заболевания живых организмов, и всевозможные природные аномалии. Важной спецификой является необходимость понимания функциональной структуры ("анатомии") облачной технологической системы.

Пример 2:

- диагностика и терапия сужения коронарных сосудов — ANGY;

- диагностика ошибок в аппаратуре и математическом обеспечении ЭВМ — система CRIB и др.

Мониторинг. Основная задача мониторинга — непрерывная интерпретация данных в реальном масштабе времени и сигнализация о выходе тех или иных параметров за допустимые пределы. Главные проблемы — "пропуск" тревожной ситуации и инверсная задача "ложного" срабатывания. Сложность этих проблем в размытости

симптомов тревожных ситуаций и необходимость учета временного контекста.

Пример 3:

- контроль за работой электростанций СПРИНТ, помощь диспетчерам атомного реактора — REACTOR;

- контроль аварийных датчиков на химическом заводе — FALCON и др.

Проектирование. Проектирование состоит в подготовке спецификаций на создание "объектов" с заранее определенными свойствами. Под **спецификацией** понимается весь набор необходимых документов — **чертеж, пояснительная записка и т.д**. Основные проблемы здесь — получение четкого структурного описания знаний об объекте и проблема "следа". Для организации эффективного проектирования и, в еще большей степени, перепроектирования необходимо формировать не только сами проектные решения, но и мотивы их принятия. Таким образом, в задачах проектирования тесно связываются два основных процесса, выполняемых в рамках соответствующей ОТС: *процесс вывода решения* и *процесс объяснения.*

Пример 4:

- проектирование конфигураций ЭВМ VAX — 11/780 в системе XCON (или R1), проектирование БИС — CADHELP;

- синтез электрических цепей — SYN и др.

Прогнозирование. Прогнозирующие системы логически выводят вероятные следствия из заданных ситуаций. В прогнозирующей системе обычно используется параметрическая динамическая модель, в которой значения параметров "подгоняются" под заданную ситуацию. Выводимые из этой модели следствия составляют основу для прогнозов с вероятностными оценками.

Пример 5:

- предсказание погоды — система WILLARD;

• оценки будущего урожая — PLANT;
• прогнозы в экономике — ECON и др.

Планирование. Под планированием понимается нахождение планов действий, относящихся к объектам, способным выполнять некоторые функции. В таких ОТС используются модели поведения реальных объектов с тем, чтобы логически вывести последствия планируемой деятельности.

Пример 6 :

- планирование поведения робота — STRIPS;
- планирование промышленных заказов — ISIS;
- планирование эксперимента — MOLGEN и др.

Обучение. Системы обучения диагностируют ошибки при изучении какой-либо дисциплины с помощью ЭВМ и подсказывают правильные решения. Они аккумулируют знания о гипотетическом "ученике" и его характерных ошибках, затем в работе способны диагностировать слабости в знаниях обучаемых и находить соответствующие средства для их ликвидации. Кроме того, они планируют акт общения с учеником в зависимости от успехов ученика с целью передачи знаний.

Пример 7:

- обучение языку программирования Лисп в системе "Учитель Лиспа";
- система PROUST — обучение языку Паскаль и др.

В общем случае все ОТС, основанные на знаниях, можно подразделить на *системы, решающие задачи анализа,* и на *системы, решающие задачи синтеза.* Основное отличие задач анализа от задач синтеза заключается в следующем: если в задачах анализа множество решений может быть перечислено и включено в систему, то в задачах синтеза множество решений потенциально строится из решений компонентов или подпроблем. **Задача анализа — это интерпретация**

данных, диагностика; к задачам синтеза относятся проектирование, планирование. Комбинированные задачи: обучение, мониторинг, прогнозирование.

1.3.3. Классификация по связи с реальным временем

Статические ОТС разрабатываются в предметных областях, в которых база знаний и интерпретируемые данные не меняются во времени. Они стабильны.

Пример 8.
Диагностика неисправностей в автомобиле.

Квазидинамические ОТС интерпретируют ситуацию, которая меняется с некоторым фиксированным интервалом времени.

Пример 9. Микробиологические ОТС, в которых снимаются лабораторные измерения с технологического процесса один раз в 4 - 5 ч (производство лизина, например) и анализируется динамика полученных показателей по отношению к предыдущему измерению.

Динамические ОТС работают в сопряжении с датчиками объектов в режиме реального времени с непрерывной интерпретацией поступаемых данных.

Пример 10. Управление гибкими производственными комплексами, мониторинг в реанимационных палатах и т.д. Пример инструментария для разработки динамических систем — G2,

1.3.4. Классификация по типу ЭВМ

Существуют:

- ОТС для уникальных стратегически важных задач на суперЭВМ (Эльбрус, CRAY, CONVEX и др.);

- ОТС на ЭВМ средней производительности (типа ЕС ЭВМ, mainframe);

- ОТС на символьных процессорах и рабочих станциях (SUN, APOLLO);

- ОТС на мини- и супермини-ЭВМ (VAX, micro-VAX и др.);

- ОТС на персональных компьютерах (IBM PC, MAC II и подобные).

1.3.5. Классификация по степени интеграции с другими программами

Автономные ОТС работают непосредственно в режиме консультаций с пользователем для специфически "экспертных" задач, для решения которых не требуется привлекать традиционные методы обработки данных (расчеты, моделирование и т.д.).

Гибридные ОТС представляют программный комплекс, агрегирующий стандартные пакеты прикладных программ (например, математическую статистику, линейное программирование или системы управления базами данных) и средства манипулирования знаниями. Это может быть интеллектуальная надстройка над ППП или интегрированная среда для решения сложной задачи с элементами знаний.

Несмотря на внешнюю привлекательность гибридного подхода, следует отметить, что разработка таких систем являет собой задачу, на порядок более сложную, чем разработка автономной ОТС. Стыковка не просто разных пакетов, а разных методологий (что происходит в гибридных системах) порождает целый комплекс теоретических и практических трудностей.

1.3.6. Классификация по научно-практическим направлениям

1.3.6.1. Информацио́нные технолóгии

Информацио́нные технолóгии (**ИТ**, также — информационно-коммуникационные технологии) — процессы, методы поиска, сбора, хранения, обработки, предоставления, распространения информации и способы осуществления таких процессов и методов (ФЗ № 149-ФЗ); приёмы, способы и методы применения средств вычислительной техники при выполнении функций сбора, хранения, обработки, передачи и использования данных (ГОСТ 34.003-90); ресурсы,

необходимые для сбора, обработки, хранения и распространения информации (ISO/IEC 38500:2008).

Термин «информационные технологии» в его современном смысле впервые появился в статье 1958 года, опубликованной в Harvard Business Review; авторы Гарольд Дж. Ливитт и Томас Л. Уислер прокомментировали, что «у новой технологии еще нет единого установленного имени. Мы будем называть это информационными технологиями (ИТ)». Их определение состоит из трех категорий: методов обработки, применения статистических и математических методов для принятия решений и моделирования мышления более высокого порядка с помощью компьютерных программ.

Основываясь на используемых технологиях хранения и обработки, можно выделить четыре отдельных этапа развития ИТ: предварительные механические (3000 до н. э. — 1450 н. э.), механические (1450—1840), электромеханические (1840—1940) и электронные (1940 — настоящее время).

Основные черты современных ИТ

- Структурированность стандартов цифрового обмена данными алгоритмов;
- Широкое использование компьютерного сохранения и предоставление информации в необходимом виде;
- Передача информации посредством цифровых технологий на практически безграничные расстояния.

Основные средства

Информационные технологии охватывают все ресурсы, необходимые для управления информацией, особенно компьютеры, программное обеспечение и сети, необходимые для создания, хранения, управления, передачи и поиска информации. Информационные технологии могут быть сгруппированы следующим образом:

- Технические средства;
- Коммуникационные средства;
- Организационно-методическое обеспечение;
- Стандартизация.

1.3.6.2. Мультимедийные технологии

Виды, задачи, роль, применение мультимедийных технологий

Характеристика мультимедийных технологий – основа развития информационного направления. Сегодня это одно из наиболее перспективных, популярных, непрерывно развивающихся направлений информатики.

Под понятием «мультимедийная технология» подразумевается создание продукта, который путем внедрения и использования новых технологий, набора изображений, текстов и данных, сопровождающихся звуком, видео, анимацией и прочими визуальными эффектами, информирует аудиторию.

Мультимедийные технологии включают также интерактивный интерфейс и прочие механизмы управления. С целью того, чтобы лучше разобраться и понять, какие существуют виды мультимедийных технологий, следует определить и выделить основные направленности их использования. Это действительно важно.

Виды мультимедийных технологий

Применение мультимедийных технологий подразделяется на:

- общее или индивидуальное пользование;
- для профессионалов или для рядового потребителя;
- для применения интерактивного и неинтерактивного;
- для использования информации по месту или на расстоянии.

Стоит более подробно остановиться на каждом из перечисленных пунктов.

1. **Технологии общего или индивидуального пользования**. Касательно технологий общего пользования можно выделить следующие виды: интерактивные терминалы, некоторые

технологии презентаций посредством компьютера, те, что ширятся по сетям. В свою очередь, к технологиям индивидуального пользования можно отнести мультимедийные рабочие места, учебные классы, мультимедийные компьютеры для ведения различных документов. К основным местам их применения можно отнести общественные зоны, а также дома и рабочие места потребителей.

2. **Технологии для профессионалов и рядовых потребителей**. В эту категорию можно отнести рабочие зоны мультимедиа (компьютерная графика, проекты и т.п.). Также сюда могут входить системы, применяемые не знатоками. Они, как правило, используются в общественных местах, это системы со встроенными микропроцессорами, которые предназначены для функционирования в быту. Это игровые приставки, CD-I, Play Station.

3. **Использование информации по месту и на расстояниях**. Стремительное развитие на начальном этапе мультимедиа можно объяснить быстрым процессом развития стационарных компьютеров, которые сегодня есть дома у каждого. Тогда стала вероятной запись и хранение информации на специально предназначенных компакт-дисках. Современность диктует свои правила. Сегодняшнее стремительное развитие цифровых сетей средней и высокой пропускной способности позволяет говорить о стремительном развитии дистанционных мультимедийных технологий.

4. **Применение интерактивных и неинтерактивных технологий**. Подходя к данной категории, следует акцентировать внимание на том, что большое количество специалистов не согласны с тем, что неинтерактивные системы можно назвать мультимедийными. Но важно

понимать, что их количество может существенно увеличиться. Так, неинтерактивные мультимедиа применяются для привлечения внимания и развлечения аудитории посредством демонстрации презентаций и выставок.

Особенно важно понимать, в чем заключается роль мультимедийных технологий. На этом следует остановиться более подробно.

Значение и роль мультимедийных технологий

Значение мультимедиа сегодня достаточно велико. Одной из основных сфер, где данные технологии проявили себя, можно назвать образовательную. Их сегодня очень активно внедряют и успешно применяют для обучения. Разрабатываются новые эффективные и действенные средства подачи информации и ее донесения до учеников. Так, одним из распространенных и привычных сегодня способов внедрения в образовательный процесс можно назвать презентацию.

В ходе проведения на экранах больших масштабов предлагается информация для изучения. Такая мультимедийная технология, как презентация, может проходить на разных этапах обучения:

- в момент актуализации опорных знаний;
- в ходе фронтального опроса осуществляется вывод текста вопроса на экран, а после достоверного ответа учениками происходит переход по гиперссылке к слайду с визуализацией ответа;
- под видом фреймовых опор отображаются этапы решения задач, от которых быстро можно перейти на слайд с новыми начальными условиями или рисунком, а после продолжить решение.

Такой подход способствует существенной экономии времени, которое отводится на занятие. У преподавателя появляется возможность оценить уровень знаний большего количества учеников.

И это лишь один пример. Роль мультимедиа достаточно велика во всех сферах жизнедеятельности в современном мире.

Основные цели мультимедиа

Цель мультимедийных технологий может варьироваться в зависимости от специфики применения. Как правило, это:

- популяризаторская и развлекательная;
- образовательная и научно-просветительская;
- научно-исследовательская и т.п.

Рассматривая подробнее каждую из них, следует сказать, что, к примеру, популяризаторская цель является одной из основных. Рекламная деятельность активно использует мультимедиа с целью привлечения потенциальных покупателей и клиентов.

Научно-просветительское стремление активно применяется в следующих направлениях:

- отбор посредством жесткого анализа представленной на рынке продукции, которая может применяться в соответствующих рамках;
- разработка мультимедийного продукта преподавателями, исходя их преследуемых целей и поставленных задач в ходе учебного, образовательного процесса.

Говоря о научно-исследовательских целях, на ум сразу приходит применение мультимедийных технологий для создания всяческих электронных архивов. Так или иначе, но особенности мультимедийных технологий кроются в их вездесущности и широте применения.

Применение, функции и задачи мультимедийных технологий

Примечательно, что функции мультимедийные технологии выполняют, исходя из сферы их применения.

Мультимедиа применяется в таких сферах:

- медицина;
- техника;
- промышленность;
- образование;
- научные исследования;
- искусство;
- реклама и т.д.

Говоря об основных, следует сказать, что в образовательной сфере, как уже говорилось ранее, мультимедиа выполняет функцию образовательного характера. Технологии применяются для создания компьютерных учебных курсов. В промышленной отрасли обширно используются в качестве презентации данных для лиц, занимающих руководящие должности.

Значение для медицины особенно велико. Докторам представляется сегодня уникальная возможность пройти качественную подготовку посредством операций виртуального характера. Разработчики ПО применяют мультимедиа в компьютерных симуляторах чего угодно.

Отталкиваясь от сфер применения и функций данных технологий, очевидным является и постановка задач. Для каждой отдельной отрасли ставятся свои цели и задания, достижение которых посредством мультимедиа позволяет совершенствоваться.

Так, задачи мультимедийных технологий в образовательной сфере построены на повышении эффективности процесса обучения. В рекламе, главная задача – достижение поставленных целей, донесение информации до аудитории и продвижение в такой способ товара либо услуги.

Подробнее о средствах мультимедийных технологий

Средства мультимедийных технологий подразделяют на два класса. Основанные на взаимодействии и на их применении.

К первой категории правильно будет отнести средства синхронного, асинхронного взаимодействия, онлайн режим.

Вторая категория включает разнообразные виртуальные объекты, реальные видео-, аудиофрагменты, анимационную графику и т.п.

Для создания и воплощения таких технологий потребуется ПК, соответствующее программное обеспечение, а также средства конструирования мультимедийных проекторов для отображения на больших экранах.

Для того чтобы получить изображение, а также звуковое сопровождение, требуется подсоединить мультимедийный проектор к компьютеру.

1.3.6.3. Информационно-коммуникационные технологии

Информационно-коммуникационные технологии (ИКТ) – технологии, основная задача которых заключается в обеспечении фиксации информации, ее обработки, передачи, распространении и раскрытии. ИКТ подразумевает под собой методы и программно-технологические средства, которые позволяют в значительной мере снизить всю сложность процесса использования информации.

ИКТ включают в себя компьютеры, программное обеспечение и средства электронной связи. Иногда, данный ряд расширяется и технологиями управленческого консультирования, и проектированием бизнеса, административных процессов. Они делятся на три группы:

1. Сберегающие – технологии, которые призваны хранить данные.
2. Рационализирующие – автоматические системы поиска и заказов.
3. Творческие – технологии, которые с помощью которых человек включается в активную работу с информацией

ИКТ призваны экономить много времени, труд и материальную базу.

Современное и развитое информационное общество сегодня широко применяет информационно-коммуникационные технологии в разных сферах своей жизнедеятельности: в образовании, производстве, промышленности и в других сферах жизнедеятеольности. Необходимость в использовании информационно-коммуникационных технологий обусловлена несколькими основными факторами:

1. Внедрение ИКТ в сферы жизнедеятельности человека ускоряет передачу знаний и накопленного социального опыта, который передается веками от одного человека к другому.

2. ИКТ значительно повышает качество обучения и способствует быстрому освоению необходимых навыков в той или иной сфере жизнедеятельности.

3. ИКТ позволяет человеку быстрее и более успешно адаптироваться к происходящему социальному изменению.
4. ИКТ – это эффективный способ обновления существующих систем в соответствии с требованиями современного общества.

ИКТ предоставляют человеку дополнительные возможности для формирования и развития его информационной компетенции.

Их внедрение в жизненоважные сферы человека приведет к систематизации и интегрированию информационных потоков в определенном социальном пространстве, формированию субъективной позиции человека на основе освоения знаний ИКТ, успешному проектированию и проверки достижений человека в процессе освоения общих и профессиональных компетенций.

1.3.6.4. Управленческие технологии

Управленческие технологии: типы и характеристика

Эффективность деятельности предприятия, его положение на рынке принципиально зависят от стратегии развития, выбора и применения как отдельных стратегических типов управленческих технологий, так и их различных сочетаний.

Управленческие технологии – это набор управленческих средств и методов достижения поставленных целей организации, включающий методы и средства сбора и обработки информации; приемы эффективного воздействия на работников; принципы, законы и закономерности организации и управления; системы контроля.

Для фирм и предприятий, различающихся по численности, организационно-правовой форме, организации технологического процесса, могут быть эффективны различные типы управленческих технологий, а именно: управление по целям; управление по результатам; управление на базе потребностей и интересов; управление путем постоянных проверок и указаний; управление в исключительных случаях; управление на базе активизации деятельности персонала; управление на базе «искусственного интеллекта» и др.

Управление по целям применимо для средних и малых предприятий с сильным аналитическим подразделением. Оно бывает простым целевым, программно-целевым и регламентным.

При *простом целевом управлении* руководитель организации определяет только сроки и конечную цель, но не механизм ее достижения. Цель может быть достигнута в любой срок или не достигнута вовсе. Такой способ управления применяется в основном обществом с ограниченной ответственностью численностью 3– 5 человек.

Программно-целевое управление предусматривает определение целей, механизмов и сроков для каждого этапа достижения целей. Общая цель достигается в предусмотренные сроки. Такой способ управления применяется, как правило, обществами с ограниченной ответственностью, акционерными обществами всех типов.

Регламентное управление используется на уровне всей экономики. При этом определяются конечная цель и ограничения по параметрам и ресурсам. При этом цель достигается обязательно, но сроки ее достижения установить трудно.

Управление по результатам базируется на усилении функции координации и интеграции деятельности всех подразделений. Эта технология хорошо реализуется в средних и малых орган/нациях, где невелико время между принятием решения и его результатом.

Для реализации технологии необходимо создание в рамках отделов аналитических групп (1– 3 человека) с включением специалистов в области психологии, социологии, маркетинга, экономики, работающих в рамках матричной структуры управления. Задачи группы: анализ текущей информации, проведение опросов, определение проблем и подготовка предложений по корректировке тактических и стратегических решений.

Управление на базе потребностей и интересов основано на стимулировании деятельности человека через его потребности и интересы, к которым относятся основные потребности в пище, жилье, отдыхе, здоровье, социальные потребности в творческом труде, семье, порядке и стабильности, интересы материальные, социальные, эстетические. Данную технологию управления рекомендуется использовать в небольших регионах (малых городах, поселках и т.д.), где деятельность организации непосредственно влияет на муниципальную инфраструктуру.

Управление на базе активизации деятельности персонала реализуется путем стимулирования (морального и материального) персонала и мобилизации его интеллектуального потенциала. Основная задача такого рода управления состоит во влиянии на эмоциональное состояние человека. Применяется в организациях самых разнообразных форм.

Управление в исключительных случаях заключается в четком распределений всех' управленческих и производственных функций, основной формальный руководитель осуществляет лишь связи с внешней средой. Данная технология рекомендуется для организаций с жестко регламентированной технологией или **с** доверительной (функциональной) структурой управления.

Управление путем постоянных проверок и указаний основано на жестком планировании деятельности подчиненных и постоянном контроле руководителя за всей текущей деятельностью. Предусматривает линейную структуру управления. Данная технология эффективна для небольших организаций, в которых высок авторитет и профессионализм руководителя; обычно это временная технология.

Управление на базе «искусственного интеллекта» реализуется на базе информационных систем с применением современных технических средств.

Эти и ряд других типов управления должны реализовываться на базе современных информационных технологий. Главное требование к управлению предприятием в условиях рынка – обеспечение адаптивности (приспособляемости и гибкости) экономики предприятия к изменяющимся условиям хозяйствования.

Управленческий консалтинг

"Управленческое консультирование является набором услуг, оказываемых специально обученными и имеющими соответствующую квалификацию лицами, которые в объективной и независимой манере помогают клиенту выявить и проанализировать проблемы данной организации и рекомендуют решения этих проблем, а также, при необходимости, оказывают помощь в реализации предложенных решений" (Ларри Грейнер и Роберт Метцгер)

"Управленческое консультирование заключается в предоставлении независимых рекомендаций и поддержки для клиентов, обладающих руководящими полномочиями, в вопросах, касающихся процесса управления» (Международный совет институтов управленческого консультирования)

Основные цели консультирования

В книге "Управленческое консультирование" под редакцией М.Кубра выделятеся пять общих целей, которые преследуют клиенты, прибегающие к услугам консультантов:

- достижение целей и задач организации
- решение управленческих и деловых проблем
- выявление и использование новых возможностей
- обучение
- внедрение изменений

Виды управленческого консалтинга

В процессе управленческого консалтинга может решаться самый широкий круг задач. По типам решаемых задач управленческий консалтинг может быть (достаточно условно) разделен на:

- стратегический консалтинг, в ходе которого осуществляется анализ глобального и регионального рынка сырья и готовой продукции, анализ конкурентов, динамики производства и потребления, рассматривается эволюция технологий, строится эффективная бизнес-модель, осуществляется расчет логистики;
- маркетинговый консалтинг, в ходе которого осуществляется построение эффективной маркетинговой стратегии, разрабатывается программа маркетинга компании, строится система маркетинга как технологии управления рыночным поведением потенциальных и актуальных покупателей;
- построение эффективной системы управления (распределение функций, полномочий, ответственности, материальных стимулов, построение системы бизнес-процессов, оптимальной системы информационного обмена и документооборота, внедрение системы прогнозирования, планирования и анализа деятельности, построение оптимальной структурно-функциональной схемы);
- кадровый консалтинг (подбор кадров), построение и развитие корпоративной культуры (конфигурирование системы нематериальных стимулов, привнесение смысла в коллективную деятельность сотрудников компании).

Стратегический консалтинг

Это базовый вид управленческого консалтинга, представляющий собой видение и общее описание бизнес-модели, её преимуществ и недостатков в сравнении с основными конкурентами, изучение рыночной ситуации, тенденций производства и потребления сырья и готовой продукции на глобальном и региональном рынках. По сути стратегия — это базовое самоопределение компании, из которого напрямую следуют цели, условия их достижения и средства, которые компания должна для этого использовать. Существует также иная точка зрения о том, что стратегический консалтинг не относится к управленческому консалтингу, и является самостоятельной отраслью.

Маркетинговый консалтинг

Каждая компания стремится управлять рыночным поведением своих потенциальных и актуальных клиентов, давая им множество причин выбрать именно эту компанию именно сейчас и именно для покупки данного товара (услуги). Управление рыночным поведением потенциальных и актуальных клиентов компании представляет собой достаточно комплексную задачу. Решить её можно только внедрив в деятельность компании функцию системного маркетинга, представляющую собой привнесение маркетингового смысла в каждый контакт с потенциальным клиентом и в саму систему предоставления услуг и продажи товаров.

Построение эффективной системы управления (Операционный консалтинг)

Проектирование и внедрение эффективной системы управления включает в себя продумывание формулы эффективности бизнеса (системы индикаторов, показателей, на основе которых можно оценивать состояние бизнеса). Также эффективная система управления предполагает наличие оптимальной системы распределения функций, полномочий, ответственности, а также построенной системы мотивации персонала. В ходе управленческого консалтинга оптимизируется система бизнес-процессов, а также структурно-функциональная схема, упорядочивается информационный обмен и документооборот между подразделениями консультируемого предприятия.

Кадровый консалтинг. Развитие корпоративной культуры

В ходе кадрового консалтинга осуществляется подбор кадров, которые могут органично дополнить друг друга по своим деловым качествам, происходит подбор топ-менеджеров, правильное распределение между ними функций и полномочий, формируется команда людей, которые готовы работать вместе в рамках осуществляемого ими бизнес-проекта.

В ходе формирования и развития корпоративной культуры управленческие консультанты формируют систему мотивационных и смысловых установок, которые начинают играть в компании роль внутренних норм, которые определяют деятельность сотрудников и

дают сотрудникам ценностные и нормативные ориентиры при принятии решений. Кадровый консалтинг - это вид деятельности, включающий в себя комплекс мероприятий по анализу персонала, диагностики юридической и делопроизводственной корректности оформления кадровых документов и предложений по устранению нарушений (кадровый аудит), оценке соответствия профессиональных и личностных компетенций выполняемым обязанностям, уровня лояльности сотрудников и др.

Роли консультанта

Понять суть управленческого консалтинга помогают роли консультанта, отражающие его действия в отношениях с руководителями, которым оказывается помощь. Шейн, Эдгар считает, что для эффективного консультирования специалисту необходимо переключаться с одной роли на другие по мере изменения и/или понимания ситуации, для чего он должен видеть разницу между отдельными видами действий:

- позиция эксперта, говорящего клиенту, что делать;
- внушение решений, кажущихся консультанту подходящими, и популяризацией методик, которыми консультант умеет пользоваться;
- вовлечением клиента в процесс, результатом которого станет выработка решения, которое и клиент, и консультант сочтут полезным в данной ситуации.

Из этого разделения вытекают три модели консультирования:

- "Продаю и говорю" - организация определяет потребность и ищет консультанта, способного снабдить ее информацией или необходимыми услугами. Нанятый консультант выступает в роли эксперта по удовлетворению запроса, сформулированного менеджментом компании.
- "Врач-пациент" - консультант приглашается для выявления организационных областей, в которых функционирование происходит не так, как надо. Возможен вариант, когда компания нанимает специалиста по реализации какой-то программы (всеобщего управления качеством, реинжиниринг и т.д.), но в этой модели консультант-врач сам "ставит диагноз, прописывает лекарство и управляет лечением".

- Консультирование по процессу - установление между консультантом и клиентом таких отношений, которые увеличивают способности организации к обучению, как этого хочет сам клиент. Вместо того, чтобы давать советы и ставить диагнозы консультант учит компанию самостоятельно удовлетворять потребности.

Подходы к управленческому консалтингу

В процессе работы со специалистами и управленцами консультируемой организации управленческие консультанты используют два принципиально отличающихся подхода: рекомендательный и социально-инжиниринговый.

Рекомендательный подход состоит в разработке тщательных и профессиональных рекомендаций, которые стали результатом работы экспертов, вовлеченных в консалтинговый проект.

Социально-инжиниринговый подход предполагает использование социальных технологий, суть которых — в вовлечении управленцев и специалистов консультируемой компании в процесс консалтинга, в конструирование будущего устройства компании, её системы управления, структурно-функциональной схемы, системы бизнес-процессов и т. д. Вовлечение персонала компании в процесс формирования будущего позволяет решить множество проблем, связанных с лояльностью сотрудников к предложенным изменениям. Сотрудники, принявшие участие в формировании будущего своей компании, охотно соглашаются с необходимостью перемен, и активно участвуют в формировании новой системы управления.

При этом наиболее острые и принципиальные позиционные конфликты оказывается возможным разрешить как раз на стадии консалтингового проекта, ещё до начала внедрения. Таким образом, использование социальных технологий существенно повышает внедряемость рекомендаций консультантов. Социальные технологии — это способы формирования команд исполнителей и управленцев, которые могут осуществлять коллективную деятельность с заранее заданным уровнем эффективности. Одна из форм социальных технологий — рефлексивная игра.

В решении текущих рабочих задач статус советника может быть ограниченным или абсолютным: от анализа причин и поиска решений до возможности влиять на решение или участвовать в его принятии и реализации.

На него может возлагаться обзор ситуации в целом или разработка детального, конкретного, измеримого, согласованного, реалистичного и ограниченного по времени плана действий. Иногда обстановка может потребовать от консультанта личного присутствия непосредственно на объекте, например при ведении переговоров и заключении соглашения. При непосредственном присутствии на объекте ситуация изучается на месте с применением "живого моделирования" и многодневных стратегических сессий, во время которых разработка и внедрение управленческих решений происходит при прямом взаимодействии с персоналом компании. Формирование проекта изменений в таком формате автоматически способствует его внедрению и оптимизации.

Услуги управленческих консультантов

Работа с компаниями (бизнес-структурами)

- Повышение эффективности бизнеса, системы управления, команды управленцев и специалистов.
- Обеспечение эффективных слияний, поглощений, реструктуризаций.
- Анализ действующей бизнес-модели и её оптимизация.
- Построение финансовой модели бизнеса и обучение топ-менеджеров использованию этой модели для прогнозирования результатов деятельности компании.
- Создание и развитие корпоративной культуры, системы мотивации персонала.
- Формирование команды для реализации бизнес-модели (для новых бизнес-проектов).
- Формирование или развитие системы бизнес-процессов, системы информационного обмена и документооборота.
- Формирование системной функции маркетинга и её интеграция в коллективную деятельность компании.
- Повышение эффективности топ-менеджмента компании, обогащение арсенала управленческих инструментов.

- Формирование дальнего, среднего и ближнего горизонтов развития компании, моделирование многовариантного будущего, выбор направления и маршрута развития.

Работа с мегасистемами

- Разработка стратегии социально-экономического развития города, области, региона.
- Разработка стратегии развития отрасли.
- Расчет мультипликативного эффекта инноваций и построение модели взаимовлияния отраслей экономики друг на друга и на основные индикаторы социально-экономического развития.
- Решение стратегических задач, связанных с управлением развитием экономики страны, области, региона, например:

    - повышение эффективности управления государственными активами в экономике;
    - повышение привлекательности региона для туризма;
    - разработка и сопровождение внедрения концепции свободной экономической зоны;
    - разработка и сопровождение внедрения концепции бизнес-инкубатора;
    - разработка и сопровождение внедрения концепции технопарка;
    - разработка и сопровождение внедрения концепции развития системы образования;
    - повышение эффективности государственных администраций в вопросах развития региона, области, города;
    - повышение эффективности министерств, ведомств во внедрении различных стратегических программ.

Этапы оказания услуги управленческого консалтинга Процесс оказания услуги управленческого консультирования подразделяется на три стадии: предпроектную, проектную и постпроектную. Проектная стадия в свою очередь подразделяется на три этапа: диагностика, планирование и внедрение. Таким образом, мы получаем пять этапов. Предпроектный этап являет собой обнаружение клиентом проблемы и принятие решения обратиться в консалтинговую компанию. Далее следуют этапы работы консультантов: диагностика бизнеса клиента,

разработка решений и их внедрение в компанию. Постпроектная стадия — это анализ достигнутых результатов, определение реализованных и нереализованных планов, оценка нынешнего положения дел в компании. Итак, в непростой период экономического кризиса управленческое консультирование — эффективный способ уберечь компанию от финансовых и кадровых потерь. Кроме того, такая услуга может выявить проблему еще до того, как она принесет какие-либо последствия. Управленческий консалтинг — не самая дешевая услуга для бизнеса, но если ее оказывают квалифицированные специалисты, она может принести доход, в разы превосходящий ее стоимость.

1.3.6.5. ОБЛАЧНЫЕ ВЫЧИСЛЕНИЯ: НОВЫЕ ТЕХНОЛОГИИ В ОБРАЗОВАНИИ

В настоящее время умение эффективно использовать компьютерные технологии для решения прикладных задач является необходимым атрибутом профессиональной деятельности любого специалиста и во многом определяет уровень его востребованности в обществе. Поэтому подготовка студентов высших учебных заведений не возможна без использования современных технологий обучения. Речь, прежде всего, идёт о применении в учебном процессе компьютерных информационно-коммуникационных технологий.

Программа подготовки студентов предусматривает оснащенность высшего учебного заведения комплексом современных аппаратных средств (компьютерная и цифровая техника) и соответствующим программным обеспечением. Повседневной реальностью становится использование мобильных вычислительных устройств (планшетов, ультрабуков, электронных книг, смартфонов и т.п.), постоянный широкополосный доступ в Интернет.

Облачные вычисления – быстро развивающаяся область IT. Термин «облачные вычисления» появился чуть более пяти лет назад.[1] На рынке информационных технологий уже предлагаются комплексные решения, которые позволяют предоставлять облачные сервисы различным категориям потребителей: финансовому сектору, промышленности, торговле, сфере услуг, сектору телекоммуникаций и, конечно науке и образованию.

«Облако» (cloud computing) обозначает сложную инфраструктуру с большим количеством технических деталей, спрятанных в «облаках».

Национальный институт стандартов и технологий США (National Institute of Standards and Technology – NIST) в документе «The NIST Definition of Cloud Computing v15» [5] определил «облачные вычисления» следующим образом – это особая модель предоставления повсеместного и удобного сетевого доступа (по мере необходимости) к общему пулу конфигурируемых вычислительных ресурсов (например: сетей, серверов, систем хранения, приложений и сервисов), которые могут быть быстро предоставлены и освобождены с минимальными усилиями по управлению и необходимостью взаимодействия с провайдером услуг. Таким образом, облако – это предоставление провайдером удаленных вычислительных ресурсов и услуг по запросу потребителя.

Модель облака содействует доступности и характеризуется пятью основными элементами (самообслуживание по требованию, широкий доступ к сети, объединенный ресурс, независимое расположение, быстрая гибкость, измеряемые сервисы). Облако содержит три сервисные модели (программное обеспечение как услуга, платформа как услуга, инфраструктура как услуга) и четыре модели развертывания (частное облако, коммунальное облако, публичное облако, гибридные облако).

Профессор Массачусетского технологического института (MIT) Карл Хеввит отметил, что при облачных вычислениях данные постоянно хранятся на виртуальных серверах, расположенных в облаке, а также временно кэшируются на клиентской стороне – на компьютерах, ноутбуках, нетбуках, мобильных устройствах и т.п.[4]

Для построения облака используют одну из трех базовых моделей: программное обеспечение как сервис, платформу как сервис, инфраструктуру как сервис. На рис. 1 представлена сервисная модель архитектуры облачных вычислений, из которой видно, что основу облака составляет инфраструктура как сервис (IaaS – Infrastructureas a Service), затем на нее накладывается платформа как сервис (PaaS – Platform as a Service), а поверх PaaS – программное обеспечение как сервис (SaaS – Software as a Service).

Инфраструктура как сервис (IaaS, infrastructure as a service) — предоставление компьютерной инфраструктуры как услуги на основе концепции облачных вычислений. На этом уровне пользователи получают базовые вычислительные ресурсы. Например, процессоры и устройства для хранения информации используют их для создания своих собственных операционных систем и приложений. Одним из примеров такого подхода является Amazon Elastic Compute Cloud (Amazon EC2) — организации могут использовать эту инфраструктуру, устанавливая на виртуальных машинах linux-серверы, и при необходимости наращивать вычислительные мощности. Такая модель подразумевает бесплатное предоставление ресурсов хранения данных, функций электронной почты и систем совместной работы, что может быть интересным для образовательных учреждений.

Платформа как сервис (PaaS, platform as a service) — это предоставление интегрированной платформы для разработки, тестирования, развертывания и поддержки веб-приложений как услуги. Здесь пользователи имеют возможность устанавливать собственные приложения на платформе, предоставляемой провайдером услуги. В качестве примера можно привести сервис Google Apps Engine, позволяющий разработчикам создавать и устанавливать приложения на языке Python. Программное обеспечение как сервис (SaaS, software as a service) — модель развертывания приложения, которая подразумевает предоставление приложения конечному пользователю как услуги по требованию. При этом в облаке хранятся не только данные, но и связанные с ними приложения, а пользователю для работы требуется только веб-браузер. Именно этот уровень представляет наибольший интерес для образовательного процесса. Лучшими примерами такого подхода являются системы Google Apps for Education и Microsoft Live@edu, предоставляющие как средства поддержки коммуникации, так и офисные приложения, такие, как электронная почта, электронные таблицы, приложения для обработки текстов и т. п..

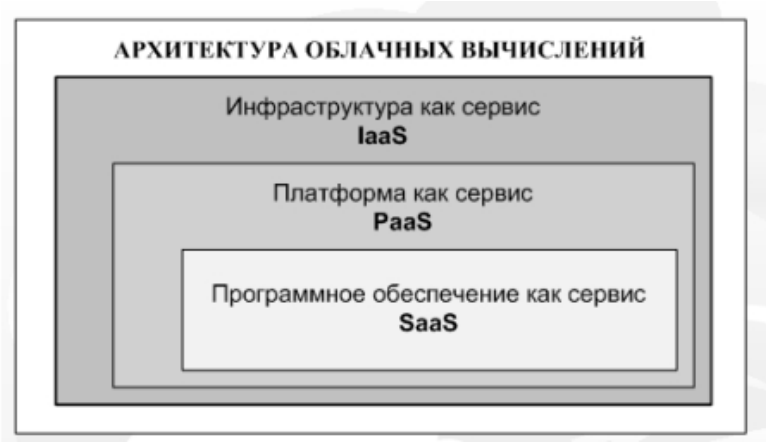


Рис.1. Архитектура облачных вычислений

Рассмотрим модели облаков с целью выявления возможности их применения в образовательном процессе. В настоящее время в мировой практике реализуются четыре модели развертывания облачных систем:− частное облако (private cloud) – используется для предоставления сервисов в одной организации. Она может включать несколько потребителей, например: подразделения организации, расположенные в разных зданиях, ее клиенты и подрядчики. Публичное облако (public cloud) – вычислительная инфраструктура, которая предназначена для свободного использования самым широким кругом пользователей, включая физических и юридических лиц. Публичным облаком могут порознь или совместно владеть или управлять (в том числе и эксплуатировать) государственные, коммерческие, научные организации. Публичное облако обычно находится под юрисдикцией его владельца – поставщика услуг. Гибридное облако (hybrid cloud) – это комбинация из двух или более различных облачных инфраструктур (частных, коммунальных или публичных), каждая из которых остается уникальным объектом.гибридное облако сочетает в себе преимущества частного и публичного облаков. Коммунальное облако (community cloud) – вид вычислительной инфраструктуры, предназначенный для использования конкретным сообществом потребителей (организаций), имеющих общие задачи. Примерами коммунальных облаков является платформа Windows Azure, веб-сервисы Amazon, Google App Engine и

Force.com [3]. Для образовательных учреждений наиболее подходящими являются публичные и коммунальные облака.

Сегодня облачные вычисления перестали быть образами из будущего. Несмотря на относительную новизну облачных технологий (первый проект был реализован в 1999 г.), уже накоплен опыт их применения в образовательном процессе учебных заведений разных стран и уровней. К использованию данных технологий переходят некоторые зарубежные образовательные учреждения. В Литве Каунасский технологический университет (Kaunas University of Technology) в течение трех последних лет использует облачные сервисы, предоставляемые Microsoft Live@edu. В США целые штаты переходят на использование облачных технологий. Так, в университете Хофстра (Hofstra University) используют облачные сервисы, предоставляемые Google Apps. Также университету была предоставлена возможность поддержки электронной почты для своих студентов и преподавателей. Еще одним вариантом использования облачных сервисов, который начинает распространяться в сфере образования, является перемещение в облако систем управления обучением (Learning Management Systems, LMS). Передача поддержки внешним провайдерам LMS (Blackboard, Moodle и т. д.) имеет смысл для образовательных учреждений, которые не могут позволить себе покупку и поддержку дорогостоящего оборудования и программного обеспечения.

Корпорация Microsoft начала широкое распространение облачного сервиса Office 365 в образовательных учреждениях нашей страны. Он включает в себя облачную версию Microsoft Office (Outlook, Word, Excel, Power Point, OneNote Web Apps), а также инструменты для совместной работы (Lync Online, SharePoint Online и Exchange Online). Внедрение этой разработки в Финансовом университете позволило расширить возможности мобильной работы, увеличит надежность и безопасность системы. Студентам и сотрудникам стало проще работать совместно с коллегами, получать доступ к ресурсам и сервисам университета из любого места и в любое время, используя различные устройства (компьютеры, планшеты, смартфоны). Количество пользователей Office 365 в университете на данный момент составляет 95000 человек, включая студентов и работников вуза.

Проанализировав опыт применения облачных вычислений, можно сделать вывод, что чаще всего образовательные учреждения

используют модель облака «программное обеспечение как сервис». Использование этой модели не требует от образовательного учреждения создания собственного сервера и его обслуживания, позволяет избежать экономических и организационных затрат и дает возможность устанавливать собственные приложения на платформе, предоставляемой провайдером услуги.

Можно выделить следующие преимущества использования облачных технологий в образовательном процессе:

- Экономические (основным преимуществом для многих образовательных учреждений является экономичность). Это особенно заметно, когда услуги, подобные электронной почте, бесплатно предоставляются внешними провайдерами. Оборудование для этих услуг может использоваться для других целей или ликвидироваться. Помещения освобождаются, что является актуальным в условиях, когда все чаще ощущается недостаток учебных аудиторий;

- Технические (минимальные требования к аппаратному обеспечению – обязательным условием является лишь наличие доступа к сети Интернет);

- Технологические (большинство облачных услуг высокого уровня достаточно просты в использовании, либо требуют минимальной поддержки);

- Дидактические (широкий спектр онлайн-инструментов и услуг, которые обеспечивают безопасное соединение и возможности сотрудничества преподавателей и студентов).

Можно выделить и некоторые недостатки облачных технологий, которые носят в основном технический и технологический характер и не влияют на их дидактические возможности и преимущества. К таким недостаткам можно отнести ограничение использования функциональных возможностей программного обеспечения по сравнению с локальными аналогами, отсутствие отечественных провайдеров облачных сервисов (Amazon, Goggle, Saleforce и др. сосредоточены в США), отсутствие отечественных и международных стандартов, а также отсутствие законодательной базы применения облачных технологий.

Распространению облачных вычислений препятствует ряд объективных факторов. Традиционно большинство отечественных образовательных учреждений с недоверием относятся к аренде виртуальных мощностей, предпочитая работать с конкретным, желательно собственным, оборудованием, программным обеспечением и данными, которые хранятся локально и доступны в любой момент времени.

Выводы: Облачные технологии предлагают альтернативу традиционным формам организации учебного процесса, создавая возможности для персонального обучения, интерактивных занятий и коллективного преподавания. Внедрение облачных технологий снизит затраты на приобретение необходимого программного обеспечения, повысит качество и эффективность образовательного процесса. Распространение облачных вычислений ставит перед образовательной средой задачи интеграции облачных сервисов в систему образовательного учреждения, пересмотра своей ИТ-инфраструктуры и внедрения инновационных технологий в образовательный процесс.

2. Процессы в облачных технологиях

2.1. Предмет теории процессов

Теория математического моделировани процессов является одним из разделов математической теории программирования, который изучает математические модели поведения динамических систем, называемые **процессами.** Говоря неформально, **процесс** представляет собой модель такого поведения, которое заключается в **исполнении действий.** Такими действиями могут быть, например,
- приём или передача каких-либо объектов, или
- преобразование этих объектов.

Основные достоинства **теории математического моделировани процессов как математического аппарата, предназначенного для моделирования и анализа динамических систем,** заключаются в следующем.

1. Аппарат теории математического моделировани процессов хорошо подходит для формального описания и анализа поведения **распределённых динамических систем,** т.е. таких систем, которые состоят из нескольких взаимодействующих компонентов, причем

• все эти компоненты работают параллельно, и
• взаимодействие компонентов происходит путём пересылки сигналов или сообщений от одних компонентов другим компонентам.

Важнейшим примером распределенных динамических систем являются программно-аппаратные вычислительные комплексы, в которых

• один класс компонентов определяется совокупностью компьютерных программ, которые функционируют в этом комплексе
• другой класс компонентов связан с аппаратной платформой, на базе которой функционирует этот комплекс
• третий класс компонентов представляет собой совокупность информационных ресурсов (базы данных, базы знаний, базы услуг, электронные библиотеки, и т.п.), которые используются для обеспечения функционирования этого комплекса
• также может приниматься во внимание класс компонентов, связанных с человеческим фактором.

2. Методы теории процессов позволяют анализировать с приемлемой сложностью модели с очень большим и даже бесконечным множеством состояний. Это возможно благодаря разработанной в теории математического моделировани процессов технике символьных преобразований выражений, описывающих процессы.

Важнейшим примером моделей с бесконечным множеством состояний являются модели компьютерных программ с переменными, множества значений которых имеют очень большой размер. В таких моделях программ в целях удобства проведения рассуждений большие множества значений некоторых переменных заменяются на соответствующие бесконечные множества. Например, множество значений переменных типа double, представляющее собой конечную (но очень большую) совокупность действительных чисел, может быть заменено на бесконечное множество всех действительных чисел.

В некоторых случаях представление анализируемой программы в виде модели с бесконечным множеством состояний существенно упрощает проведение рассуждений об этой программе. Анализ модели этой программы с конечным, но очень большим множеством состояний классическими методами теории автоматов, которые основаны на

• явном представлении множества состояний этой модели, и
• анализе этой модели путём явного оперирования с её состояниями

может иметь очень высокую вычислительную сложность, и в некоторых случаях замена задачи анализа исходной конечной модели на задачу анализа соответствующей бесконечной модели такими методами, которые основаны на символьных преобразованиях

выражений, описывающих эту модель, может дать существенный выигрыш с точки зрения вычислительной сложности.
3. Методы теории математического моделировани процессов хорошо подходят для изучения **иерархических** систем, т.е. таких систем, которые имеют многоуровневую структуру.
Каждая компонента таких систем рассматривается как подсистема, которая, в свою очередь, может состоять из нескольких подкомпонентов. Каждая из этих подкомпонентов может взаимодействовать
- с другими подкомпонентами, и
- с системами более высокого уровня.

Основными источниками задач и объектами применения результатов теории процессов являются распределенные компьютерные системы. Вместе с тем, теория процессов может использоваться также для моделирования и анализа поведения систем самой различной природы, важнейшими примерами которых являются **организационные системы.** К их числу относятся
- системы управления деятельностью предприятий,
- государственные структуры,
- системы организации коммерческих процессов (например, системы организации торговых сделок, аукционов, и т.п.)

Процессы, относящиеся к функционированию таких систем, принято называть **бизнес-процессами.**

2.2 Верификация процессов

Наиболее важный класс задач, для решения которых предназначена теория процессов, связан с проблемой верификации процессов. Проблема **верификации** процесса заключается в построении формального доказательства того, что анализируемый процесс обладает заданными свойствами.
Для многих процессов данная задача представляет исключительную актуальность. Например, безопасная эксплуатация таких систем, как
- системы управления атомными электростанциями,
- медицинские устройства с компьютерным управлением,
- бортовые системы управления самолетов и космических аппаратов,
- системы управления секретными базами данных,
- системы электронной коммерции

невозможна без удовлетворительного решения задачи верификации свойств корректности и безопасности процессов, функционирующих в таких системах, так как нарушения данных свойств в системах

подобного типа могут привести к существенному ущербу для экономики и самой жизни людей.

Точная постановка задачи верификации состоит из следующих частей.

1. Построение процесса, представляющего собой математическую модель поведения анализируемой системы.
2. Представление проверяемого свойства в виде математического объекта (называемого **спецификацией).**
3. Построение **математического доказательства** утверждения о том, что построенный процесс удовлетворяет спецификации.

2.3. Спецификация процессов

Спецификация процесса представляет собой описание свойств этого процесса в виде некоторого математического объекта.

Примером спецификации является требование надежности передачи данных через ненадежную среду. При этом не указывается, как именно должна обеспечиваться эта надежность.

В качестве спецификации могут выступать, например, следующие объекты.

1. Логическая формула, выражающая некоторое требование к процессу.

Например, таким требованием может быть условие того, что если процесс получил некоторый запрос, то через некоторое заданное время процесс выдаст ответ на этот запрос.

2. Представление анализируемого процесса на более высоком уровне абстракции.

Данный вид спецификаций используется при многоуровневом проектировании процессов: реализацию процесса на каждом уровне проектирования можно рассматривать как спецификацию для реализации этого процесса на следующем уровне проектирования.

3. Некоторый эталонный процесс, относительно которого предполагается, что он обладает заданным свойством.

В этом случае задача верификации заключается в построении доказательства эквивалентности эталонного и анализируемого процессов.

При построении спецификаций следует руководствоваться следующими принципами.

1. Одно и то же свойство процесса может быть выражено на разных языках спецификаций (ЯС), и

- на одном ЯС оно может иметь простую спецификацию,
- а на другом - сложную.

Например, спецификация, описывающая связь между входными и выходными значениями для программы, вычисляющей разложение целого числа на простые множители, имеет
• сложный вид на языке логики предикатов, но
• простой вид, если выразить эту спецификацию в виде некоторой эталонной программы.
Поэтому для представления свойства процесса в виде спецификации важно выбрать такой ЯС, на котором спецификация этого свойства имела бы наиболее ясный и простой вид.
2. Если свойство процесса изначально было выражено на естественном языке, то при переводе его в спецификацию важно обеспечить соответствие между
• естественно-языковым описанием этого свойства, и
• его спецификацией, т.к. в случае несоблюдения этого условия результаты верификации не будут иметь смысла.

2.4. Понятие процесса

2.4.1 Представление поведения динамических систем в виде процессов

Один из возможных методов математического моделирования поведения динамических систем заключается в представлении поведения этих систем в виде **процессов.**
Процесс, как правило, не учитывает всех деталей поведения анализируемой системы. Одно и то же поведение может быть представлено различными процессами, отражающими
• разную степень абстракции при построении модели этого поведения, и
• разные уровни детализации действий, исполняемых системой.
Если целью построения процесса для представления поведения некоторой системы является проверка свойств этого поведения, то выбор уровня детализации действий системы должен производиться с учётом тех свойств, которые необходимо проанализировать.
Построение процесса, представляющего поведение анализируемой системы, должно производиться с учётом следующих принципов.
1. Описание процесса не должно быть чрезмерно детальным, т.к. излишняя сложность этого описания может вызвать существенные вычислительные проблемы при формальном анализе этого процесса.
2. Описание процесса не должно быть чрезмерно упрощённым, оно должно

• отражать те аспекты поведения моделируемой системы, которые имеют отношение к проверяемым свойствам, и
• сохранять все свойства поведения этой системы, которые представляют интерес для анализа
т.к. в случае несоблюдения этого условия результаты анализа такого процесса не будут иметь смысла.

2.4.2 Неформальное понятие процесса и примеры процессов

Прежде чем сформулировать точное определение процесса, мы приведём неформальное понятие процесса, и рассмотрим простейшие примеры процессов.

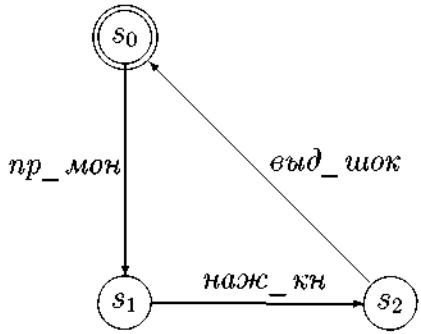
Неформальное понятие процесса

Как было сказано выше, мы понимаем под процессом модель поведения динамической системы на некотором уровне абстракции.
Процесс можно представлять себе как граф *P,* компоненты которого имеют следующий смысл.
• Вершины графа *P* называются **состояниями,** и изображают ситуации (или классы ситуаций), в которых может находиться моделируемая система в различные моменты своего функционирования.
Одно из состояний является выделенным, оно называется **начальным состоянием** процесса *P.*
• Рёбра графа *P* имеют метки, изображающие **действия,** которые может исполнять моделируемая система.
• Функционирование процесса *P* описывается переходами по рёбрам графа *P* от одного состояния к другому. Функционирование начинается из начального состояния.
Метка каждого ребра изображает действие процесса, исполняемое при переходе от состояния в начале ребра к состоянию в его конце.

Пример процесса

В качестве первого примера рассмотрим процесс, представляющий собой простейшую модель поведения некоторого торгового автомата.
Мы будем представлять себе этот автомат как машину, которая имеет
• монетоприемник,
• кнопку, и
• лоток для выдачи товара.
Когда покупатель хочет приобрести товар, он
• опускает монету в монетоприемник,
• нажимает на кнопку

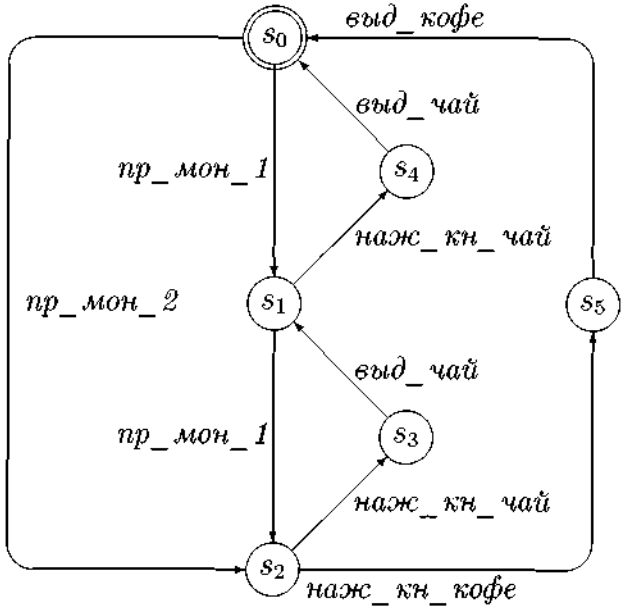
и после этого в лотке появляется товар.
Предположим, что наш автомат торгует шоколадками по цене 1 монета за штуку.
Опишем действия такого автомата.
• По инициативе покупателя, в автомате могут происходить следующие действия:
— попадание в щель монеты, и
— нажатие кнопки.
• В ответ автомат может осуществлять реакцию: выдавать в лоток шоколадку.
Обозначим действия короткими именами:
• прием монеты мы обозначим символом *пр_мон,*
• нажатие кнопки - символом *паж,_ кн,* и
• выдачу шоколадки - символом *выд_ шок.*
Процесс нашего торгового автомата выглядит следующим образом:



Данная диаграмма объясняет, как именно функционирует торговый автомат:
• вначале автомат находится в состоянии s_0, в этом состоянии он ожидает появления в приемнике монеты
(то, что состояние s_0 является начальным, изображается на диаграмме двойным кружочком вокруг идентификатора этого состояния)
• когда монета появляется, автомат переходит в состояние s_1 и ждет нажатия на кнопку,
• после нажатия кнопки автомат
— переходит в состояние S_2,
— выдает шоколадку, и
— возвращается в состояние s_0.

Другой пример процесса

Рассмотрим более сложный пример торгового автомата, который отличается от предыдущего тем, что продаёт два вида товаров: чай и кофе, причём стоимость чая - 1 рубль, а стоимость кофе -2 рубля. Автомат имеет две кнопки: одну - для чая, другую - для кофе. Покупатель может платить монетами достоинством в 1 рубль и 2 рубля. Данные монеты будут обозначаться знакосочетаниями *мон_ 1* и *мон_ 2* соответственно.
Если покупатель опустил в монетоприемник монету *мон_1,* он может купить только чай. Если же он опустил монету *мон_2,* он может купить кофе или два чая. Кофе можно купить также, опустив в монетоприёмник две монеты *мон_1.*
Процесс такого торгового автомата выглядит следующим образом:



Для формального определения понятия процесса мы должны уточнить понятие действия. Это уточнение излагается дальше.

2.4.3.Действия

Для задания процесса *P,* представляющего собой модель поведения некоторой динамической системы, должно быть указано некоторое множество *Act(P)* **действий,** которые может выполнять процесс *P.*
Мы будем предполагать, что действия всех процессов являются элементами некоторого универсального множества *Act* всех возможных действий, которые может выполнить какой-либо процесс, т.е. для любого процесса *P*

$$Act(P) \subseteq Act$$

Выбор множества *Act(P)* действий процесса *P* зависит от целей моделирования. В разных ситуациях для представления модели анализируемой системы в виде некоторого процесса могут выбираться разные множества действий.
Мы будем предполагать, что множество действий *Act* делится на 3 следующих класса.
1. Входные действия, которые изображаются знакосочетаниями вида α?
Действие вида α? интерпретируется как ввод в процесс некоторого объекта с именем *α.*
2. **Выходные действия,** которые изображаются знакосочетаниями вида
α!
Действие вида *α!* интерпретируется как вывод из процесса некоторого объекта с именем *a.*
3. **Внутреннее** (или **невидимое)** действие, которое обозначается символом τ.
Внутренним мы называем такое действие процесса *P,* которое не связано с его взаимодействием с **окружающей средой** (т.е. с процессами, являющимися внешними по отношению к процессу *P,* и с которыми он может взаимодействовать) .
Например, внутреннее действие может быть связано с взаимодействием компонентов процесса *P.*
В действительности, внутренние действия могут быть самыми разнообразными, но для обозначения всех внутренних действий мы будем использовать один и тот же символ τ. Это отражает наше желание не различать все внутренние действия, т.к. они не являются "наблюдаемыми" извне процесса *P.*
Обозначим знакосочетанием *Names* совокупность имён объектов, которые могут вводиться в какой-либо процесс нли выводиться из него.

Мы не накладываем никаких ограничений на виды объектов, с которыми могут оперировать процессы, поэтому множество *Names* предполагается бесконечным.
Множество *Act* по определению представляет собой дизъюнкное объединение

$$\begin{aligned} Act \;=\; & \{\alpha? \mid \alpha \in Names\} \;\cup \\ \cup\; & \{\alpha! \mid \alpha \in Names\} \;\cup \\ \cup\; & \{\tau\} \end{aligned} \qquad (2.1)$$

Отметим, что объекты, которые вводятся в процесс и выводятся из него, могут иметь самую различную природу (как материальную, так и не материальную). Например, ими могут быть
- материальные ресурсы,
- люди,
- деньги,
- информация,
- энергия,
- и т.д.

Кроме того, сами понятия ввода и вывода могут иметь виртуальный характер, т.е. слова "ввод" и "вывод" могут использоваться лишь как метафоры, а в действительности никакого ввода или вывода какого-либо реального объекта может и не происходить. Говоря неформально, мы будем рассматривать действие процесса *P* как
- **входное,** если его инициатором является процесс, внешний по отношению к *P,* и
- **выходное,** если оно не является внутренним, и его инициатором является сам процесс *P.*

Для каждого имени $\alpha \in Names$ действия $\alpha?$ **и** $\alpha!$ называются **комплементарными**.
Мы будем использовать следующие обозначения.
1. Для каждого действия $a \in Act \setminus \{\tau\}$ знакосочетание *α* обозначает действие, комплементарное к *α,* т.е.

$$\overline{\alpha?} \stackrel{\text{def}}{=} \alpha!, \quad \overline{\alpha!} \stackrel{\text{def}}{=} \alpha?$$

2. Для каждого действия $a \in Act \setminus \{\tau\}$ знакосочетание *name (α)* обозначает имя, указанное в действии *α,* т.е.

$$name(\alpha?) \stackrel{\text{def}}{=} name(\alpha!) \stackrel{\text{def}}{=} \alpha$$

3. Для каждого подмножества $L \subseteq Act \setminus \{\tau\}$

- $\overline{L} \stackrel{\text{def}}{=} \{\overline{a} \mid a \in L\}$
- $names(L) \stackrel{\text{def}}{=} \{name(a) \mid a \in L\}$

2.4.4. Определение понятия процесса

Процессом называется тройка *P* вида

$$P = (S, s^0, R) \quad (2.2)$$

компоненты которой имеют следующий смысл.

• *S* - множество, элементы которого называются **состояниями** процесса *P.*

• $s^0 \in S$ - некоторое выделенное состояние, называемое **начальным состоянием** процесса *P.*

• *R* - подмножество вида

$$R \subseteq S \times Act \times S$$

Элементы множества *R* называются **переходами.** Если переход из *R* имеет вид (s_1, a, s_2), то

— мы будем говорить, что этот переход является переходом из состояния s_1 в состояние S_2 с выполнением действия *α,*

— состояния S_1 и S_2 называются **началом** и **концом** этого перехода соответственно, а действие *α* называется **меткой** этого перехода, и

— иногда, в целях повышения наглядности, мы будем обозначать данный переход знакосочетанием

$$s_1 \xrightarrow{a} s_2 \quad \textbf{(2.3)}$$

Функционирование процесса $P = (S, s^0, R)$ заключается в порождении последовательности переходов вида

$$s^0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \ldots$$

и выполнении действий $a_0, a_1, a_2 \ldots$, соответствующих этим переходам.

Более подробно: на каждом шаге функционирования $i \geq 0$

• процесс находится в некотором состоянии s_i $(s_0 = s^0)$,

• если есть хотя бы один переход из *R* с началом в s_i, то процесс

— недетерминированно выбирает переход с началом в s_i, помеченный таким действием a_i, которое можно выполнить в текущий момент времени,

(если таких переходов нет, то процесс временно приостанавливает свою работу до того момента, когда появится хотя бы один такой переход)
— выполняет действие a_i, и после этого
— переходит в состояние s_{i+1}, которое является концом выбранного перехода
• если в *R* нет переходов с началом в s_i, то процесс заканчивает свою работу.
Знакосочетание *Act(P)* обозначает множество всех действий из $Act \setminus \{\tau\}$, которые могут быть выполнены процессом *P,* т.е.

$$Act(P) \stackrel{\text{def}}{=} \{a \in Act \setminus \{\tau\} \mid \exists\, (s_1 \xrightarrow{a} s_2) \in R\}$$

Процесс (2.2) называется **конечным,** если его компоненты *S* и *R* являются конечными множествами.
Конечный процесс можно изображать геометрически, в виде диаграммы на плоскости, в которой
• каждому состоянию соответствует некоторый кружочек на плоскости, в котором может быть написан идентификатор, представляющий собой имя этого состояния,
• каждому переходу соответствует стрелка, соединяющая начало этого перехода и его конец, причём на стрелке написана метка этого перехода,
• начальное состояние выделяется некоторым образом (например, вместо обычного кружочка рисуется двойной кружочек) .
Примеры таких диаграмм содержатся ниже.

2.4.5 Понятие трассы

Пусть $P = (S, s^0, R)$ - некоторый процесс.
Трассой процесса *P* называется конечная или бесконечная последовательность

$$a_1, a_2, \ldots$$

элементов множества *Act,* такая что существует последовательность состояний процесса *P*

$$s_0, s_1, s_2, \ldots$$

обладающая следующими свойствами:
• s_0 совпадает с начальным состоянием s^0 процесса *P*
• для каждого $i \geq 1$ множество *R* содержит переход

$$s_i \xrightarrow{a_i} s_{i+1}$$

Множество всех трасс процесса *P* мы будем обозначать через $Tr(P)$.

2.4.6 Достижимые и недостижимые состояния

Пусть *P* - процесс вида (2.2).
Состояние *s* процесса *P* называется достижимым, если $s = s^0$ или существует последовательность переходов в *P,* имеющая вид

$$s_0 \xrightarrow{a_1} s_1\,, \quad s_1 \xrightarrow{a_2} s_2\,, \quad \ldots \quad s_{n-1} \xrightarrow{a_n} s_n$$

в которой $n \geq 1,\ s_0 = s^0$ **и** $s_n = s$.
Состояние называется **недостижимым,** если оно не является достижимым.
Нетрудно видеть, что после после того, как
- из *S* будут удалены недостижимые состояния, и
- из R будут удалены переходы, в которых присутствуют недостижимые состояния,

получившийся процесс *P′* (который иногда называют **достижимой частью** процесса *P)* будет представлять точно такое же поведение, которое представлял исходный процесс. По этой причине мы будем рассматривать такие процессы *P* и *P′* как одинаковые.

2.4.7 Замена состояний

Пусть
- *P* - процесс вида (2.2),
- *s* - некоторое состояние из *S*
- *s′* - произвольный элемент, не принадлежащий множеству *S.*

Обозначим символом *P′* процесс, который получается из *P* заменой *s* на *s′* в множествах *S* и *R,* т.е., в частности, каждый переход в *P* вида

$$s \xrightarrow{a} s_1 \quad \textbf{ИЛИ} \quad s_1 \xrightarrow{a} s$$

заменяется на переход

$$s' \xrightarrow{a} s_1 \quad \textbf{или} \quad s_1 \xrightarrow{a} s'$$

соответственно.
Как и в предыдущем параграфе, нетрудно видеть, что *P′* будет представлять точно такое же поведение, которое представлял *P,* и по этой причине мы можем рассматривать такие процессы *P* и *P′* как одинаковые.

Также отметим, что заменять можно не одно состояние, а произвольное подмножество состояний процесса *P.* Такую замену можно представить как задание взаимно однозначного отображения

$$f : S \to S' \quad (2.4)$$

и результатом такой замены по определению является процесс *P′* вида

$$P' = (S', (s')^0, R') \quad (2.5)$$

где

- $(s')^0 \stackrel{\text{def}}{=} f(s^0)$, (
- для каждой пары $s_1, s_2 \in S$ и каждого $a \in Act$

$$(\, s_1 \xrightarrow{a} s_2 \,) \in R \quad \Leftrightarrow \quad (\, f(s_1) \xrightarrow{a} f(s_2) \,) \in R'.$$

Поскольку такие процессы *P* и *P′* представляют одинаковое поведение, мы можем рассматривать их как одинаковые.
Отметим, что в литературе такие процессы *P* и *P′* иногда называют не одинаковыми, а **изоморфными.** Отображение (2.4) с указанными выше свойствами называют **изоморфизмом** между *P* и *P′.* Процесс *P′* называют **изоморфной копией** процесса *P.*

3. Операции на процессах

В этой главе мы определим некоторые операции на процессах, при помощи которых из одних процессов мы сможем строить другие, более сложные процессы.

3.1 Префиксное действие

Первая такая операция - префиксное действие. Пусть заданы

- процесс $P = (S, s^0, R)$, и
- действие $a \in Act.$

Действие операции **префиксного действия** *a.* на процесс *P* заключается в том, что

- к множеству состояний *P* добавляется новое состояние *s,* которое будет начальным состоянием нового процесса, и
- к множеству переходов добавляется переход

$$s \xrightarrow{a} s^0$$

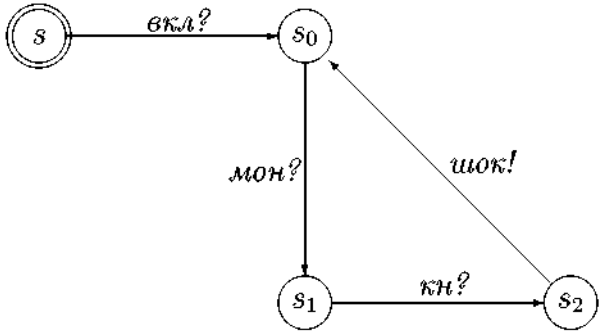
Получившийся процесс обозначается знакосочетанием

$$a.P$$

Проиллюстрируем действие данной операции на примере торгового автомата. Обозначим процесс, представляющий поведение этого автомата, символом $P_{та}$.
Расширим множество действий данного автомата новым входным действием *вкл*?, которое будет означать включение этого автомата в сеть.
Процесс *вкл?*. $P_{та}$ представляет поведение нового торгового автомата, который в начальном состоянии не может

- ни принимать монет,
- ни воспринимать нажатия на кнопку,
- ни выдавать шоколадок.

Единственное, что он может - это стать включенным. После этого его поведение ничем не будет отличаться от поведения исходного автомата.
Графовое представление процесса *вкл*?. $P_{та}$ выглядит следующим образом:

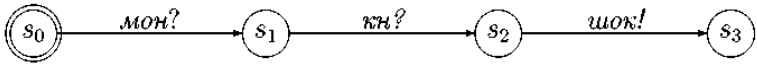


3.2 Пустой процесс

Среди всех процессов существует один наиболее простой. Этот процесс имеет всего одно состояние, и не имеет переходов. Для обозначения такого процесса мы будем использовать константу (т.е. нульарную операцию) **0**.
Возвращаясь к примерам с торговыми автоматами, можно сказать, что процесс **0** представляет поведение сломанного автомата, то есть такого автомата, который вообще не может ничего делать.
Путем применения операций префиксного действия к процессу **0** можно определять поведение более сложных автоматов. Рассмотрим, например, такой процесс:

$$P = мон?.кн?.шок!.\mathbf{0}$$

Графовое представление этого процесса выглядит следующим образом:



Этот процесс задает поведение автомата, который обслуживает ровно одного покупателя, и после этого ломается.

3.3 Альтернативная композиция

Следующая операция на процессах - это бинарная операция альтернативной композиции.
Данная операция используется в том случае, когда по паре процессов P_1 и P_2, надо построить процесс P, который будет функционировать
- либо как процесс P_1,
- либо как процесс P_2,

причём выбор процесса, в соответствии с которым P будет функционировать, может определяться
- как самим P,
- так и окружающей средой, в которой функционирует P.

Например, если P_1 и P_2 имеют вид

$$P_1 = \alpha? \,.\, P_1' \qquad P_2 = \beta? \,.\, P_2' \qquad (3.1)$$

и в начальный момент времени окружающая среда
- может ввести в P объект α, но
- не может ввести в P объект β

то P должен выбрать то поведение, которое является единственно возможным в данной ситуации, т.е. работать так же, как процесс P_1.
Отметим, что в данном случае выбирается такой процесс, первое действие в котором может быть выполнено в текущий момент времени.
Выбрав P_1, и выполнив действие α?, процесс P обязан продолжать работу в соответствии со своим выбором, т.е. в соответствии с процессом P'_1. Не исключено, что после выполнения действия α?
- будет невозможно выполнить ни одного действия, работая в соответствии с процессом P'_1
- хотя в этот момент появится возможность выполнить первое действие в P_2.

Но в этот момент процесс P уже не может изменить свой выбор (т.е. выбрать P'_2 вместо P'_1). Процесс P может лишь находиться в состоянии

ожидания того, когда появится возможность работать в соответствии с процессом P'_1.

Если же в начальный момент времени окружающая среда может ввести в P как α, так и β, то P выбирает процесс, в соответствии с которым он будет работать,

- недетерминированно (т.е. произвольно), или
- с учётом некоторых дополнительных факторов.

Точное определение операции альтернативной композиции выглядит следующим образом.

Пусть процессы P_1 и P_2 имеют вид

$$P_i = (S_i, s_i^0, R_i) \quad (i = 1, 2)$$

причём множества состояний S_1 и S_2 не имеют общих элементов.

Альтернативной композицией процессов P_1 и P_2 называется процесс

$$P_1 + P_2 = (S, s^0, R)$$

компоненты которого определяются следующим образом.

- S получается добавлением к $S_1 \cup S_2$ нового состояния s^0, которое будет начальным состоянием процесса $P_1 + P_2$
- R содержит все переходы из R_1 и R_2.
- для каждого перехода из R_i ($r = 1, 2$) вида

$$s_i^0 \xrightarrow{a} s$$

R содержит переход

$$s^0 \xrightarrow{a} s$$

Если же множества S_1 и S_2 имеют общие элементы, то для определения процесса $P_1 + P_2$ сначала надо заменить в S_2 те состояния, которые входят также и в S_1, на новые элементы, а также модифицировать соответствующим образом R_2 и s^0_2.

Рассмотрим в качестве примера торговый автомат, который продаёт газированную воду, причём

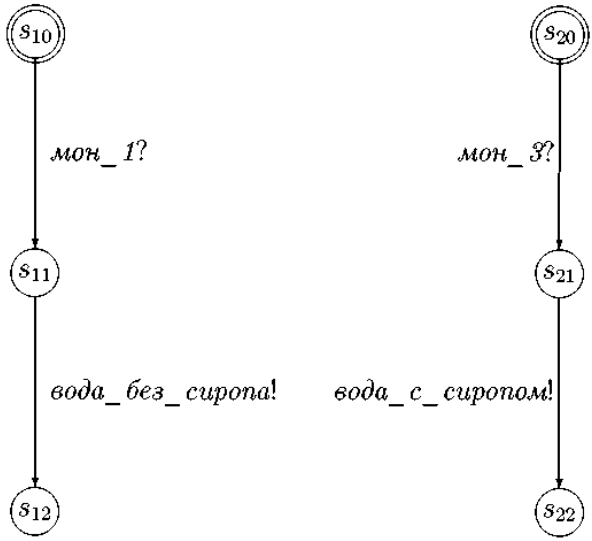
- если покупатель опускает в него монету *мон_ 1* достоинством в 1 копейку, то автомат выдаёт стакан воды без сиропа,
- а если покупатель опускает в него монету *мон_ 3* достоинством в 3 копейки, то автомат выдаёт стакан воды с сиропом

причём сразу после продажи одного стакана воды автомат ломается.

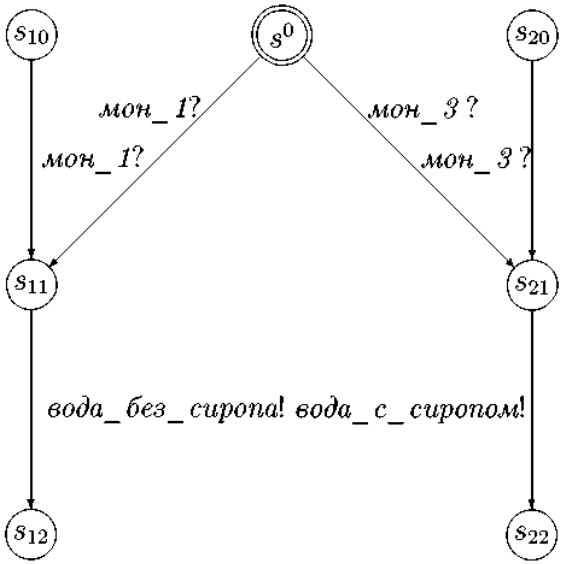
Поведение данного автомата описывается следующим процессом:

$$P_{\text{газ\_вода}} = \textit{мон\_1}\,?\,.\,\textit{вода\_без\_сиропа}\,!\,.\,\mathbf{0} + \textit{мон\_3}\,?\,.\,\textit{вода\_с\_сиропом}\,!\,.\,\mathbf{0} \quad \textbf{(3.2)}$$

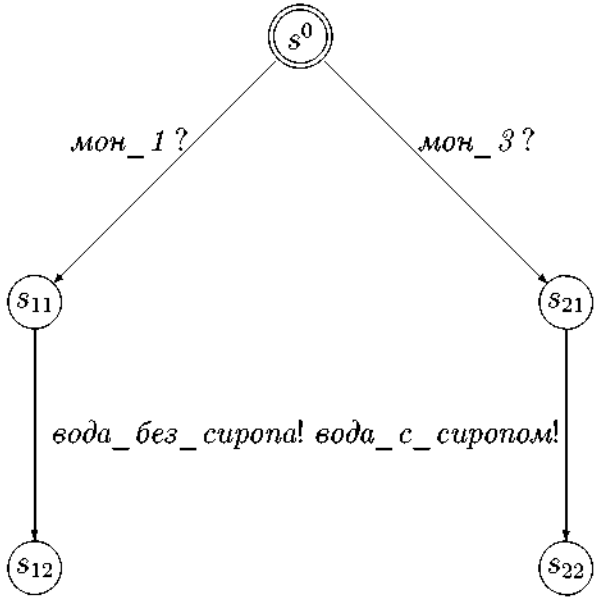
Рассмотрим графовое представление процесса (3.2). Графовые представления слагаемых в сумме (3.2) имеют вид



Согласно определению альтернативной композиции, графовое представление процесса (3.2) получается добавлением к предыдущей диаграмме нового состояния и соответствующих переходов, в результате чего получается следующая диаграмма:



Поскольку состояния s_{10} и s_{20} недостижимы, то, следовательно, их (а также связанные с ними переходы) можно удалить, в результате чего получится диаграмма



которая и является искомым графовым представлением процесса (3.2).
Рассмотрим другой пример. Опишем разменный автомат, в который можно вводить купюры достоинством в 1000 рублей. Автомат должен выдать

- либо 2 купюры по 500 рублей,
- либо 10 купюр по 100 рублей

причем выбор способа размена осуществляется независимо от желания клиента. Сразу после одного сеанса размена автомат ломается.

$$P_{\text{размен}} = \\ = 1\_no\_1000?.(2\_no\_500!.0 + 10\_no\_100!.0)$$

На этих двух примерах видно, что альтернативная композиция может использоваться для описания как минимум двух принципиально различных ситуаций.

1. Во-первых, она может выражать зависимость поведения системы от поведения её окружения.

Например, в случае автомата $P_{\text{газ вода}}$, реакция автомата определяется действием покупателя, а именно, достоинством монеты, которую он ввел в автомат.

В данном случае процесс, представляющий поведение моделируемого торгового автомата, является **детерминированным,** то есть его фукнционирование однозначно определяется входными действиями.

2. Во-вторых, на примере автомата $P_{\text{размен}}$ мы видим, что при одних и тех же входных действиях возможна различная реакция автомата.

Это - пример **недетерминизма,** то есть неопределенности поведения системы.

Неопределённость в поведении систем может происходить по крайней мере по двум причинам.

(a) Во-первых, поведение систем может зависеть от **случайных факторов.**

Такими факторами могут быть, например,

- сбои в аппаратуре,
- коллизии в компьютерной сети
- отсутствие купюр необходимого достоинства в банкомате
- или что-либо еще

(b) Во-вторых, модель всегда есть некоторая абстракция или упрощение реальной системы. А при этом некоторые факторы, влияющие на поведение этой системы, могут быть просто исключены из рассмотрения.

В частности, на примере процесса $P_{размен}$ мы видим, что реальная причина выбора варианта поведения автомата может не учитываться в процессе, представляющем собой модель поведения этого автомата. Можно схематически изобразить вышеперечисленные варианты использования альтернативной композиции следующим образом:

3.4 Параллельная композиция

Операция параллельной композиции используется для построения моделей поведения динамических систем, состоящих из нескольких взаимодействующих компонентов.

Прежде чем дать формальное определение этой операции, мы обсудим понятие параллельного функционирования двух систем Sys_1 и Sys_2, которые мы рассматриваем как компоненты одной системы *Sys*, т.е.

$$Sys \stackrel{\text{def}}{=} \{Sys_1, Sys_2\} \quad (3.3)$$

Пусть поведение систем Sys_1 и Sys_2 представлено процессами P_1 и P_2 соответственно. Поведение системы $Sys_i \ \ (i = 1, 2)$ в составе системы *Sys* описывается тем же процессом P_i, которым описывается поведение этой системы, рассматриваемой индивидуально.

Обозначим символом *{P1,P2}* процесс, описывающий поведение системы (3.3). Целью этого параграфа является явное определение процесса *{P1,P2}* (т.е. построение множеств его состояний и переходов) по информации о процессах P_1 и P_2.

Ниже для упрощения изложения мы будем отождествлять понятия

"процесс P", и

"система, поведение которой описывается процессом P"

Как было отмечено выше, функционирование произвольного процесса *P* может быть интерпретировано как обход графа, соответствующего этому процессу, с выполнением действий, которые являются метками проходимых рёбер.

Мыбудем предполагать, что при прохождении каждого ребра

$s \xrightarrow{a} s'$

• переход от S к S' происходит мгновенно, и

• факт выполнения действия *a* имеет место именно в момент этого перехода.

В действительности, выполнение каждого из действий происходит в течение некоторого промежутка времени, но мы будем считать, что для каждого проходимого ребра $s \xrightarrow{a} s'$

• до завершения выполнения действия *a* процесс *P* находится в состоянии *s*, и

• после завершения выполнения действия *a* процесс *P* мгновенно переходит в состояние *s'*.

Поскольку выполнение разных действий имеет разную продолжительность, то мы будем считать, что во время своего функционирования процесс *P* находится в каждом состоянии, в которое он попадает, неопределённый промежуток времени.

Таким образом, работа процесса *P* предсталяет собой чередование следующих двух видов деятельности:

• ожидание в течение неопределённого промежутка времени в одном из состояний, и

• мгновенный переход из одного состояния в другое.

Ожидание в одном из состояний может происходить

• не только по причине того, что в этот момент происходит выполнение некоторого действия,

• но также и по причине того, что процесс *P* в текущий момент просто не может выполнить какое-либо действие.

Например, если

• $P = \alpha?.P'$,и

• в начальный момент никто не предлагает процессу *P* объект α

то *P* будет ждать, когда какой-либо процесс предложит ему объект α.

Как мы знаем, для каждого процесса

• его действия являются либо входными, либо выходными, либо внутренними, и

• каждое входное и выходное действие является результатом взаимодействия этого процесса с другим процессом.

Каждое входное или выходное действие процесса P_i (i = 1,2) представляет собой

• либо результат взаимодействия P_i с процессом, не входящим в совокупность $\{P_1, P_2\}$,

•либо результат взаимодействия P_i с процессом P_j, где $j \in \{1,2\} \setminus \{i\}$.

С точки зрения процесса $\{P_1,P_2\}$, действия второго типа являются внутренними действиями этого процесса, так как они

• не представляют собой результат взаимодействия процесса $\{P_1,P_2\}$ с окружающей средой, а

• являются результатом взаимодействия компонентов этого процесса.

Таким образом, каждое действие процесса $\{P_1, P_2\}$ представляет собой

(a) либо результат взаимодействия одного из процессов P_i с процессом, не входящим в $\{P_1,P_2\}$,

(b) либо внутреннее действие одного из процессов, входящих в $\{P_1,P_2\}$ (т.е. внутреннее действие P_1 или P_2),

(c) либо внутреннее действие, которое является результатом взаимодействия процессов P_1 и P_2, и заключается в том, что

— один из этих процессов P_i (i = 1, 2) передаёт другому процессу P_j ($j \in \{1,2\} \setminus \{i\}$) некоторый объект, и

— процесс P_j в тот же самый момент времени принимает от процесса P_i этот объект

(такой вид взаимодействия называют **синхронным** взаимодействием, или **рукопожатием)**.

Каждому возможному варианту поведения процесса P_i (i = 1, 2) можно сопоставить **нить,** обозначаемую символом σ_i и представляющую собой вертикальную линию, на которой нарисованы точки с метками, где

• метки точек обозначают действия, исполняемые процессом P_i, и

• помеченные точки расположены в хронологическом порядке, т.е.

— сначала идёт точка, помеченная первым действием процесса P_i,

— под ней - точка, помеченная вторым действием процесса P_i,

— и т.д.

Для каждой помеченной точки p на нити мы будем обозначать знакосочетанием *act(p)* метку этой точки.

Представим себе, что на плоскости параллельно нарисованы нити

$\sigma_1 \quad \sigma_2$ (3.4)

где σ_i (i = 1, 2) представляет возможный вариант поведения процесса P_i в составе процесса *{P1, P2}*.

Рассмотрим те помеченные точки на нитях из (3.4), которые соответствуют действиям типа (c), т.е взаимодействиям процессов P_1 и P_2. Пусть p - одна из таких точек, и пусть она находится, например, на нити σ_1 .

Согласно определению понятия взаимодействия, в тот же самый момент времени, в который выполняется действие *act(p),* процесс P_2 выполняет комплементарное действие, т.е. на нити σ_2 есть точка p', такая, что

- $act(p') = \overline{act(p)}$, и
- действия $act(p)$ **и** $act(p')$ исполняются в один и тот же момент времени.

Отметим, что

- на нити σ_2 может быть несколько точек с меткой $\overline{act(p)}$,

но ровно одна из этих точек соответствует тому действию, которое исполняется совместно с действием, соответствующим точке *p,* и

- на нити σ_1 может быть несколько точек с меткой *act(p),* но ровно одна из этих точек соответствует тому действию, которое исполняется совместно с действием, соответствующим точке p'.

Преобразуем нашу диаграмму нитей (3.4) следующим образом: для каждой пары точек p, p' с вышеуказанными свойствами

- соединим точки p и p' стрелкой, начало которой - та из этих точек, которая имеет метку вида $\alpha!$, а конец - точка с меткой $\alpha?$,
- нарисуем на этой стрелке метку α, и
- заменим метки точек p и p' на τ.

Стрелка, соединяющая точки p и p', называется **синхронизационной стрелкой.** Такие стрелки обычно рисуют горизонтально, для чего точки на нитях располагают так, чтобы соединяемые стрелками точки располагались на одинаковой высоте.

После того, как мы сделаем такие преобразования для всех пар точек, в которых происходят действия вида (c), у нас получится диаграмма, которую в литературе по параллельным вычислениям принято называть **Message Sequence Chart (MSC).** Эта диаграмма представляет собой один из вариантов функционирования процесса $\{P_1,P_2\}$.

Мы будем обозначать знакосочетанием

$$Beh\{P_1, P_2\}$$

совокупность всех MSC, каждая из которых соответствует некоторому варианту функционирования процесса $\{P_1,P_2\}$.

Рассмотрим следующий пример: $\{P_1,P_2\}$ состоит из двух процессов P_1 и P_2, где

- P_1 - торговый автомат, поведение которого задается выражением

$$P_1 = мон?.\ шок!.\mathbf{0} \quad (3.5)$$

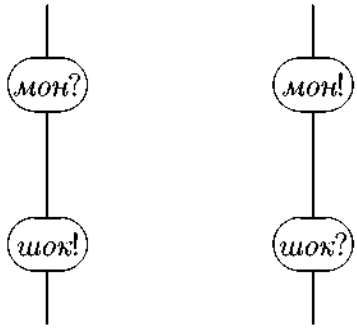
(т.е. он получает монету, выдаёт шоколадку, и после этого ломается)

- P_2 - покупатель, поведение которого задается выражением

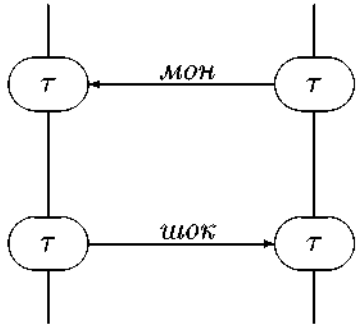
$$P_2 = мон!.\ шок?.\mathbf{0} \quad (3.6)$$

(т.е. он опускает монету, получает шоколадку, и после этого перестаёт функционировать в роли покупателя)

Нити этих процессов имеют вид



Если все действия на этих нитях являются действиями типа (c), то данная диаграмма преобразуется в следующую MSC:



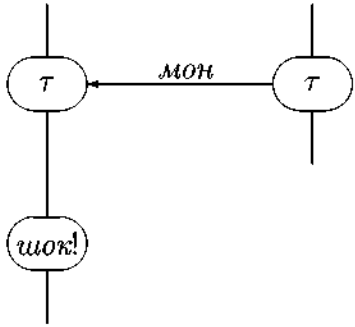
Однако возможен и следующий вариант функционирования процесса $\{ P_1, P_2\}$:

- первое действие P_1 и P_2 имеет тип (c), т.е. покупатель опускает в автомат монету, а автомат её принимает
- второе действие автомата P_1 является взаимодействием с процессом, который является внешним по отношению к $\{P_1, P_2\}$, т.е., например, к автомату подошёл вор, и взял шоколадку, прежде чем это смог сделать покупатель P_2.

В этой ситуации покупатель не может выполнить второе действие как внутреннее действие процесса $\{P_1,P_2\}$. Согласно описанию процесса, в данном случае возможны два варианта поведения.

1. P_2 будет находиться в состоянии бесконечного ожидания.

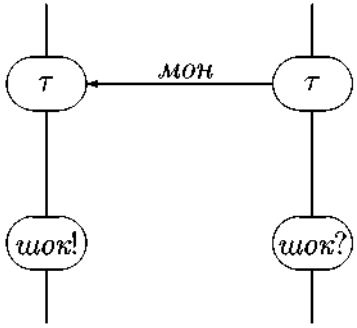
Соответствующая этому варианту MSC имеет вид



2. P_2 сможет успешно завершить работу.

Это произойдёт в том случае, если некоторый процесс, внешний по отношению к $\{P_1, P_2\}$, передаст шоколадку процессу

Соответствующая этому варианту MSC имеет вид



Теперь рассмотрим вопрос о том, каким образом процесс $\{P_1, P_2\}$ может быть определён явно, т.е. в терминах состояний и переходов с метками из множества *Act.*

На первый взгляд, данный вопрос является некорректным, так как $\{P_1, P_2\}$ представляет собой модель **параллельного** функционирования процессов P_1 и P_2, при котором

- в один и тот же момент времени могут быть исполнены действия обоими процессами, входящими в $\{P_1,P_2\}$, и,

• следовательно, процесс {P_1, P_2} может совершать такие действия, которые представляют собой пары действий из множества *Act,* т.е. они не принадлежат множеству *Act.*

Заметим на это, что абсолютная одновременность имеет место лишь для тех пар действий, которые порождают одно внутреннее действие процесса {P_1,P_2} типа (c). Для всех остальных пар действий процессов P_1 *и* P_2, даже если они произошли с точки зрения внешнего наблюдателя одновременно, мы можем предполагать без ограничения общности, что одно из них произошло немного раньше или немного позже другого.

Таким образом, мы можем считать, что процесс {P_1, P_2} функционирует последовательно, т.е. что при любом варианте функционирования процесса {P_1, P_2} выполняемые им действия образуют некоторую линейно упорядоченную последовательность

$$tr = (act_1, act_2, \ldots) \quad \textbf{(3.7)}$$

в которой действия упорядочены по времени их выполнения: сначала произошло act_1, затем - act_2, и т.д.

Поскольку каждый возможный вариант функционирования процесса {P_1, P_2} представляется некоторой MSC, то можно считать, что последовательность (3.7) получается некоторой *линеаризацией* этой MSC (т.е. "вытягиванием" её в цепочку).

Для определения понятия линеаризации MSC мы введём несколько вспомогательных понятий и обозначений. Пусть *C* - некоторая MCS. Тогда

• знакосочетание *Points(C)* обозначает множество всех точек, входящих в MSC *C,*

• для каждой точки $p \in Points(C)$ знакосочетание *act(p)* обозначает действие, приписанное точке *p*

• для каждой пары точек $p, p' \in Points(C)$ знакосочетание

$$p \to p'$$

означает, что имеет место одно из следующих условий:

— *p* и *p′* находятся на одной и той же нити, и *p′* расположена ниже, чем *p,* или

— существует синхронизационная стрелка с началом в *p* и концом в *p′*

• для каждой пары точек $p, p' \in Points(C)$ знакосочетание

$$p \le p'$$

означает, что либо $p = p'$, либо существует последовательность точек $p_1,\ldots,p_k$, такая, что

$$- \; p = p_1, \;\; p' = p_k$$

— для каждого $i = 1, \ldots, k-1 \quad p_i \to p_{i+1}$

Отношение ≤ на точках MSC можно рассматривать как отношение хронологической упорядоченности, т.е. знакосочетание $p \le p'$ можно интерпретировать как утверждение о том, что

• точки *p* и *p′* совпадают или соединены синхронизационной стрелкой (т.е. действия в *p* и в *p′* совпадают),

• или действие в *p′* произошло позже, чем произошло действие в *p*.

Точное определение понятия линеаризации MSC имеет следующий вид. Пусть заданы MSC *C* и последовательность действий *tr* вида (3.7). Обозначим множество индексов элементов последовательности *tr* знакосочетанием *Ind(tr),* т.е.

$$Ind(tr) = \{1, 2, \ldots\}$$

(данное множество может быть как конечным, так и бесконечным) . Последовательность *tr* называется **линеаризацией** MSC C, если существует сюрьективное отображение

$$lin : Points(C) \to Ind(tr)$$

удовлетворяющее следующим условиям.

1. для каждой пары $p, p' \in Points(C)$

$$p \le p' \quad \Rightarrow \quad lin(p) \le lin(p')$$

2. для каждой пары $p, p' \in Points(C)$ соотношение

$$lin(p) = lin(p')$$

имеет место тогда и только тогда, когда

•$p = p'$, или

• существует синхронизационная стрелка с началом в *p* и концом в *p′*

3. для каждой точки $p \in Points(C)$

$$act(p) = act_{lin(p)}$$

т.е. отображение *lin*

• сохраняет хронологический порядок,

• отождествляет те точки MSC C, которые соответствуют одному действию процесса *{P1,P2},* и

• не отождествляет никакие другие точки.

Обозначим символом *Li(C)* совокупность всех линеаризации MSC *C.* Теперь задачу явного описания процесса {P1, *P2}* можно сформулировать следующим образом: построить процесс *P,* удовлетворяющий условию

$$Tr(P) = \bigcup_{C \in Beh\{P_1, P_2\}} Lin(C) \qquad \textbf{(3.8)}$$

т.е. в процессе *P* должны быть представлены все линеаризации любого возможного совместного поведения процессов P_1 и P_2.
Условие (3.8) обосновывается следующим соображением: поскольку мы не знаем,

- как согласованы между собой часы в процессах P_1 и P_2, и
- какова продолжительность пребывания в каждом из состояний, в которые эти процессы попадают

то мы должны считать возможным любой порядок исполнения действий, который не противоречит отношению хронологической упорядоченности.
Приступим к построению процесса *P,* удовлетворяющему условию (3.8). Пусть процессы P_1 и P_2 имеют вид

$$P_i = (S_i, s_i^0, R_i) \quad (i = 1, 2)$$

Рассмотрим произвольную линеаризацию *tr* произвольной MSC из $Beh\{P_1, P_2\}$

$$tr = (a_1, a_2, \ldots)$$

Нарисуем линию, которую мы будем интерпретировать как временную шкалу. Выделим на ней точки p_1, p_2, ..., помеченные действиями $a_1, a_2, \ldots$, причём помеченные точки расположены в том порядке, в котором соответствующие действия перечислены в *tr*, т.е. сначала идёт точка p_1 с меткой a_1, за ней - точка p_2 с меткой a_2, и т.д.
Обозначим символами $I_0, I_1, I_2, \ldots$ следующие участки этой линии:

- I_0 представляет собой совокупность всех точек линии перед точкой p_1, т.е.

$$I_0 \stackrel{\text{def}}{=}]-\infty, p_1[$$

- для каждого $i \geq 1$ участок I_i состоит из точек между p_i и p_{i+1}, т.е.

$$I_i \stackrel{\text{def}}{=}]p_i, p_{i+1}[$$

Каждый из этих участков I_i можно интерпретировать как промежуток времени, в течение которого процесс *P* не выполняет никаких действий, т.е. в моменты времени между p_i и p_{i+1} процессы P_1 и P_2 находятся в фиксированных состояниях $(s_1)_i$ и $(s_2)_i$ соответственно.

Обозначим символом s_i пару $((s_1)_i, (s_2)_i)$. Данную пару можно интерпретировать как состояние всего процесса *P,* в котором он находится в каждый момент времени из промежутка I_i .
По определению последовательности *tr,* имеет место одна из двух ситуаций.
1. Действие a_i имеет тип (a) или (b), т.е. оно было выполнено одним из процессов, входящих в *P.*
Возможны два случая.
(a) Это действие было выполнено процессом P_1.
В этом случае мы имеем следующую связь между состояниями s_i и s_{i+1}:

- $(s_1)_i \xrightarrow{a_i} (s_1)_{i+1} \quad \in R_1$
- $(s_2)_{i+1} = (s_2)_i$

(b) Это действие было выполнено процессом P_2.
В этом случае мы имеем следующую связь между состояниями s_i и s_{i+1}:

- $(s_2)_i \xrightarrow{a_i} (s_2)_{i+1} \quad \in R_2$
- $(s_1)_{i+1} = (s_1)_i$

2. Действие a_i имеет тип (c).
В этом случае мы имеем следующую связь между состояниями s_i и s_{i+1}:

- $(s_1)_i \xrightarrow{a} (s_1)_{i+1} \quad \in R_1$
- $(s_2)_i \xrightarrow{\overline{a}} (s_2)_{i+1} \quad \in R_2$

для некоторого $a \in Act \setminus \{\tau\}$.

Сформулированные выше свойства последовательности *tr* можно переформулировать следующим образом: *tr* является трассой процесса

$$(S, s^0, R) \quad (3.9)$$

компоненты которого определяются следующим образом:

- $S \stackrel{\text{def}}{=} S_1 \times S_2 \stackrel{\text{def}}{=} \{(s_1, s_2) \mid s_1 \in S_1, s_2 \in S_2\}$
- $s^0 \stackrel{\text{def}}{=} (s_1^0, s_2^0)$

• для
— каждого перехода $s_1 \xrightarrow{a} s_1'$ из R_1, и
— каждого состояния $s \in S_2$

R содержит переход

$$(s_1, s) \xrightarrow{a} (s_1', s)$$

- для

— каждого перехода $s_2 \xrightarrow{a} s_2'$ из R_2, и

— каждого состояния $s \in S_1$

R содержит переход

$$(s, s_2) \xrightarrow{a} (s, s_2')$$

- для каждой пары переходов с комплементарными метками

$$s_1 \xrightarrow{a} s_1' \quad \in R_1$$
$$s_2 \xrightarrow{\bar{a}} s_2' \quad \in R_2$$

R содержит переход

$$(s_1, s_2) \xrightarrow{\tau} (s_1', s_2')$$

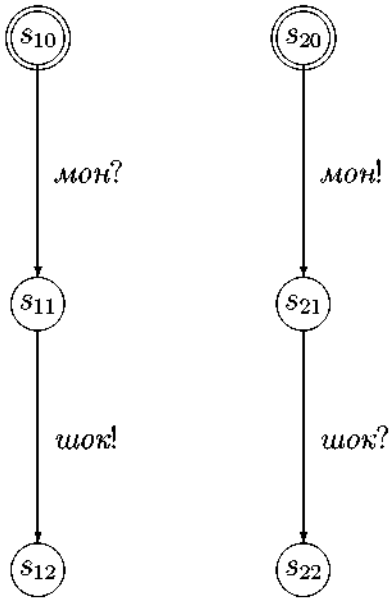
Нетрудно показать и обратное: каждая трасса в вышеопределённом процессе (3.9) является линеаризацией некоторой MSC *C* из множества $Beh\{P_1, P_2\}$.

Таким образом, в качестве искомого процесса P можно взять процесс (3.9). Данный процесс называется **параллельной композицией** процессов P_1 и P_2, и обозначается знакосочетанием

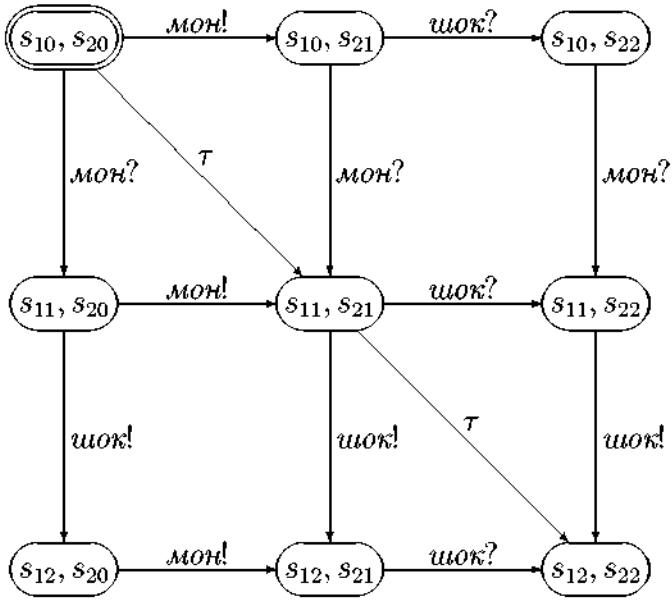
$$P_1 \mid P_2$$

Приведём в качестве примера процесс $P_1 \mid P_2$, где процессы P_1 и P_2 представляют поведение торгового автомата и покупателя (см. (3.5) и (3.6)).

Графовые представления этих процессов имеют вид



Графовое представление процесса $P_1 | P_2$ имеет вид



Заметим, что размер множества состояний процесса $P_1|P_2$ равен произведению размеров множеств состояний процессов P_1 и P_2, т.е. размер описания процесса $P_1 \mid P_2$ может существенно превышать суммарную сложность размеров описаний его компонентов P_1 и P_2. Это может сделать невозможным анализ такого процесса по причине его высокой сложности.

Поэтому в практических задачах при анализе процессов вида $P_1 \mid P_2$, вместо явного построения процесса $P_1 \mid P_2$ строится процесс, в котором каждая MSC из $Beh\{P_1, P_2\}$ представлена не всеми возможными линеаризациями, а хотя бы одной линеаризацией. Сложность такого процесса может быть существенно меньше по сравнению со сложностью процесса $P_1|P_2$.

Построение процесса такого вида имеет смысл, например, в том случае, когда анализируемое свойство φ процесса $P_1 \mid P_2$ обладает следующим качеством:

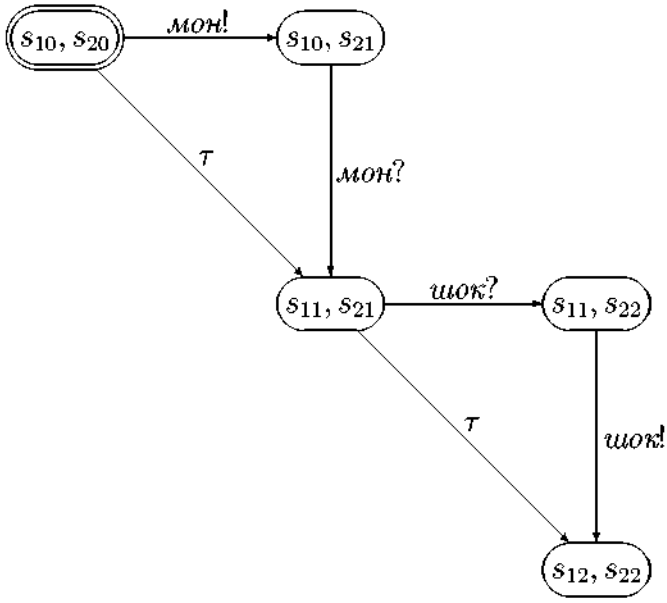
- если φ истинно *для одной из линеаризации* произвольной MSC $C \in Beh\{P_1, P_2\}$,
- то φ истинно *для всех линеаризации* этой MSC.

Как правило, процесс, в котором каждая MSC из $Beh\{P_1, P_2\}$ представлена не всеми возможными линеаризациями, а хотя бы одной линеаризацией, строится как некоторый **подпроцесс** процесса $P_1|P_2$, т.е.

получается из $P_1|P_2$ удалением некоторых состояний и связанных с ними переходов. Поэтому такие процессы называют **редуцированными.**

Проблема построения редуцированных процессов называется проблемой **редукции частичных порядков (partial order reduction).** Эта проблема интенсивно исследуется многими ведущими специалистами в области верификации.

Приведём в качестве примера редуцированный процесс для рассмотренного выше процесса $P_1 \mid P_2$, состоящего из торгового автомата и покупателя.



В заключение отметим, что задача анализа процессов, состоящих из нескольких взаимодействующих компонентов, наиболее часто возникает в ситуации, когда такими компонентами являются компьютерные программы и аппаратные устройства, функционирующие совместно в рамках единого программно-аппаратного комплекса.

Взаимодействие между такими программами осуществляется при помощи процессов-посредников, т.е. программы взаимодействуют друг с другом через посредство некоторых процессов, которые синхронно взаимодействуют с каждой из программ.

Взаимодействие между параллельно работающими программами обычно реализуется одним из следующих двух способов.

1. Взаимодействие через общую память.

В данном случае такими посредниками являются ячейки памяти, к которым имеют доступ обе программы.

Взаимодействие может осуществляться, например, так: одна программа пишет информацию в эти ячейки, а другая читает содержимое ячеек.

2. **Взаимодействие путем посылки сообщений.**

В данном случае посредником является канал, с которым программы могут осуществлять следующие операции:

- посылка сообщения в канал передающей программой, и
- приём сообщения принимающей программой.

Канал может представлять собой буфер, хранящий несколько сообщений. Сообщения в канале могут быть организованы по принципу очереди (т.е. сообщения покидают канал в том же порядке, в котором они в него поступают).

3.5 Ограничение

Пусть $P = (S, s^0, R)$некоторый процесс, и *L* - произвольное подмножество множества *Names.*

Ограничением *P* по *L* называется процесс

$$P \setminus L = (S, s^0, R')$$

который получается из *P* удалением тех переходов, которые имеют метки с именами из *L,* т.е.

$$R' \stackrel{\text{def}}{=} \left\{ (s \xrightarrow{a} s') \in R \;\middle|\; \begin{array}{l} a = \tau, \;\; \text{или} \\ name(a) \notin L \end{array} \right\}$$

Операция ограничения используется, как правило, совместно с операцией параллельной композиции для представления таких процессов, которые

- состоят из нескольких компонентов, и
- взаимодействие между этими компонентами должно удовлетворять некоторым ограничениям.

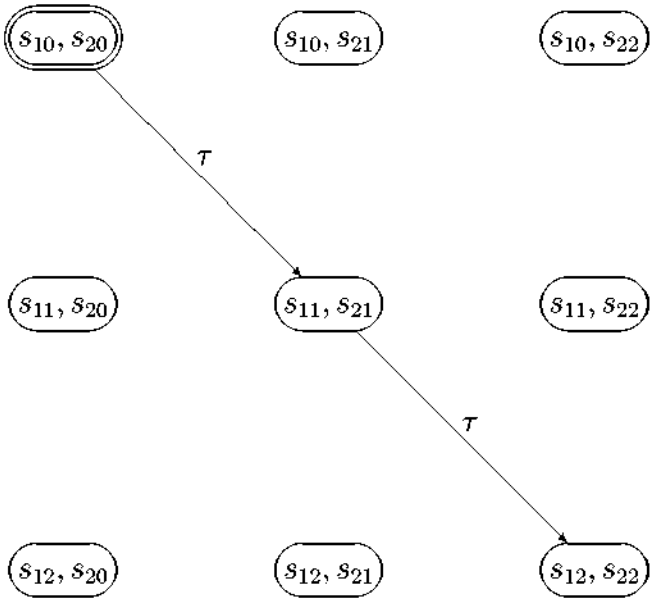
Например, пусть P_1 и P_2 - торговый автомат и покупатель, которые рассматривались в предыдущем параграфе.

Мы хотели бы описать процесс, являющийся моделью такого параллельного функционирования процессов P_1 и P_2, при котором эти процессы могут совершать действия, связанные с покупкой-продажей шоколадки, только совместно.

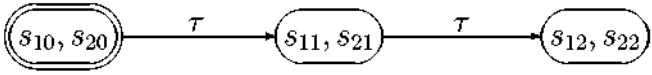
Искомый процесс может быть получен из процесса $P_1 | P_2$ применением операции ограничения по множеству имён всех действий, которые связаны с покупкой-продажей шоколадки, т.е. данный процесс описывается выражением

$$P \stackrel{\text{def}}{=} (P_1 | P_2) \setminus \{мон, шок\} \quad (3.10)$$

Графовое представление процесса (3.10) имеет вид



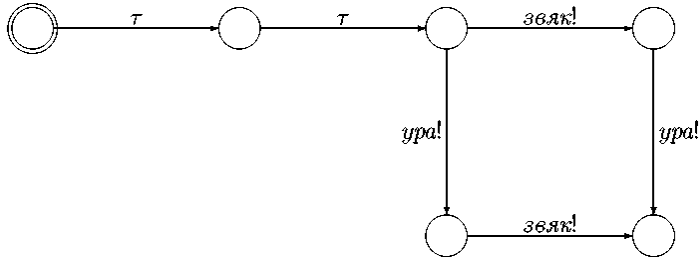
После удаления недостижимых состояний получится процесс, имеющий следующее графовое представление:



Рассмотрим другой пример. Немного изменим определения торгового автомата и покупателя: пусть они еще и сигнализируют об успешном выполнении своей работы, т.е. их процессы имеют, например, следующий вид:

$$P_1 \stackrel{\text{def}}{=} мон?.шок!.звяк!.\mathbf{0}$$
$$P_2 \stackrel{\text{def}}{=} мон!.шок?.ура!.\mathbf{0}$$

В этом случае графовое представление процесса (3.10) после удаления недостижимых состояний имеет вид



Данный процесс допускает исполнение тех невнутренних действий, которые не связаны с покупкой и продажей шоколадки.
Отметим, что в данном случае в процессе (3.10) присутствует недетерминизм, хотя в компонентах P_1 и P_2 его нет. Причиной возникновения этого недетерминизма является наше неполное знание о моделируемой системе: поскольку мы не имеем точных знаний о длительности действий *звяк!* и *ура!*, то модель системы должна допускать любой порядок их выполнения.

3.6 Переименование

Следующая операция, которую мы рассмотрим - это унарная операция **переименования.**
Для задания этой операции необходимо определить функцию

$$f : Names \to Names$$

называемую **переименованием.**
Действие данной операции на процесс *P* заключается в изменении меток переходов:

- метки вида α? заменяются на $f(\alpha)?$,и
- метки вида *α!* заменяются на $f(\alpha)!$

Получившийся процесс обозначается знакосочетанием *P[f]*.
Если переименование *f* действует нетождественно лишь на имена из списка

$$\alpha_1, \ldots, \alpha_n$$

и отображает их в имена

$\beta_1, \ldots, \beta_n$

соответственно, то для *P[f]* мы будем иногда использовать эквивалентное обозначение

$P[\beta_1/\alpha_1, \ldots, \beta_n/\alpha_n]$

Операция переименования позволяет многократно использовать один и тот же процесс *P* в качестве компоненты при построении более сложного процесса *P′*. Эта операция используется для предотвращения "конфликтов" между именами действий, используемых в различных вхождениях *P* в *P′*.

3.7 Свойства операций на процессах

В этом параграфе мы приводим некоторые простейшие свойства определённых выше операций на процессах. Все эти свойства имеют вид равенств. Для первых двух свойств мы даём их обоснование, остальные свойства приводятся без комментариев ввиду их очевидности.

Напомним, что мы считаем два процесса равными, если

- они изоморфны, или
- один из этих процессов можно получить из другого путём удаления некоторых недостижимых состояний и связанных с ними переходов.

1. Операция + ассоциативна, т.е. для любых процессов P_1, P_2 и P_3 верно равенство

$$(P_1 + P_2) + P_3 = P_1 + (P_2 + P_3)$$

Действительно, пусть процессы P_i (i = 1, 2, 3) имеют вид

$$P_i = (S_i, s_i^0, R_i) \quad (i = 1, 2, 3) \quad (3.11)$$

причём множества состояний S_1, S_2 и S_3 попарно не пересекаются. Тогда обе части данного равенства равны процессу $P = (S, s^0, R)$, компоненты которого определяются следующим образом:

- $S \stackrel{\text{def}}{=} S_1 \cup S_2 \cup S_3 \cup \{s^0\}$, где s^0 - новое состояние, не входящее в S_1, S_2 и S_3
- R содержит все переходы из R_1, R_2 и R_3
- для каждого перехода из R_i (i = 1, 2, 3) вида

$s_i^0 \xrightarrow{a} s$

R содержит переход $s^0 \xrightarrow{a} s$

Из свойства ассоциативности операции + следует, что допустимы выражения вида

$$P_1 + \ldots + P_n \quad (3.12)$$

так как при любой расстановке скобок в выражении (3.12) получится один и тот же процесс.

Процесс, являющийся значением выражения (3.12) можно описать явно следущим образом.

Пусть процессы $P_i \ (i = 1, \ldots, n)$ имеют вид

$$P_i = (S_i, s_i^0, R_i) \quad (i = 1, \ldots, n) \quad (3.13)$$

причём множества состояний $S_1, \ldots, S_n$ попарно не пересекаются. Тогда процесс, являющийся значением выражения (3.12), имеет вид

$$P = (S, s^0, R)$$

где компоненты S, s^0, R определяются следующим образом:

• $S \stackrel{\text{def}}{=} S_1 \cup \ldots \cup S_n \cup \{s^0\}$, где s^0 - новое состояние, не входящее в $S_1, \ldots, S_n$

• *R* содержит все переходы из $R_1, \ldots, R_n$

• для каждого перехода из $R_i \ (i = 1, \ldots, n)$ вида

$s_i^0 \xrightarrow{a} s$

R содержит переход $s^0 \xrightarrow{a} s$

2. Операция | ассоциативна, т.е. для любых процессов P_1, P_2 и P_3 верно равенство

$$(P_1 \mid P_2) \mid P_3 = P_1 \mid (P_2 \mid P_3)$$

Действительно, пусть процессы $P_i \ (i = 1, 2, 3)$ имеют вид (3.11). Тогда обе части данного равенства равны процессу $P = (S, s^0, R)$ компоненты которой определяются следующим образом:

- $S \stackrel{\text{def}}{=} S_1 \times S_2 \times S_3 \stackrel{\text{def}}{=} \stackrel{\text{def}}{=} \{(s_1, s_2, s_3) \mid s_1 \in S_1, s_2 \in S_2, s_3 \in S_3\}$
- $s^0 \stackrel{\text{def}}{=} (s_1^0, s_2^0, s_3^0)$

• для

— каждого перехода $s_1 \xrightarrow{a} s_1'$ из R_1, и

— каждой пары состояний $s_2 \in S_2, s_3 \in S_3$

R содержит переход

$$(s_1, s_2, s_3) \xrightarrow{a} (s_1', s_2, s_3)$$

• для

— каждого перехода $s_2 \xrightarrow{a} s_2'$ из R_2,и

— каждой пары состояний $s_1 \in S_1, s_3 \in S_3$

R содержит переход

$$(s_1, s_2, s_3) \xrightarrow{a} (s_1, s_2', s_3)$$

• для

— каждого перехода $s_3 \xrightarrow{a} s_3'$ из R_s, и

— каждой пары состояний $s_1 \in S_1, s_2 \in S_2$

R содержит переход

$$(s_1, s_2, s_3) \xrightarrow{a} (s_1, s_2, s_3')$$

• для

— каждой пары переходов с комплементарными метками

$$s_1 \xrightarrow{a} s_1' \quad \in R_1$$
$$s_2 \xrightarrow{\overline{a}} s_2' \quad \in R_2$$

и

— каждого состояния $s_3 \in S_3$ *R* содержит переход

$$(s_1, s_2, s_3) \xrightarrow{\tau} (s_1', s_2', s_3)$$

• для

— каждой пары переходов с комплементарными метками

$$s_1 \xrightarrow{a} s_1' \quad \in R_1$$
$$s_3 \xrightarrow{\overline{a}} s_3' \quad \in R_3$$

и

— каждого состояния $s_2 \in S_2$

R содержит переход

$$(s_1, s_2, s_3) \xrightarrow{\tau} (s_1', s_2, s_3')$$

• для

— каждой пары переходов с комплементарными метками

$$s_2 \xrightarrow{a} s_2' \quad \in R_2$$
$$s_3 \xrightarrow{\overline{a}} s_3' \quad \in R_3$$

и

— каждого состояния $s_1 \in S_1$

R содержит переход

$$(s_1, s_2, s_3) \xrightarrow{\tau} (s_1, s_2', s_3')$$

Из свойства ассоциативности операции $\mid$ следует, что допустимы выражения вида

$$P_1 \mid \ldots \mid P_n \quad (3.14)$$

так как при любой расстановке скобок в выражении (3.14) получится один и тот же процесс.

Процесс, являющийся значением выражения (3.14) можно описать явно следущим образом.

Пусть процессы $P_i \ (i = 1, \ldots, n)$ имеют вид (3.13). Тогда процесс, являющийся значением выражения (3.14), имеет вид

$$P = (S, s^0, R)$$

где компоненты S, s^0, R определяются следующим образом:

- $S \stackrel{\text{def}}{=} S_1 \times \ldots \times S_n \stackrel{\text{def}}{=}$
  $\stackrel{\text{def}}{=} \{(s_1, \ldots, s_n) \mid s_1 \in S_1, \ldots, s_n \in S_n\}$
- $s^0 \stackrel{\text{def}}{=} (s_1^0, \ldots, s_n^0)$

• для

— каждого $i \in \{1, \ldots, n\}$

— каждого перехода $s_i \xrightarrow{a} s_i'$ из R_i, и

— каждого списка состояний

$s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n$

где $\forall\, j \in \{1, \ldots, n\} \quad s_j \in S_j$

R содержит переход

$$(s_1, \ldots, s_n) \xrightarrow{a} (s_1, \ldots, s_{i-1}, s_i', s_{i+1}, \ldots, s_n)$$

• для

— каждой пары индексов $i, j \in \{1, \ldots, n\}$, где $i < j$

— каждой пары переходов с комплементарными метками

$$s_i \xrightarrow{a} s_i' \quad \in R_i$$
$$s_j \xrightarrow{\bar{a}} s_j' \quad \in R_j$$

и

— каждого списка состояний

$$s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_{j-1}, s_{j+1}, \ldots, s_n$$

где $\forall\, k \in \{1, \ldots, n\} \quad s_k \in S_k$

R содержит переход

$$(s_1, \ldots, s_n) \xrightarrow{\tau} \left(\begin{array}{l} s_1, \ldots, s_{i-1}, s_i', s_{i+1}, \ldots, s_{j-1}, s_j', \\ s_{j+1}, \ldots, s_n \end{array} \right)$$

3. Операция + коммутативна, т.е. для любых процессов P_1 и P_2 верно равенство

$$P_1 + P_2 = P_2 + P_1$$

4. Операция | коммутативна, т.е. для любых процессов P_1 и P_2 верно равенство

$$P_1 \,|\, P_2 = P_2 \,|\, P_1$$

5. 0 является нейтральным элементом относительно операции |:

$$P \,|\, \mathbf{0} = P$$

Для операции + аналогичное свойство тоже имеет место, если вместо равенства процессов будет использовано понятие сильной эквивалентности процессов (которое определяется ниже). Данное свойство, а также свойство идемпотентности операции + доказываются в параграфе 4.5 (теорема 4).

6. $\mathbf{0} \setminus L = \mathbf{0}$
7. $\mathbf{0}[f] = \mathbf{0}$
8. $P \setminus L = P$, если $L \cap names(Act(P)) = \emptyset$.

   (напомним, что $Act(P)$ обозначает множество действий $a \in Act \setminus \{\tau\}$, таких, что P содержит переход с меткой a)
9. $(a.P) \setminus L = \begin{cases} \mathbf{0}, & \text{если } a \neq \tau \text{ и } name(a) \in L \\ a.(P \setminus L), & \text{иначе} \end{cases}$
10. $(P_1 + P_2) \setminus L = (P_1 \setminus L) + (P_2 \setminus L)$
11. $(P_1 \mid P_2) \setminus L = (P_1 \setminus L) \mid (P_2 \setminus L)$, если

$$L \cap names(Act(P_1) \cap \overline{Act(P_2)}) = \emptyset$$

12. $(P \setminus L_1) \setminus L_2 = P \setminus (L_1 \cup L_2)$
13. $P[f] \setminus L = (P \setminus f^{-1}(L))[f]$
14. $P[id] = P$, где id – тождественная функция
15. $P[f] = P[g]$, если сужения функций f и g на множество $names(Act(P))$ совпадают.
16. $(a.P)[f] = f(a).(P[f])$
17. $(P_1 + P_2)[f] = P_1[f] + P_2[f]$
18. $(P_1 \mid P_2)[f] = P_1[f] \mid P_2[f]$,если сужение функции f на множество

$$names(Act(P_1) \cup Act(P_2))$$

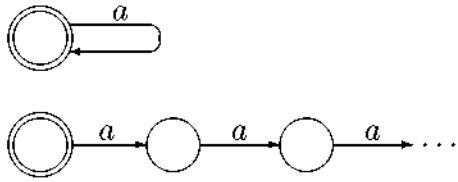
является инъективной функцией.

19. $(P \setminus L)[f] = P[f] \setminus f(L)$, если f иньективна.
20. $P[f][g] = P[g \circ f]$

4.Эквивалентность процессов

4.1 Понятие эквивалентности процессов и связанные с ним задачи

Одно и то же поведение может быть представлено различными процессами. Например, рассмотрим два процесса:



Хотя у первого процесса всего одно состояние, а у второго множество состояний бесконечно, но эти процессы представляют одно и то же поведение, которое заключается в постоянном выполнении одного и того же действия *a.*

Представляет интерес поиск подходящего определения эквивалентности процессов, согласно которому процессы эквивалентны тогда и только тогда, когда они имеют одно и то же поведение.

В этой главе мы излагаем несколько определений понятия эквивалентности процессов. Выбор того или иного варианта данного понятия в конкретной ситуации должен определяться тем, как именно в данной ситуации понимается одинаковость поведения процессов.

В параграфах 4.2 и 4.3 вводятся понятия трассовой эквивалентности и сильной эквивалентности процессов. Данные понятия используются в той ситуации, когда все действия, выполняющиеся в процессах, имеют одинаковый статус.

В параграфах 4.8 и 4.9 предлагаются другие варианты понятия эквивалентности процессов: наблюдаемая эквивалентность и наблюдаемая конгруэнция. Данные понятия используются в тех ситуациях, когда мы рассматриваем невидимое действие τ как несущественное, и считаем две трассы одинаковыми, если одна может быть получена из другой путём вставок и/или удалений невидимых действий τ.

С каждым из возможных вариантов понятия эквивалентности процессов связаны две естественные задачи.

1. Распознавание для двух заданных процессов, являются ли они эквивалентными.

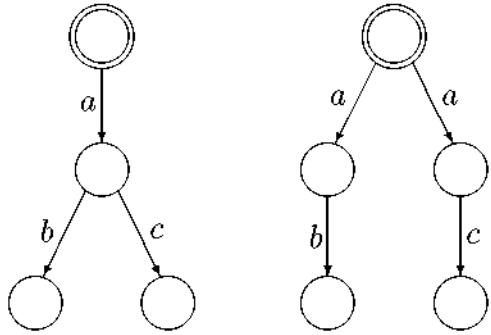
2. Построение по заданному процессу *P* такого процесса *P′,* который является наименее сложным (например, имеет минимальное число состояний) среди всех процессов, которые эквивалентны *P.*

4.2 Трассовая эквивалентность процессов

Как уже было сказано выше, мы хотели бы рассматривать два процесса как эквивалентные, если они описывают одно и то же поведение. Поэтому, если мы рассматриваем поведение процесса как порождение некоторой трассы, то необходимым условием эквивалентности процессов P_1 и P_2 является совпадение множеств их трасс:

$$Tr(P_1) = Tr(P_2) \quad (4.1)$$

В некоторых ситуациях условие (4.1) можно использовать в качестве определения эквивалентности между P_1 и P_2. Однако нижеследующий пример показывает, что это условие не отражает один важный аспект поведения процессов.



(4.2)

Множества трасс этих процессов совпадают:

$$Tr(P_1) = Tr(P_2) = \{\varepsilon, a, ab, ac\}$$

(где ε - пустая последовательность).

Однако эти процессы отличаются тем, что

• в левом процессе после выполнения первого действия (a) сохраняется возможность выбора следующего действия (b или c), в то время как

• в правом процессе после выполнения первого действия такого выбора нет:

— если первый переход был совершён по левому ребру, то вторым действием может быть только действие *b,*

— а если по правому - то только действие c,

т.е. второе действие было предопределено ещё до выполнения первого действия.

Если мы не хотели бы рассматривать эти процессы как эквивалентные, то условие (4.1) надо некоторым образом усилить. Один из вариантов такого усиления излагается ниже. Для того, чтобы его сформулировать, определим сначала понятие трассы из некоторого состояния процесса. Каждый вариант поведения процесса $P = (S, s^0, R)$ мы интерпретируем как порождение некоторой последовательности переходов

$$s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} \dots \quad (4.3)$$

начинающейся с начального состояния, т.е. $s_0 = s^0$.

Мы можем рассматривать порождение последовательности (4.3) не только с начального, а с произвольного состояния $s \in S$, т.е. рассматривать последовательность вида (4.3), в которой $s_0 = s$. Последовательность (a_1, a_2,...) меток этих переходов мы будем называть **трассой с началом в** *s*. Множество всех таких трасс мы будем обозначать знакосочетанием $Tr_s(P)$.

Пусть P_1 и P_2 - процессы вида

$$P_i = (S_i, s_i^0, R_i) \qquad (i = 1, 2)$$

Рассмотрим произвольную конечную последовательность переходов в P_1 вида

$$s_1^0 = s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \dots \xrightarrow{a_n} s_n \quad (n \geq 0) \quad \textbf{(4.4)}$$

(случай $n = 0$ соответствует пустой последовательности переходов (4.4), в которой $s_n = s_1^0$)

Последовательность (4.4) можно рассматривать как начальный этап функционирования процесса P_1, а каждую трассу из $Tr_{s_n}(P_1)$ - как некоторое продолжение этого этапа.

Процессы P_1 и P_2 называются **трассово эквивалентными,** если

- для каждого начального этапа (4.4) функционирования процесса *Pi* существует начальный этап функционирования процесса P_2

$$s_2^0 = s_0' \xrightarrow{a_1} s_1' \xrightarrow{a_2} \dots \xrightarrow{a_n} s_n' \quad \textbf{(4.5)}$$

который имеет точно такую же трассу $a_1 \dots a_n$, что и (4.4), и в конце которого имеется точно такой же выбор дальнейшего продолжения, что и в конце этапа (4.4), т.е.

$$Tr_{s_n}(P_1) = Tr_{s_n'}(P_2) \quad \textbf{(4.6)}$$

- и, кроме того, имеет место симметричное условие: для каждой последовательности переходов в P_2 вида (4.5) должна существовать

последовательность переходов в P_1 вида (4.4), такая, что имеет место равенство (4.6).
Данные условия имеют недостаток: в их формулировке участвуют неограниченные множества последовательностей переходов вида (4.4) и (4.5), а также неограниченные множества трасс из (4.6). Поэтому проверка данных условий представляется затруднительной даже в том случае, когда процессы P_1 и P_2 конечны.
Представляет интерес задача нахождения условий, равносильных условиям трассовой эквивалентности, которые можно было бы алгоритмически проверять для заданных процессов P_1 и P_2 в том случае, когда эти процессы конечны.
Иногда рассматривают такую эквивалентность между процессами, которая отличается от трассовой эквивалентности заменой условия (4.6) на более слабое условие:

$$Act(s_n) = Act(s'_n)$$

где для каждого состояния *s* знакосочетание *Act(s)* обозначает множество всех действий $a \in Act$, таких, что существует переход с началом в *s* и помеченный действием *a.*

4.3 Сильная эквивалентность

Ещё одним вариантом понятия эквивалентности процессов является **сильная эквивалентность.** Для определения этого понятия мы введём вспомогательные обозначения. После того, как процесс

$$P = (S, s^0, R) \quad \textbf{(4.7)}$$

выполнит первое действие и перейдёт в новое состояние s^1, его поведение будет неотличимо от поведения процесса

$$P' \stackrel{\text{def}}{=} (S, s^1, R) \quad \textbf{(4.8)}$$

имеющего те же компоненты, что и *P,* за исключением начального состояния.
Мы будем использовать обозначение

$$P \xrightarrow{a} P' \quad (4.9)$$

как сокращённую запись утверждения о том, что
- *P* и *P′* - процессы вида (4.7) и (4.8) соответственно, и
- *R* содержит переход $s^0 \xrightarrow{a} s^1$.

(4.9) можно интерпретировать как утверждение о том, что процесс *P,* может
- выполнить действие *a,* и после этого

• вести себя как процесс *P'*.

Понятие сильной эквивалентности основано на следующем понимании эквивалентности процессов: если мы рассматриваем процессы P_1 и P_2 как эквивалентные, то должно быть выполнено следующее условие:

• если один из этих процессов P_i может

— выполнить некоторое действие $a \in Act$,

— и после этого вести себя как некоторый процесс P_i'

• то и другой процесстоже должен обладать

$$P_j \quad (j \in \{1,2\} \setminus \{i\})$$

способностью

— выполнить то же самое действие *a,*

— после чего вести себя как некоторый процесс P'$_j$, который эквивалентен P_i'.

Таким образом, искомая эквивалентность должна представлять собой некоторое бинарное отношение μ на множестве всех процессов, обладающее следующими свойствами.

(1) Если $(P_1, P_2) \in \mu$, и для некоторого процесса P_1' верно утверждение

$$P_1 \xrightarrow{a} P_1' \qquad (4.10)$$

то должен существовать процесс P_2', такой, что выполнены условия

$$P_2 \xrightarrow{a} P_2' \qquad (4.11)$$

и

$$(P_1', P_2') \in \mu \quad (4.12)$$

(2) Симметричное свойство: если $(P_1, P_2) \in \mu$,и для некоторого процесса P'_2 верно (4.11), то должен существовать процесс P_1', такой, что выполнены условия (4.10) и (4.12).

Обозначим символом $\mathcal{M}$ совокупность всех бинарных отношений, которые обладают вышеприведёнными свойствами.

Множество $\mathcal{M}$ непусто: оно содержит, например, диагональное отношение, которое состоит из всех пар вида *(P,* P), где *P* - произвольный процесс.

Встаёт естественный вопрос о том, какое же из отношений, входящих в $\mathcal{M}$,можно использовать для определения понятия сильной эквивалентности.

Мы предлагаем наиболее простой ответ на этот вопрос: мы будем считать P_1 и P_2 сильно эквивалентными в том и только в том случае, когда существует хотя бы одно отношение $\mu \in \mathcal{M}$, которое содержит пару (P_1,P_2).

Таким образом, искомое отношение сильной эквивалентности на множестве всех процессов мы определяем как объединение всех отношений из$\mathcal{M}$. Данное отношение обозначается символом ~.

Нетрудно доказать, что

• $\sim \in \mathcal{M}$, и

• ~ является отношением эквивалентности, т.к.

— рефлексивность ~ следует из того, что диагональное отношение принадлежит$\mathcal{M}$,

— симметричность ~ следует из того, что если $\mu \in \mathcal{M}$, то $\mu^{-1} \in \mathcal{M}$

— транзитивность ~ следует из того, что если $\mu_1 \in \mathcal{M}$ и $\mu_2 \in \mathcal{M}$, то $\mu_1 \circ \mu_2 \in \mathcal{M}$.

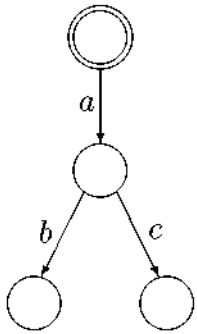
Если процессы P_1 и P_2 сильно эквивалентны, то этот факт обозначается знакосочетанием

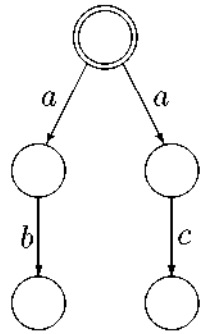
$$P_1 \sim P_2$$

Нетрудно доказать, что если процессы P_1 и P_2 сильно эквивалентны, то они трассово эквивалентны.

Для иллюстрации понятия сильной эквивалентности рассмотрим пару примеров.

1. Процессы

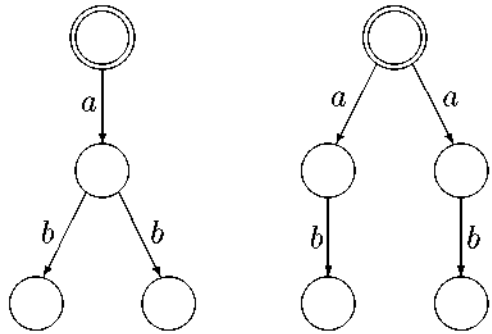




(4.13)

не являются сильно эквивалентными, так как они не являются трассово эквивалентными

2. Процессы



являются сильно эквивалентными.

4.4 Критерии сильной эквивалентности

4.4.1 Логический критерий сильной эквивалентности

Обозначим символом *Fm* множество **формул,** определяемое следующим образом.

- Символы $\top$ **и** $\bot$ являются формулами.
- Если φ - формула, то $\neg\varphi$ тоже формула.
- Если φ и ψ- формулы, то $\varphi \wedge \psi$тоже формула.
- Если φ - формула и $a \in Act$, то *(a)*φ тоже формула.

Пусть заданы процесс *P* и формула $\varphi \in Fm$. **Значение** формулы φ на процессе *P* представляет собой элемент *P(φ)* множества {0,1}, определяемый следующим образом.

- $P(\top) = 1, \;\; P(\bot) = 0$
- $P(\neg\varphi) = 1 - P(\varphi)$
- $P(\varphi \wedge \psi) = P(\varphi) \cdot P(\psi)$
- $$P(\langle a \rangle \varphi) = \begin{cases} 1, & \text{если существует процесс } P' : \\ & P \xrightarrow{a} P',\ P'(\varphi) = 1 \\ 0, & \text{в противном случае} \end{cases}$$

Теория процесса *P* - это совокупность *Th(P)* формул, определяемая следующим образом:

$$Th(P) = \{\varphi \in Fm \mid P(\varphi) = 1\}$$

Теорема 1.

Пусть процессы P_1 и P_2 конечны. Тогда

$$P_1 \sim P_2 \quad \Leftrightarrow \quad Th(P_1) = Th(P_2)$$

Доказательство.

Импликация "$\Rightarrow$" доказывается индукцией по структуре формулы φ.

Докажем импликацию "$\Leftarrow$". Пусть имеет место равенство

$$Th(P_1) = Th(P_2) \quad (4.14)$$

Обозначим символом μ бинарное отношение на множестве всех процессов, которое состоит из всех пар процессов с одинаковыми теориями, т.е.

$$\mu \stackrel{\text{def}}{=} \{(P_1, P_2) \mid Th(P_1) = Th(P_2)\}$$

Докажем, что μ удовлетворяет определению сильной эквивалентности. Пусть это не так, т.е., например, для некоторого

$$a \in Act$$

(a) существует процесс P_1', такой, что

$$P_1 \xrightarrow{a} P_1'$$

(b) но не существует процесса P_2', такого, что

$$P_2 \xrightarrow{a} P_2' \quad (4.15)$$

и $Th(P_1') = Th(P_2')$.

Условие (b) может иметь место в двух ситуациях:

1. не существует процесса P_2', для которого верно (4.15)
2. существует процесс P_2', для которого верно (4.15), но для каждого такого процесса

$$Th(P_1') \neq Th(P_2')$$

Докажем, что в обоих ситуациях существует формула φ, удовлетворяющая условию

$$P_1(\varphi) = 1, \quad P_2(\varphi) = 0$$

что будет противоречить предположению (4.14).

1. Если имеет место первая ситуация, то в качестве φ можно взять формулу $\langle a \rangle \top$.
2. Если имеет место вторая ситуация, то пусть список всех таких процессов P'_2, для которых верно (4.15), имеет вид

$$P'_{2,1}, \ldots, P'_{2,n}$$

По предположению, для каждого $i = 1,..., n$ имеет место неравенство

$$Th(P_1') \neq Th(P_{2,i}')$$

т.е. для каждого $i = 1,..., n$ существует формула φ_i, такая, что

$$P_1'(\varphi_i) = 1, \qquad P_{2,i}'(\varphi_i) = 0$$

В этой ситуации в качестве искомой формулы φ можно взять формулу $\langle a \rangle (\varphi_1 \wedge \ldots \wedge \varphi_n)$.

Например, пусть P_1 и P_2 - процессы, изображённые на рисунке (4.13). Как было сказано выше, эти процессы не являются сильно эквивалентными. В качестве обоснования утверждения $P_1 \not\sim P_2$ можно, например, предъявить формулу

$$\varphi \stackrel{\text{def}}{=} \langle a \rangle (\langle b \rangle \top \wedge \langle c \rangle \top)$$

Нетрудно доказать, что $P_1(\varphi) = 1$ и $P_2(\varphi) = 0$.

Представляет интерес задача нахождения по двум заданным процессам P_1 и P_2 списка формул

$$\varphi_1, \ldots, \varphi_n$$

как можно меньшего размера, таких, что $P_1 \sim P_2$ тогда и только тогда, когда

$$\forall\, i = 1, \ldots, n \qquad P_1(\varphi_i) = P_2(\varphi_i)$$

4.4.2 Критерий сильной эквивалентности, основанный на понятии бимоделирования

Теорема 2.

Пусть заданы два процесса

$$P_i = (S_i, s_i^0, R_i) \qquad (i = 1, 2)$$

$P_1 \sim P_2$ тогда и только тогда, когда существует отношение

$$\mu \subseteq S_1 \times S_2$$

удовлетворяющее следующим условиям.

0. $(s_1^0, s_2^0) \in \mu$.

1. Для каждой пары $(s_1, s_2) \in \mu$ и каждого перехода из R_1 вида

$$s_1 \xrightarrow{a} s_1'$$

существует переход из R_2 вида

$$s_2 \xrightarrow{a} s_2'$$

такой, что $(s_1', s_2') \in \mu$.

2. Для каждой пары $(s_1, s_2) \in \mu$ и каждого перехода из R_2 вида

$$s_2 \xrightarrow{a} s_2'$$

существует переход из R_1 вида

$$s_1 \xrightarrow{a} s_1'$$

такой, что $(s_1', s_2') \in \mu$.

Отношение μ, удовлетворяющее данным условиям, называется **бимоделированием (БМ)** между P_1 и P_2.

4.5 Алгебраические свойства сильной эквивалентности

Теорема 3.
Сильная эквивалентность является конгруэнцией, т.е. если $P_1 \sim P_2$, то

- для каждого $a \in Act \quad a.P_1 \sim a.P_2$
- для каждого процесса $P \quad P_1 + P \sim P_2 + P$
- для каждого процесса $P \quad P_1|P \sim P_2|P$
- для каждого $L \subseteq Names \quad P_1 \setminus L \sim P_2 \setminus L$
- для каждого переименования $f \quad P_1[f] \sim P_2[f]$

Доказательство.
Как было установлено в параграфе 4.4.2, соотношение

$$P_1 \sim P_2$$

эквивалентно тому, что существует БМ μ, между P_1 и P_2. Используя это μ, мы построим БМ для обоснования каждого из вышеприведённых соотношений.

•Пусть символыобозначают $s_{(1)}^0$ и $s_{(2)}^0$ начальные состояния

$a.P_1$ и $a.P_2$ соответственно.

Тогда отношение

$$\{(s^0_{(1)}, s^0_{(2)})\} \cup \mu$$

является БМ между $a.P_1$ и $a.P_2$.

• Пусть

— символы $s^0_{(1)}$ и $s^0_{(2)}$ обозначают начальные состояния $P_1 + P$ и $P_2 + P$ соответственно, и

— символ *S* обозначает множество состояний процесса *P*.

Тогда

— отношение

$$\{(s^0_{(1)}, s^0_{(2)})\} \cup \mu \cup Id_S$$

является БМ между $P_1 + P$ и $P_2 + P$,и

— отношение

$$\{((s_1, s), (s_2, s)) \mid (s_1, s_2) \in \mu,\ q \in S\}$$

является БМ между $P_1|P$ и $P_2|P$.

• Отношение μ, является БМ

– между $P_1 \setminus L$ и $P_2 \setminus L$, и

– между $P_1[f]$ и $P_2[f]$.

Теорема 4.

Для каждого процесса $P = (S, s^0, R)$ имеют место следующие свойства.

1. $P + \mathbf{0} \sim P$

2. $P + P \sim P$

Доказательство.

1. Обозначим символом символом s^0_0 начальное состояние процесса *P* + 0.

БМ между *P* + 0 и *P* имеет вид

$$\{(s^0_0, s^0)\} \cup Id_S$$

2. Согласно определению операции +, процессы в левой части доказываемого соотношения следует рассматривать как две дизъюнктные изоморфные копии процесса *P* вида

$$P_{(i)} = (S_{(i)}, s^0_{(i)}, R_{(i)}) \quad (i = 1, 2)$$

где $S_{(i)} = \{s_{(i)} \mid s \in S\}$.

Обозначим символом символом s^0_0 начальное состояние процесса

P + P.

БМ между *P + P* и *P* имеет вид

$$\{(s_0^0, s^0)\} \cup \{(s_{(i)}, s) \mid s \in S,\ i = 1,2\}$$

Ниже для

- каждого процесса $P = (S, s^0, R)$, и
- каждого состояния $s \in S$

мы будем использовать знакосочетание *P(s)* для обозначения процесса (S, s, R)

который отличается от *P* только начальным состоянием.

Теорема 5.

Пусть $P = (S, s^0, R)$ - некоторый процесс, и совокупность всех выходящих из s^0 переходов имеет вид

$$\{ s^0 \xrightarrow{a_i} s^i \mid i = 1, \ldots, n\}$$

Тогда

$$P \sim a_1.P_1 + \ldots + a_n.P_n \quad (4.16)$$

где для каждого *i* = 1,..., *n*

$$P_i \stackrel{\text{def}}{=} P(s^i) \stackrel{\text{def}}{=} (S, s^i, R)$$

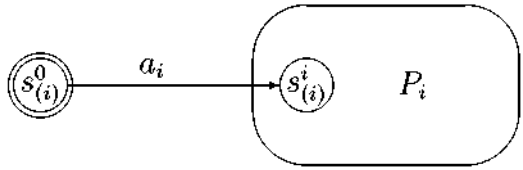
Доказательство.

Для построения БМ, доказывающего соотношение (4.16), мы заменим все процессы P_i в его правой части на их дизъюнктные копии, т.е. будем считать, что для каждого *i* = 1,... ,*n* процесс P_i имеет вид

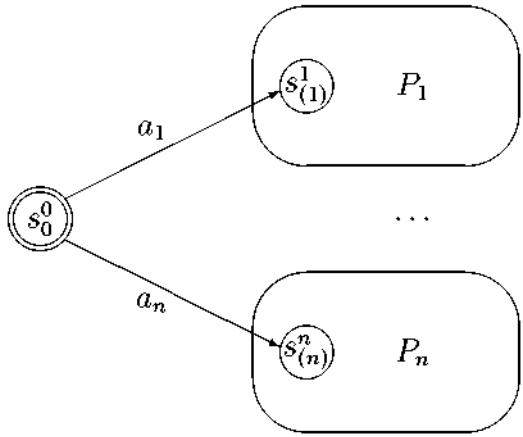
$$P_i = (S_{(i)}, s^i_{(i)}, R_{(i)})$$

и для каждого *i* = 1,..., *n* соответствующая биекция между *S* и $S_{(i)}$ сопоставляет каждому состоянию $s \in S$ состояние, обозначаемое СИМВОЛОМ $s_{(i)}$.

Таким образом, можно считать, что каждое из слагаемых $a_i.P_i$ в правой части (4.16) имеет вид



и множества состояний этих слагаемых попарно не пересекаются. Согласно определению операции +, правая часть (4.16) имеет вид



БМ между левой и правой частями (4.16) имеет вид

$$\{(s^0, s^0_0)\} \cup \{(s, s_{(i)}) \mid s \in S,\ i = 1, \ldots, n\}$$

Теорема 6 (теорема о разложении).
Пусть *P* - процесс вида

$$P = P_1 \mid \ldots \mid P_n \quad (4.17)$$ где для

каждого $i \in \{1, \ldots, n\}$процесс P_i имеет вид

$$P_i = \sum_{j=1}^{n_i} a_{ij}.\ P_{ij} \quad (4.18)$$

Тогда *P* сильно эквивалентен сумме
1. всех процессов вида

$$a_{ij}.\ \left(P_1 \mid \ldots \mid P_{i-1} \mid P_{ij} \mid P_{i+1} \mid \ldots \mid P_n\right) \quad (4.19)$$

2. и всех процессов вида

$$\tau.\left(\begin{array}{l} P_1 \mid \ldots \mid P_{i-1} \mid P_{ik} \mid P_{i+1} \mid \ldots \\ \ldots \mid P_{j-1} \mid P_{jl} \mid P_{j+1} \mid \ldots \mid P_n \end{array}\right) \quad (4.20)$$

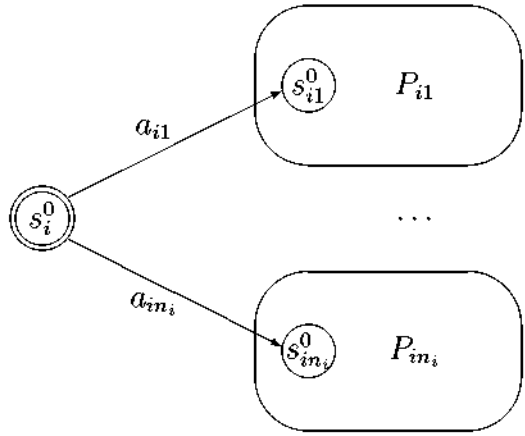
где $1 \le i < j \le n, \quad a_{ik}, a_{jl} \ne \tau$, и $a_{ik} = \overline{a_{jl}}$.

Доказательство.
По теореме 5, *P* сильно эквивалентен сумме, каждое слагаемое в которой соответствует некоторому переходу, выходящему из начального состояния s^0 процесса *P:* для каждого перехода в *P* вида

$$s^0 \xrightarrow{a} s$$

эта сумма содержит слагаемое $a.P(s)$.

Согласно (4.18), для каждого $i = 1,\ldots,n$ процесс P_i имеет вид



где $s_i^0, s_{i1}^0, \ldots, s_{in_i}^0$ - начальные состояния процессов

$$P_i,\ \ P_{i1},\ \ \ldots,\ \ P_{in_i}$$

соответственно.

Обозначим

- символом S_i множество состояний процесса P_i, и
- символом S_{ij} (где $j = 1,\ldots,n_i$) - множество состояний процесса P_{ij}.

Мы можем считать, что S_i является дизъюнктным объединением

$$S_i = \{s_i^0\} \cup S_{i1} \cup \ldots \cup S_{in_i} \quad (4.21)$$

Согласно описанию процесса вида (4.17), которое приведено в пункте 2 параграфа 3.7, можно считать, что компоненты процесса P имеют следующий вид.

- Множество состояний процесса P имеет вид
  $S_1 \times \ldots \times S_n$ (4.22)
- Начальным состоянием s^0 процесса P является список
  $(s_1^0, \ldots, s_n^0)$
- Переходы процесса P, выходящие из его начального состояния, имеют следующий вид.

— Переходы вида

$$s^0 \xrightarrow{a_{ij}} (s_1^0, \ldots, s_{i-1}^0, s_{ij}^0, s_{i+1}^0, \ldots, s_n^0) \quad (4.23)$$

— Переходы вида

$$s^0 \xrightarrow{\tau} \begin{pmatrix} s_1^0, \ldots, s_{i-1}^0, s_{ik}^0, s_{i+1}^0, \ldots \\ \ldots s_{j-1}^0, s_{jl}^0, s_{j+1}^0, \ldots, s_n^0 \end{pmatrix} \quad (4.24)$$

где $1 \le i < j \le n, \quad a_{ik}, a_{jl} \ne \tau$, и $a_{ik} = \overline{a_{jl}}$.

Таким образом, множество переходов процесса *P,* выходящих из s^0, находится во взаимно однозначном соответствии с множеством слагаемых вида (4.19) и (4.20).

Для доказательства теоремы 6 достаточно доказать, что

• Для каждого $i = 1,..., n,$ и каждого $j = 1,..., n_i$ имеет место эквивалентность

$$P(s_1^0, \ldots, s_{i-1}^0, s_{ij}^0, s_{i+1}^0, \ldots, s_n^0) \sim \\ \sim \left(P_1 \mid \ldots \mid P_{i-1} \mid P_{ij} \mid P_{i+1} \mid \ldots \mid P_n\right) \quad (4.25)$$

• Для

— любых *i,j,* таких, что $1 \le i < j \le n,$ и

— любых $k = 1, \ldots, n_i, \quad l = 1, \ldots, n_j$

имеет место эквивалентность

$$P \begin{pmatrix} s_1^0, \ldots, s_{i-1}^0, s_{ik}^0, s_{i+1}^0, \ldots \\ \ldots s_{j-1}^0, s_{jl}^0, s_{j+1}^0, \ldots, s_n^0 \end{pmatrix} \sim \\ \sim \begin{pmatrix} P_1 \mid \ldots \mid P_{i-1} \mid P_{ik} \mid P_{i+1} \mid \ldots \\ \ldots \mid P_{j-1} \mid P_{jl} \mid P_{j+1} \mid \ldots \mid P_n \end{pmatrix} \quad (4.26)$$

Мы докажем лишь эквивалентность (4.25) (эквивалентность (4.26) доказывается аналогично). Множество состояний процесса

$$\left(P_1 \mid \ldots \mid P_{i-1} \mid P_{ij} \mid P_{i+1} \mid \ldots \mid P_n\right) \quad (4.27)$$

имеет вид

$$S_1 \times \ldots \times S_{i-1} \times S_{ij} \times S_{i+1} \times \ldots \times S_n \quad (4.28)$$

Из (4.21) следует, что $S_{ij} \subseteq S_i$,т.е. множество (4.28) является подмножеством множества (4.22) состояний процесса

$$P(s_1^0, \ldots, s_{i-1}^0, s_{ij}^0, s_{i+1}^0, \ldots, s_n^0) \quad (4.29)$$

Определим искомое БМ μ между процессами (4.27) и (4.29) как диагональное отношение

$$\mu \stackrel{\text{def}}{=} \{(s, s) \mid s \in (4.28)\}$$

Очевидно, что

• пара начальных состояний процессов (4.27) и (4.29) принадлежит μ,

• каждый переход процесса (4.27) является также переходом процесса (4.29), и
• если начало некоторого перехода процесса (4.29) принадлежит подмножеству (4.28), то конец этого перехода тоже принадлежит подмножеству (4.28) (для обоснования этого утверждения отметим, что все переходы в P_i с началом в S_{ij} имеют конец тоже в S_{ij}).
Таким образом, μ является БМ, и это доказывает эквивалентность (4.25).
Следующая теорема является усилением теоремы 6. Для её формулировки мы будем использовать следующее обозначение. Если f - произвольное переименование, то тот же символ f обозначает функцию вида

$$f : Act \to Act$$

определяемую следующим образом.

- $\forall\, \alpha \in Names \quad f(\alpha!) \stackrel{\text{def}}{=} f(\alpha)!$, и $f(\alpha?) \stackrel{\text{def}}{=} f(\alpha)?$
- $f(\tau) \stackrel{\text{def}}{=} \tau$

Теорема 7.
Пусть P - процесс вида

$$P \;=\; \Big(P_1[f_1] \,|\, \ldots \,|\, P_n[f_n]\Big) \setminus L$$

где для каждого $i \in \{1, \ldots, n\}$

$$P_i \sim \sum_{j=1}^{n_i} a_{ij}.\; P_{ij}$$

Тогда P сильно эквивалентен сумме
1. всех процессов вида

$$f_i(a_{ij}).\; \left(\left(\begin{array}{l} P_1[f_1] \,|\, \ldots \\ \ldots \,|\, P_{i-1}[f_{i-1}] \,|\, P_{ij}[f_i] \,|\, P_{i+1}[f_{i+1}] \,|\, \ldots \\ \ldots \,|\, P_n[f_n] \end{array} \right) \setminus L \right)$$

где $a_{ij} = \tau$ или $name(f_i(a_{ij})) \notin L$, и
2. всех процессов вида

$$\tau.\; \left(\left(\begin{array}{l} P_1[f_1] \,|\, \ldots \\ \ldots \,|\, P_{i-1}[f_{i-1}] \,|\, P_{ik}[f_i] \,|\, P_{i+1}[f_{i+1}] \,|\, \ldots \\ \ldots \,|\, P_{j-1}[f_{j-1}] \,|\, P_{jl}[f_j] \,|\, P_{j+1}[f_{j+1}] \,|\, \ldots \\ \ldots \,|\, P_n[f_n] \end{array} \right) \setminus L \right)$$

где $1 \le i < j \le n, \quad a_{ik}, a_{jl} \ne \tau$, и $f_i(a_{ik}) = \overline{f_j(a_{jl})}$.

Доказательство.
Данная теорема непосредственно следует из
- предыдущей теоремы,
- теоремы 3,
- свойств 6, 9, 10, 16 и 17 из параграфа 3.7, и
- первого утверждения из теоремы 4.

4.6 Распознавание сильной эквивалентности

4.6.1 Отношение $\mu(P_1, P_2)$

Пусть заданы два процесса
$P_i = (S_i, s_i^0, R_i) \qquad (i = 1,2)$

Определим функцию ' на множестве всех отношений из S_1 в S_2, которая сопоставляет каждому отношению $\mu \subseteq S_1 \times S_2$ отношение $\mu' \subseteq S_1 \times S_2$,определяемое следующим образом:

$$\mu' \stackrel{\text{def}}{=} \left\{ (s_1, s_2) \in S_1 \times S_2 \;\middle|\; \begin{array}{l} \forall a \in Act \\ \forall s_1' \in S_1 : (s_1 \xrightarrow{a} s_1') \in R_1 \\ \quad \exists s_2' \in S_2 : \left\{ \begin{array}{l} (s_2 \xrightarrow{a} s_2') \in R_2 \\ (s_1', s_2') \in \mu \end{array} \right. \\ \forall s_2' \in S_2 : (s_2 \xrightarrow{a} s_2') \in R_2 \\ \quad \exists s_1' \in S_1 : \left\{ \begin{array}{l} (s_1 \xrightarrow{a} s_1') \in R_1 \\ (s_1', s_2') \in \mu \end{array} \right. \end{array} \right\}$$

Нетрудно доказать, что для каждого $\mu \subseteq S_1 \times S_2$

μ удовлетворяет условиям 1 и 2 из определения БМ $\quad \Leftrightarrow \quad \mu \subseteq \mu'$

Следовательно,

$$\mu - \text{БМ между } P_1 \text{ и } P_2 \quad \Leftrightarrow \quad \left\{ \begin{array}{l} (s_1^0, s_2^0) \in \mu \\ \mu \subseteq \mu' \end{array} \right.$$

Нетрудно доказать, что функция ' монотонна, т.е.

если $\mu_1 \subseteq \mu_2$, то $\mu_1' \subseteq \mu_2'$.

Обозначим символом μ_{max} объединение всех отношений из

совокупности

$$\{\mu \subseteq S_1 \times S_2 \mid \mu \subseteq \mu'\} \quad (4.30)$$

Заметим, что отношение μ_{max} принадлежит совокупности (4.30), так как для каждого $\mu \in (4.30)$ из

- включения $\mu \subseteq (\bigcup_{\mu \in (4.30)} \mu) = \mu_{max}$, и
- монотонности функции ' следует, что для каждого $\mu \in (4.30)$

$$\mu \subseteq \mu' \subseteq \mu'_{max}$$

Поэтому $\mu_{max} = \bigcup_{\mu \in (4.30)} \mu \subseteq \mu'_{max}$, **т.е.** $\mu_{max} \in (4.30)$.

Заметим, что имеет место равенство

$$\mu_{max} = \mu'_{max}$$

так как из включения $\mu_{max} \subseteq \mu'_{max}$ и из монотонности функции ' следует включение

$$\mu'_{max} \subseteq \mu''_{max}$$

т.е. $\mu'_{max} \in (4.30)$, откуда, в силу максимальности μ_{max}, следует включение

$$\mu'_{max} \subseteq \mu_{max}$$

Таким образом, отношение μ_{max} является

- наибольшим элементом совокупности (4.30), и
- наибольшей неподвижной точкой функции '.

Мы будем обозначать это отношение знакосочетанием

$$\mu(P_1, P_2) \quad (4.31)$$

Из теоремы 2 следует, что

$$P_1 \sim P_2 \quad \Leftrightarrow \quad (s_1^0, s_2^0) \in \mu(P_1, P_2)$$

Из определения отношения $\mu(P_1, P_2)$ вытекает, что данное отношение состоит из всех пар $(s_1, s_2) \in S_1 \times S_2$, таких, что

$$P_1(s_1) \sim P_2(s_2)$$

Отношение $\mu(P_1, P_2)$ можно рассматривать как **меру близости** между P_1 и P_2.

4.6.2 Полиномиальный алгоритм распознавания сильной эквивалентности

Пусть P_1 и P_2 - процессы вида

$$P_i = (S_i, s_i^0, R_i) \quad (* = 1,2)$$

Если множества S_1 и S_2 конечны, то задача проверки истинности соотношения

$$P_1 \sim P_2 \quad (4.32)$$

очевидно является алгоритмически разрешимой: например, можно перебрать все отношения $\mu \subseteq S_1 \times S_2$ и для каждого из них проверить условия 0, 1 и 2 из определения БМ. Алгоритм заканчивает свою работу, когда

- нашлось отношение $\mu \subseteq S_1 \times S_2$ которое удовлетворяет условиям 0, 1 и 2 из определения БМ, в этом случае он выдаёт ответ

$$P_1 \sim P_2$$

или

- все отношения $\mu \subseteq S_1 \times S_2$ перебраны, и ни одно из них не удовлетворяет условиям 0, 1 и 2 из определения БМ, в этом случае он выдаёт ответ

$$P_1 \not\sim P_2$$

Если $P_1 \not\sim P_2$, то вышеприведённый алгоритм выдаст ответ после перебора всех отношений между S_1 и S_2, число которых $2^{|S_1| \cdot |S_2|}$, т.е. данный алгоритм имеет экспоненциальную сложность.

Данную задачу можно решить гораздо более эффективным алгоритмом, который имеет полиномиальную сложность. Для построения такого алгоритма мы рассмотрим следующую последовательность отношений между S_1 и S_2

$$\{\mu_i \mid i \geq 1\} \quad (4.33)$$

где $\mu_1 \stackrel{\text{def}}{=} S_1 \times S_2$, и $\forall\, i \geq 1 \quad \mu_{i+1} \stackrel{\text{def}}{=} \mu_i'$.

Из соотношения $\mu_1 \supseteq \mu_2$ и монотонности функции ' следует,

что

$$\mu_2 = \mu'_1 \supseteq \mu'_2 = \mu_3$$
$$\mu_3 = \mu'_2 \supseteq \mu'_3 = \mu_4$$

и т.д.

Таким образом, последовательность (4.33) монотонна:

$$\mu_1 \supseteq \mu_2 \supseteq \dots$$

Поскольку все члены последовательности (4.33) являются подмножествами конечного множества $S_1 \times S_2$, то данная последовательность не может бесконечно убывать, она стабилизируется на некотором члене, т.е. для некоторого $i \geq 1$ имеет место соотношение

$$\mu_i = \mu_{i+1} = \mu_{i+2} = \dots$$

Докажем, что член μ_i, на котором наступает стабилизация, совпадает с отношением $\mu(P_1, P_2)$.

- Т.к. $\mu_i = \mu_{i+1} = \mu'_i$, т.е. μ_i - неподвижная точка функции ', то

$$\mu_i \subseteq \mu(P_1, P_2) \quad (4.34)$$

поскольку $\mu(P_1, P_2)$ - наибольшая неподвижная точка функции '.

- Т.к. для каждого $j \geq 1$ имеет место включение

$$\mu(P_1, P_2) \subseteq \mu_j \quad (4.35)$$

поскольку

— включение (4.35) верно для $j = 1$, и

— если включение (4.35) верно для некоторого j, то, в силу монотонности функции ', имеем соотношения

$$\mu(P_1, P_2) = \mu(P_1, P_2)' \subseteq \mu'_j = \mu_{j+1}$$

т.е. включение (4.35) будет верно для $j + 1$

то, в частности, (4.35) верно для $j = i$.

Из (4.34) и (4.35) для $j = i$ следует равенство

$$\mu_i = \mu(P_1, P_2) \quad (4.36)$$

Таким образом, задача проверки истинности соотношения $P_1 \sim P_2$ может быть решена путём

- нахождения первого члена μ_i последовательности (4.33), который удовлетворяет условию $\mu_i = \mu_{i+1}$, и
- проверки для этого μ_i соотношения

$$(s_1^0, s_2^0) \in \mu_i \quad (4.37)$$

Алгоритм выдаёт ответ

$$P_1 \sim P_2$$

тогда и только тогда, когда выполняется (4.37).

Для вычисления членов последовательности (4.33) можно использовать нижеследующий алгоритм, который по отношению $\mu \subseteq S_1 \times S_2$ вычисляет отношение μ':

$\mu' := \emptyset$
цикл для каждого $(s_1, s_2) \in \mu$
| включить $:= \top$
| **цикл для каждого** $s_1', a: \ s_1 \xrightarrow{a} s_1'$
| | найдено $:= \bot$
| | **цикл для каждого** $s_2': \ s_2 \xrightarrow{a} s_2'$
| | | найдено $:=$ найдено $\vee (s_1', s_2') \in \mu$
| | **конец цикла**
| | включить $:=$ включить $\wedge$ найдено
| **конец цикла**
| **цикл для каждого** $s_2', a: \ s_2 \xrightarrow{a} s_2'$
| | найдено $:= \bot$
| | **цикл для каждого** $s_1': \ s_1 \xrightarrow{a} s_1'$
| | | найдено $:=$ найдено $\vee (s_1', s_2') \in \mu$
| | **конец цикла**
| | включить $:=$ включить $\wedge$ найдено
| **конец цикла**
| **если** включить **то** $\mu' := \mu' \cup \{(s_1, s_2)\}$
конец цикла

Заметим, что данный алгоритм корректен только в том случае, когда $\mu' \subseteq \mu$ (что имеет место в том случае, когда этот алгоритм используется для вычисления членов последовательности (4.33)). В общей ситуации внешний цикл должен иметь вид

цикл для каждого $(s_1, s_2) \in S_1 \times S_2$

Оценим сложность данного алгоритма. Обозначим символом A число

$$A \stackrel{\text{def}}{=} \max(|Act(P_1)|, |Act(P_2)|) + 1$$

- Внешний цикл делает не более $|S_1| \cdot |S_2|$ итераций.

• Оба цикла, содержащиеся в во внешнем цикле, делают не более $|S_1| \cdot |S_2| \cdot A$ итераций.

Поэтому сложность этого алгоритма можно оценить функцией

$$O(|S_1|^2 \cdot |S_2|^2 \cdot A)$$

Поскольку для вычисления того члена последовательности (4.33), на котором наступает её стабилизация, нужно вычислить не больше чем $|S_1| \cdot |S_2|$ членов этой последовательности, то, следовательно, искомое отношение $\mu_i = \mu(P_1, P_2)$ может быть вычислено за время

$$O(|S_1|^3 \cdot |S_2|^3 \cdot A)$$

4.7 Минимизация процессов

4.7.1 Свойства отношений вида $\mu(P, P)$

Теорема 8.

Для каждого процесса $P \stackrel{\text{def}}{=} (S, s^0, R)$ отношение $\mu(P, P)$ является эквивалентностью.

Доказательство.

1. Рефлексивность отношения $\mu(P, P)$ следует из того, что диагональное отношение

$$Id_S = \{(s, s) \mid s \in S\}$$

удовлетворяет условиям 1 и 2 из определения БМ, т.е.

$$Id_S \in (4.30).$$

2. Симметричность отношения $\mu(P, P)$ следует из того, что если отношение μ удовлетворяет условиям 1 и 2 из определения БМ, то обратное отношение μ^{-1} тоже удовлетворяет этим условиям, т.е.

$$\text{если } \mu \in (4.30), \text{ то } \mu^{-1} \in (4.30).$$

3. **Транзитивность** отношения $\mu(P, P)$ следует из того, что произведение

$$\mu(P, P) \circ \mu(P, P)$$

удовлетворяет условиям 1 и 2 из определения БМ, т.е.

$$\mu(P, P) \circ \mu(P, P) \subseteq \mu(P, P)$$

Обозначим символом $P_\sim$ процесс, компоненты которого имеют следующий вид.

• Множество состояний процесса $P_{\sim}$ представляет собой совокупность классов эквивалентности, на которые разбивается множество *S* по отношению $\mu(P,P)$.
• Начальным состоянием является класс $[s^0]$, который содержит начальное состояние s^0 процесса *P*.
• Множество переходов процесса $P_{\sim}$ состоит из всех переходов вида

$$[s_1] \xrightarrow{a} [s_2]$$

где $s_1 \xrightarrow{a} s_2$ - произвольный переход из *R*.

Процесс $P_{\sim}$ называется **фактор-процессом** процесса *P* по эквивалентности $\mu(P,P)$.

Теорема 9.

Для каждого процесса *P* отношение

$$\mu \stackrel{\text{def}}{=} \{ (s,[s]) \mid s \in S\}$$

является БМ между *P* и $P_{\sim}$.

Доказательство.

Проверим для μ свойства из определения БМ.

Свойство 0 верно по определению начального состояния процесса $P_{\sim}$. Свойство 1 верно по определению множества переходов процесса $P_{\sim}$.

Докажем свойство 2. Пусть $P_{\sim}$ содержит переход

$$[s] \xrightarrow{a} [s']$$

Докажем, что существует переход в *R* вида

$$s \xrightarrow{a} s''$$

такой, что $(s'',[s']) \in \mu$, **т.е.** $[s''] = [s']$,т.е.

$(s'',s') \in \mu(P,P)$

Из определения множества переходов процесса $P_{\sim}$ следует, что *R* содержит переход вида

$s_1 \xrightarrow{a} s_1'$ (4.38) где $[s_1] = [s]$ **и** $[s_1'] = [s']$, т.е.

$(s_1,s) \in \mu(P,P)$ **и**

$(s_1',s') \in \mu(P,P)$

Так как $\mu(P,P)$ - БМ, то из

- $(4.38) \in R$, и
- $(s_1, s) \in \mu(P, P)$

следует, что *R* содержит переход вида

$s \xrightarrow{a} s_1''$ (4.39) где $(s_1'', s_1') \in \mu(P, P)$.

Так как $\mu(P, P)$ транзитивно, то из

$(s_1'', s_1') \in \mu(P, P)$ и
$(s_1', s') \in \mu(P, P)$

следует

$$(s_1'', s') \in \mu(P, P)$$

Таким образом, в качестве искомого *s''* можно взять *s''₁*.
Из теоремы 9 следует, что для каждого процесса *P*

$P \sim P_{\sim}$

4.7.2 Минимальные процессы относительно ~

Процесс *P* называется **минимальным относительно** ~, если
• каждое его состояние достижимо, и

- $\mu(P, P) = Id_S$

(где *S* - множество состояний процесса *P).*
Ниже минимальные процессы относительно ~ называются просто **минимальными** процессами.
Теорема 10.
Пусть процессы P_1 и P_2 минимальны, и $P_1 \sim P_2$.
Тогда P_1 и P_2 изоморфны.
Доказательство.
Пусть P_i $(i = 1, 2)$ имеет вид (S_i, s_i^0, R_i), и пусть $\mu \subseteq S_1 \times S_2$ - БМ между P_1 и P_2.
Поскольку μ^{-1} тоже является БМ, и композиция двух БМ -тоже БМ, то

- $\mu \circ \mu^{-1}$ – БМ между P_1 и P_1
- $\mu^{-1} \circ \mu$ – БМ между P_2 и P_2

откуда, используя определение отношений $\mu(P_i, P_i)$, и определение минимальности процесса, получаем включения

$$\mu \circ \mu^{-1} \subseteq \mu(P_1, P_1) = Id_{S_1}$$
$$\mu^{-1} \circ \mu \subseteq \mu(P_2, P_2) = Id_{S_2} \quad (4.40)$$

Докажем, что отношение μ является функциональным, т.е. для каждого $s \in S_1$ существует единственный элемент $s' \in S_2$, такой, что $(s, s') \in \mu$.

- Если $s = s_1^0$, то полагаем $s' \stackrel{\text{def}}{=} s_2^0$.
- Если $s \neq s_1^0$, то, поскольку каждое состояние в P_1 достижимо, то в P_1 существует путь

$$s_1^0 \xrightarrow{a_1} \dots \xrightarrow{a_n} s$$

Так как μ - БМ, то в P_2 существует путь

$$s_2^0 \xrightarrow{a_1} \dots \xrightarrow{a_n} s'$$

причём $(s, s') \in \mu$.

Таким образом, в обоих случаях существует элемент $s' \in S_2$, такой, что $(s, s') \in \mu$.

Докажем единственность элемента s' со свойством $(s, s') \in \mu$. Если для некоторого элемента $s'' \in S_2$ имеет место соотношение $(s, s'') \in \mu$, то $(s'', s) \in \mu^{-1}$, откуда следует

$$(s'', s') \in \mu^{-1} \circ \mu = Id_{S_2}$$

Поэтому $s'' = s'$.

По аналогичным соображениям, отношение μ^{-1} тоже является функциональным.

Из условий (4.40) нетрудно вывести биективность функции, которая соответствует отношению μ. По определению БМ, отсюда вытекает изоморфность P_1 и P_2.

Теорема 11.

Пусть

- процесс P_2 получается из процесса P_1 удалением недостижимых состояний,и
- $P_3 \stackrel{\text{def}}{=} (P_2)_{\sim}$.

Тогда процесс P_3 минимален, и

$$P_1 \sim P_2 \sim P_3$$

Доказательство.
Так как каждое состояние в P_2 достижимо, то из определения множества переходов процесса вида $P_{\sim}$ следует, что каждое состояние Рз тоже достижимо.
Теперь докажем, что

$$\mu(P_3, P_3) = Id_{S_3} \quad (4.41)$$

т.е. предположим, что $(s', s'') \in \mu(P_3, P_3)$,и докажем, что $s' = s''$.
Из определения процесса вида $P_{\sim}$ следует, что существуют состояния $s_1, s_2 \in S_2$,такие, что

$$\begin{aligned} s' &= [s_1] \\ s'' &= [s_2] \end{aligned}$$

где $[\cdot]$ обозначает класс эквивалентности по отношению $\mu(P_2, P_2)$.
Из теоремы 9 следует, что

$$\begin{aligned} (s_1, s') &\in \mu(P_2, P_3) \\ (s'', s_2) &\in \mu(P_3, P_2) \end{aligned}$$

Поскольку композиция любых БМ тоже является БМ, то композиция

$$\mu(P_2, P_3) \circ \mu(P_3, P_3) \circ \mu(P_3, P_2) \quad (4.42)$$

является БМ между P_2 и P_2, поэтому

$$(4.42) \subseteq \mu(P_2, P_2) \quad (4.43)$$

Поскольку $(s_1, s_2) \in (4.42)$,то, ввиду (4.43), получаем:

$$s' = [s_1] = [s_2] = s''$$

В заключение отметим, что
- соотношение $P_1 \sim P_2$ тривиально, и
- соотношение $P_2 \sim$ Рз следует из теоремы 9.

4.7.3 Алгоритм минимизации процессов

Описанный в параграфе 4.6.2 алгоритм можно использовать также для решения задачи **минимизации конечных процессов,** которая заключается в том, чтобы по заданному конечному процессу *P* построить процесс с наименьшим числом состояний, который сильно эквивалентен *P*.
Для построения такого процесса сначала строится процесс *P′,* получаемый из *P* удалением недостижимых состояний. Искомый процесс имеет вид $P'_{\sim}$.
Множество состояний процесса *P′* может быть построено, например, следующим образом. Пусть *P* имеет вид

$$P = (S, s^0, R)$$

Рассмотрим последовательность подмножеств множества *S*

$$S_0 \subseteq S_1 \subseteq S_2 \subseteq \ldots \quad (4.44)$$

определяемую следующим образом.

- $S_0 \stackrel{\text{def}}{=} \{s^0\}$

• для каждого $i \geq 0$ множество S_{i+1} получается добавлением к S_i всех состояний $s' \in S$, таких, что

$$\exists s \in S,\ \exists a \in Act:\ (s \xrightarrow{a} s') \in R$$

Поскольку множество *S* по предположению конечно, то последовательность (4.44) не может неограниченно возрастать. Пусть S_i - тот член последовательности (4.44), на котором эта последовательность стабилизируется. Очевидно, что

• все состояния из S_i достижимы, и

• все состояния из $S \setminus S_i$ недостижимы.

Поэтому множеством состояний процесса *P'* является множество S_i. Пусть *S'* - множество состояний процесса *P'*. Заметим, что при вычислении отношения $\mu(P', P')$ требуется вычислить не более чем $|S'|$ членов последовательности (4.33). Это верно потому, что в данном случае каждое из отношений в последовательности (4.33) является эквивалентностью (так как если бинарное отношение μ на множестве состояний произвольного процесса является эквивалентностью, то отношение μ' тоже будет эквивалентностью). Поэтому каждый из членов последовательности (4.33) определяет некоторое разбиение множества *S'*, и для каждого $i \geq 1$, если $\mu_{i+1} \neq \mu_i$, то разбиение, соответствующее отношению μ_{i+1}, является измельчением разбиения, соответствующего отношению μ_i. Нетрудно показать, что таких измельчений может быть не больше, чем количество элементов в множестве *S'*.

Теорема 12.

Процесс $P'_{\sim}$ имеет наименьшее число состояний среди всех конечных процессов, которые сильно эквивалентны *P*.

Доказательство.

Пусть P_1 - некоторый конечный процесс, такой, что $P_1 \sim P$. Выделим из P_1 достижимую часть P'_1 и построим процесс $(P'_1)_{\sim}$. Как было установлено выше,

$$P_1 \sim P'_1 \sim (P'_1)_{\sim}$$

Кроме того, поскольку $P \sim P' \sim P'_{\sim}$ и $P \sim P_1$, то, следовательно,

$$P'_{\sim} \sim (P'_1)_{\sim} \quad (4.45)$$

Как было доказано в теореме 11, процессы $P'_{\sim}$ и $(P'_1)_{\sim}$ минимальны. Отсюда и из (4.45) по теореме 10 следует, что процессы $P'_{\sim}$ и $(P'_1)_{\sim}$ изоморфны. В частности, они имеют одинаковое число состояний. Поскольку

- число состояний процесса $(P'_1)_{\sim}$ ~ не превосходит числа состояний процесса P'_1, так как состояния процесса $(P'_1)_{\sim}$ являются классами разбиения множества состояний процесса P'_1 и
- число состояний процесса P'_1 не превосходит числа состояний процесса P_1, так как множество состояний процесса P'_1 является подмножеством множества состояний процесса P_1, то, следовательно, число состояний процесса $P'_{\sim}$ не превосходит числа состояний процесса P_1.

4.8 Наблюдаемая эквивалентность

4.8.1 Определение наблюдаемой эквивалентности

Ещё одним вариантом понятия эквивалентности процессов является **наблюдаемая эквивалентность.** Данное понятие используется в тех ситуациях, когда мы рассматриваем невидимое действие τ как несущественное, и считаем две трассы одинаковыми, если одна может быть получена из другой путём вставок и/или удалений невидимых действий τ.

Для определения понятия наблюдаемой эквивалентности мы введём вспомогательные обозначения.

Пусть P и P' - некоторые процессы.

1. Знакосочетание

$$P \xrightarrow{\tau^*} P' \quad (4.46)$$

означает, что

- или $P = P'$
- или существует последовательность процессов

$$P_1, \ldots, P_n \quad (n \geq 2)$$

таких, что

$$- \; P_1 = P, \quad P_n = P'$$

— для каждого $i = 1,..., n — 1$

$$P_i \xrightarrow{\tau} P_{i+1}$$

(4.46) можно интерпретировать как утверждение о том, что процесс *P,* может незаметным для наблюдателя образом превратиться в процесс *P'.*

2. Для каждого действия $a \in Act \setminus \{\tau\}$ знакосочетание

$$P \xrightarrow{a_\tau} P' \quad (4.47)$$

означает, что существуют процессы P_1 и P_2 со следующими свойствами:

$$P \xrightarrow{\tau^*} P_1, \quad P_1 \xrightarrow{a} P_2, \quad P_2 \xrightarrow{\tau^*} P'$$

(4.47) можно интерпретировать как утверждение о том, что процесс P, может

• некоторым образом эволюционировать, так, что внешним проявлением этой эволюции будет лишь исполнение действия *a,*

• после чего вести себя как процесс *P'.*

Если имеет место (4.47), то мы будем говорить, что процесс *P* может **наблюдаемо выполнить** *a*, и после этого вести себя как *P'*.

Понятие наблюдаемой эквивалентности основано на следующем понимании эквивалентности процессов: если мы рассматриваем процессы P_1 и P_2 как эквивалентные, то должны быть выполнены следующие условия.

1. • Если один из этих процессов P_i может незаметно превратиться в некоторый процесс P'_i,

• то и другой процесс $P_j \quad (j \in \{1,2\} \setminus \{i\})$ тоже должен обладать способностью незаметно превратиться в некоторый процесс P'_j, который эквивалентен P'_i.

2. • Если один из этих процессов P_i может

— наблюдаемо выполнить некоторое действие $a \in Act \setminus \{\tau\}$,

— и после этого вести себя к ак некоторый процесс P'_i

• то и другой процесс $P_j \quad (j \in \{1,2\} \setminus \{i\})$ должен обладать способностью

— наблюдаемо выполнить то же действие *a,*

— после чего вести себя как некоторый процесс P'_j, который эквивалентен P'_i .

Используя обозначения (4.46) и (4.47), вышеприведённое неформально описанное понятие наблюдаемой эквивалентности можно выразить равносильным образом как некоторое бинарное отношение μ на множестве всех процессов, обладающее следующими свойствами.

(1) Если $(P_1, P_2) \in \mu$, и для некоторого процесса P'$_i$ верно утверждение

$$P_1 \xrightarrow{\tau} P_1' \quad (4.48)$$

то должен существовать процесс P'_2 такой, что выполнены условия

$$P_2 \xrightarrow{\tau^*} P_2' \quad (4.49)$$

и

$$(P_1', P_2') \in \mu \quad (4.50)$$

(2) Симметричное свойство: если $(P_1, P_2) \in \mu$,и для некоторого процесса P'_2 верно утверждение

$$P_2 \xrightarrow{\tau} P_2' \quad (4.51)$$

то должен существовать процесс P'$_i$, такой, что выполнены условия

$$P_1 \xrightarrow{\tau^*} P_1' \quad (4.52)$$

и (4.50).

(3) Если $(P_1, P_2) \in \mu$, и для некоторого процесса P'$_i$ верно утверждение

$$P_1 \xrightarrow{a} P_1' \quad (4.53)$$

то должен существовать процесс P'$_2$, такой, что выполнены условия

$$P_2 \xrightarrow{a_\tau} P_2' \quad (4.54)$$

и (4.50).

(4) Симметричное свойство: если $(P_1, P_2) \in \mu$,и для некоторого процесса P'_2 верно утверждение

$$P_2 \xrightarrow{a} P_2' \quad (4.55)$$

то должен существовать процесс P'$_i$, такой, что выполнены условия

$$P_1 \xrightarrow{a_\tau} P_1' \quad (4.56)$$

и (4.50).

Обозначим символом $\mathcal{M}_\tau$ совокупность всех бинарных отношений, которые обладают выш еприведёнными свойствами.

Множество $\mathcal{M}_\tau$ непусто: оно содержит, например, диагональное отношение, которое состоит из всех пар вида *(P, P),* где *P* - произвольный процесс.

Как и в случае сильной эквивалентности, встаёт естественный вопрос о том, какое же из отношений, входящих в $\mathcal{M}_\tau$,можно использовать для определения понятия наблюдаемой эквивалентности.
Так же, как и в случае сильной эквивалентности, мы предлагаем следующий ответ на этот вопрос: мы будем считать P_1 и P_2 наблюдаемо эквивалентными в том и только в том случае, когда существует хотя бы одно отношение $\mu \in \mathcal{M}_\tau$, которое содержит пару (P_1,P_2), т.е.

искомое отношение наблюдаемой эквивалентности на множестве всех процессов мы определяем как объединение всех отношений из $\mathcal{M}_\tau$. Данное отношение обозначается символом ≈.
Нетрудно доказать, что

- $\approx \in \mathcal{M}_\tau$,

• ≈ является отношением эквивалентности, т.к.
— рефлексивность ≈ следует из того, что диагональное отношение принадлежит $\mathcal{M}_\tau$,

— симметричность ≈ следует из того, что если $\mu \in \mathcal{M}_\tau$,

то $\mu^{-1} \in \mathcal{M}_\tau$

— транзитивность ≈ следует из того, что если $\mu_1 \in \mathcal{M}_\tau$

и $\mu_2 \in \mathcal{M}_\tau$,то $\mu_1 \circ \mu_2 \in \mathcal{M}_\tau$.
Если процессы P_1 и P_2 наблюдаемо эквивалентны, то этот факт обозначается знакосочетанием

$$P_1 \approx P_2$$

Нетрудно доказать, что если процессы P_1 и P_2 сильно эквивалентны, то они наблюдаемо эквивалентны.

4.8.2 Логический критерий наблюдаемой эквивалентности

Логический критерий наблюдаемой эквивалентности аналогичен критерию из параграфа 4.4.1. В данном критерии используется то же самое множество формул. Понятие значения формулы на процессе отличается от аналогичного понятия в параграфе 4.4.1 лишь для формул вида $\langle a \rangle \varphi$:

• значение формулы $\langle \tau \rangle \varphi$ на процессе P равно

$$\begin{cases} 1, & \text{если существует процесс } P': \\ & P \xrightarrow{\tau^*} P',\ P'(\varphi) = 1 \\ 0, & \text{в противном случае} \end{cases}$$

- значение формулы $\langle a \rangle \varphi$ (где $a \neq \tau$) в *P* равно

$$\begin{cases} 1, & \text{если существует процесс } P' : \\ & P \xrightarrow{a_\tau} P',\ P'(\varphi) = 1 \\ 0, & \text{в противном случае} \end{cases}$$

Для каждого процесса *P* мы будем обозначать знакосочетанием $Th_\tau(P)$ совокупность всех формул, которые имеют на этом процессе значение 1 (относительно модифицированного определения понятия значения формулы на процессе).

Теорема 13.

Пусть процессы P_1 и P_2 конечны. Тогда

$$P_1 \approx P_2 \quad \Leftrightarrow \quad Th_\tau(P_1) = Th_\tau(P_2)$$

Как и в случае ~, представляет интерес задача нахождения по двум заданным процессам P_1 и P_2 списка формул

$$\varphi_1, \ldots, \varphi_n$$

как можно меньшего размера, таких, что $P_1 \approx P_2$ тогда и только тогда, когда

$$\forall\, i = 1, \ldots, n \qquad P_1(\varphi_i) = P_2(\varphi_i)$$

Используя теорему 13, можно легко доказать, что

$$\text{для каждого процесса } P \qquad P \approx \tau.P \quad (4.57)$$

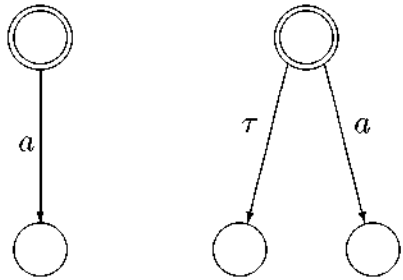
Заметим, что, согласно (4.57), имеет место соотношение

$$\mathbf{0} \approx \tau.\mathbf{0}$$

однако соотношение

$$\mathbf{0} + a.\mathbf{0} \approx \tau.\mathbf{0} + a.\mathbf{0} \quad (\text{где } a \neq \tau) \quad (4.58)$$

неверно, в чём нетрудно убедиться при рассмотрении графового представления левой и правой частей в (4.58):



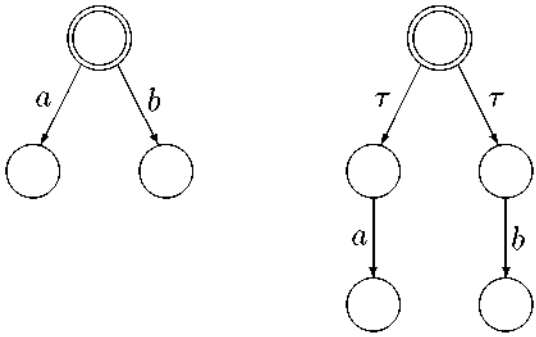
Формула, которая принимает разные значения на этих процессах, может иметь, например, такой вид:

$$\neg\langle\tau\rangle\neg\langle a\rangle\top$$

Таким образом, отношение ≈ не является конгруэнцией, т.к. оно не сохраняет операцию +.
Другой пример: если $a, b \in Act \setminus \{\tau\}$ и $a \neq b$,то

$$a.\mathbf{0} + b.\mathbf{0} \not\approx \tau.a.\mathbf{0} + \tau.b.\mathbf{0}$$

Хотя $a.\mathbf{0} \approx \tau.a.\mathbf{0}$ и $b.\mathbf{0} \approx \tau.b.\mathbf{0}$.
Графовое представление этих процессов имеет вид



Отсутствие наблюдаемой эквивалентности между этими процессами обосновывается формулой

$$\langle\tau\rangle\neg\langle a\rangle\top$$

4.8.3 Критерий наблюдаемой эквивалентности, основанный на понятии наблюдаемого БМ

Для отношения ≈ также имеет место аналог критерия, основанного на понятии БМ (теорема 2 из параграфа 4.4.2). Для его формулировки мы введём вспомогательные обозначения.
Пусть $P = (S, s^0, R)$- некоторый процесс, и s_1, s_2 - пара его состояний. Тогда
• знакосочетание

$$s \xrightarrow{\tau^*} s'$$

означает, что
— или *s* = *s'*,

— или существует последовательность состояний

$$s_1, \ldots, s_n \quad (n \geq 2)$$

такая, что $s_1 = s$, $s_n = s'$, и $\forall i = 1, \ldots, n-1$

$$(s_i \xrightarrow{\tau} s_{i+1}) \in R$$

• знакосочетание

$$s \xrightarrow{a\tau} s' \quad (\text{где } a \neq \tau)$$

означает, что существуют состояния s_1 и s_2, такие, что

$$s \xrightarrow{\tau^*} s_1, \quad s_1 \xrightarrow{a} s_2, \quad s_2 \xrightarrow{\tau^*} s'.$$

Теорема 14.

Пусть заданы два процесса

$$P_i = (S_i, s_i^0, R_i) \qquad (i = 1, 2)$$

$P_1 \approx P_2$ тогда и только тогда, когда существует отношение

$$\mu \subseteq S_1 \times S_2$$

удовлетворяющее следующим условиям.

0. $(s_1^0, s_2^0) \in \mu$.

1. Для каждой пары $(s_1, s_2) \in \mu$ и каждого перехода из R_1 вида

$$s_1 \xrightarrow{\tau} s_1'$$

существует состояние $s_2' \in S_2$, такое, что

$$s_2 \xrightarrow{\tau^*} s_2'$$

и

$$(s_1', s_2') \in \mu \quad (4.59)$$

2. Для каждой пары $(s_1, s_2) \in \mu$ и каждого перехода из R_2 вида

$$s_2 \xrightarrow{\tau} s_2'$$

существует состояние $s_1' \in S_1$, такое, что

$$s_1 \xrightarrow{\tau^*} s_1'$$

и (4.59).

3. Для каждой пары $(s_1, s_2) \in \mu$ и каждого перехода из R_1 вида

$$s_1 \xrightarrow{a} s_1' \quad (a \neq \tau)$$

существует состояние $s_2' \in S_2$, такое, что

$$s_2 \xrightarrow{a_\tau} s_2'$$

и (4.59).

4. Для каждой пары $(s_1, s_2) \in \mu$ и каждого перехода из R_2 вида

$$s_2 \xrightarrow{a} s_2' \quad (a \neq \tau)$$

существует состояние $s_1' \in S_1$, такое, что

$$s_1 \xrightarrow{a_\tau} s_1'$$

и (4.59).

Отношение μ, удовлетворяющее данным условиям, называется **наблюдаемым БМ (НБМ)** между P_1 и P_2.

4.8.4 Алгебраические свойства наблюдаемой эквивалентности

Теорема 15.

Отношение наблюдаемой эквивалентности сохраняет все операции на процессах, за исключением операции +, т.е. если $P_1 \approx P_2$, то

- для каждого $a \in Act \quad a.P_1 \approx a.P_2$
- для каждого процесса $P \quad P_1|P \approx P_2|P$
- для каждого $L \subseteq Names \quad P_1 \setminus L \approx P_2 \setminus L$
- для каждого переименования $f \quad P_1[f] \approx P_2[f]$

Доказательство.

Как было установлено в параграфе 4.8.3, соотношение $P_1 \approx P_2$ эквивалентно тому, что существует НБМ μ, между P_1 и P_2. Используя это μ, мы построим НБМ для обоснования каждого из вышеприведённых соотношений.

•Пусть символы $s^0_{(1)}$ **и** $s^0_{(2)}$ обозначают начальные состояния $a.P_1$ **и** $a.P_2$ соответственно.

Тогда отношение

$$\mu \cup \{(s^0_{(1)}, s^0_{(2)})\}$$

является НБМ между $a.P_1$ и $a.P_2$.

• Пусть символ *S* обозначает множество состояний процесса *P.* Тогда отношение

$$\{((s_1, s), (s_2, s)) \mid (s_1, s_2) \in \mu,\ q \in S\}$$

является НБМ между $P_1|P$ и $P_2|P$.

• Отношение μ является НБМ

– между $P_1 \setminus L$ и $P_2 \setminus L$, и

– между $P_1[f]$ и $P_2[f]$.

4.8.5 Распознавание наблюдаемой эквивалентности и минимизация процессов относительно ≈

Для решения задач
1. распознавания для двух заданных конечных процессов, являются ли они наблюдаемо эквивалентными, и
2. построения по заданному конечному процессу *P* такого процесса *P',* который имеет наименьшее число состояний среди всех процессов, наблюдаемо эквивалентных *P*

могут быть построены теория и основанные на ней алгоритмы, которые аналогичны теории и алгоритмам, изложенным в параграфах 4.6 и 4.7. Мы не будем детально излагать эту теорию, т.к. она почти дословно повторяет соответствующую теорию для случая ~. В этой теории для произвольной пары процессов

$$P_i = (S_i, s_i^0, R_i) \qquad (i = 1, 2)$$

тоже определяется функция на отношениях из $S_1 \times S_2$, которая сопоставляет каждому отношению μ некоторое отношение μ'_τ, такое, что

$$\mu \text{ удовлетворяет условиям 1, 2, 3, 4 из определения НБМ} \quad \Leftrightarrow \quad \mu \subseteq \mu'_\tau$$

В частности,

$$\mu - \text{НБМ между } P_1 \text{ и } P_2 \quad \Leftrightarrow \quad \begin{cases} (s_1^0, s_2^0) \in \mu \\ \mu \subseteq \mu'_\tau \end{cases}$$

Обозначим символом $\mu_\tau(P_1, P_2)$ объединение всех отношений из совокупности

$$\{\mu \subseteq S_1 \times S_2 \mid \mu \subseteq \mu'_\tau\} \quad (4.60)$$

Данное отношение является наибольшим элементом совокупности (4.60), и обладает свойством

$$P_1 \approx P_2 \quad \Leftrightarrow \quad (s_1^0, s_2^0) \in \mu_\tau(P_1, P_2)$$

Из определения отношения $\mu_\tau(P_1, P_2)$ вытекает, что оно состоит из всех пар $(s_1, s_2) \in S_1 \times S_2$, таких, что

$$P_1(s_1) \approx P_2(s_2)$$

Отношение $\mu_\tau(P_1, P_2)$ можно рассматривать как ещё одну меру близости между P_1 и P_2.

При построении полиномиального алгоритма вычисления отношения $\mu_\tau(P_1, P_2)$, аналогичного алгоритму из параграфа 4.6.2, следует учитывать следующее соображение. Всякий раз, когда для заданной пары s, s' состояний некоторого процесса P требуется проверить условие

$$s \xrightarrow{\tau^*} s'$$

достаточно анализировать последовательности переходов вида

$$s \xrightarrow{\tau} s_1 \xrightarrow{\tau} s_2 \xrightarrow{\tau} \dots$$

длина которых не превосходит числа состояний процесса P.

4.8.6 Другие критерии эквивалентности процессов

Доказать сильную или наблюдаемую эквивалентность процессов P_1 и P_2 можно также с помощью излагаемых ниже критериев. В некоторых случаях использование этих критериев гораздо проще всех других способов доказательства соответствующей эквивалентности P_1 и P_2.

Бинарное отношение μ, на множестве процессов называется

- БМ $(\mathrm{mod} \sim)$, если $\mu \subseteq (\sim \mu \sim)'$
- НБМ $(\mathrm{mod} \sim)$, если $\mu \subseteq (\sim \mu \sim)'_\tau$
- НБМ $(\mathrm{mod} \approx)$, если $\mu \subseteq (\approx \mu \approx)'_\tau$

Нетрудно доказать, что

- если μ – БМ $(\mathrm{mod} \sim)$, то $\mu \subseteq \sim$, и
- если μ – НБМ $(\mathrm{mod} \sim$ или $\mathrm{mod} \approx)$, то $\mu \subseteq \approx$.

Таким образом, для доказательства $P_1 \sim P_2$ или $P_1 \approx P_2$ достаточно найти подходящее

- БМ (mod $\sim$), или
- НБМ (mod $\sim$ или mod $\approx$)

соответственно, такое, что

$(P_1, P_2) \in \mu$

4.9 Наблюдаемая конгруэнция

4.9.1 Мотивировка понятия наблюдаемой конгруэнции

Понятие эквивалентности процессов может быть определено неоднозначно. В предыдущих параграфах уже были рассмотрены различные виды эквивалентности процессов, каждый из которых отражал определённую точку зрения на то, какие виды поведения следует считать одинаковыми. В добавление к этим понятиям эквивалентности процессов, можно ещё определить, например, такие эквивалентности, которые

• учитывают длительность исполнения действий, т.е., в частности, одно из условий эквивалентности процессов P_1 и P_2 может иметь следующий вид:

— если один из этих процессов P_i может в течение некоторого промежутка времени незаметно превратиться в процесс P'_i ,

— то и другой процесс $P_j \quad (j \in \{1,2\} \setminus \{i\})$ тоже должен обладать способностью в течение примерно такого же промежутка времени незаметно превратиться в процесс P'_j который эквивалентен P'_i

(понятие "примерно такого же промежутка времени" может уточняться различным образом)

• или учитывают свойство **справедливости (fairness)** в поведении процессов, т.е. не позволяют рассматривать как эквивалентные такие два процесса,

— один из которых обладает свойством справедливости,

— а другой - не обладает

где одно из определений свойства справедливости имеет следующий вид: процесс называется **справедливым,** если не существует бесконечной последовательности переходов этого процесса вида

$$s_0 \xrightarrow{\tau} s_1 \xrightarrow{\tau} s_2 \xrightarrow{\tau} \dots$$

такой, что состояние s_0 достижимо, и для каждого $i \geq 0$

$$Act(s_i) \setminus \{\tau\} \neq \emptyset$$

отметим, что отношение наблюдаемой эквивалентности не учитывает свойство справедливости: существуют два процесса P_1 и P_2, такие, что $P_1 \approx P_2$, но P_1 обладает свойством справедливости, а P_2 не обладает этим свойством, например

— в качестве P_1 можно взять процесс $a.\mathbf{0}$, где $a \neq \tau$,

— а в качестве P_2 - процесс $a.\mathbf{0} \mid \tau^*$, где процесс τ^* имеет одно состояние и один переход с меткой τ

• и т.д.

Решение о том, какое именно из понятий эквивалентности между процессами следует выбрать в конкретной ситуации, существенно зависит от целей, для достижения которых предназначено данное понятие.

В настоящем параграфе мы определяем ещё один вид эквивалентности процессов, называемый **наблюдаемой конгруэнцией,**

Данная эквивалентность обозначается символом $\overset{+}{\approx}$. Мы определяем эту эквивалентность, исходя из следующих условий, которым она должна удовлетворять.

1. Процессы, находящиеся в отношении $\overset{+}{\approx}$, должны быть наблюдаемо эквивалентными.
2. Пусть

• процесс P построен в виде композиции процессов

$$P_1, \ldots, P_n$$

в которой используются операции

$$a., \quad +, \quad |, \quad \setminus L, \quad [f] \qquad (4.61)$$

• и мы решили заменить одну из частей этой композиции (например, P_i), на другой процесс P'_i, который мы считаем

— эквивалентным компоненте P_i, но

— более предпочтительным для нас по некоторым причинам, чем P_i (например, P'_i имеет меньшую сложность, чем P_i).

Мы хотели бы, чтобы процесс, получаемый из P в результате такой замены, был бы эквивалентен исходному процессу P.

Нетрудно доказать, что эквивалентность μ на множестве процессов удовлетворяет сформулированным выше условиям тогда и только тогда, когда

$$\begin{cases} \mu \subseteq \approx \\ \mu \text{ является конгруэнцией} \\ \quad \text{относительно операций } (4.61) \end{cases} \quad (4.62)$$

Условия (4.62) определяют искомую эквивалентность неоднозначно. Например, данным условиям удовлетворяют

- тождественное отношение (состоящее из пар вида *(P,P)),* и
- сильная эквивалентность (~).

Ниже мы докажем, что среди всех эквивалентностей, удовлетворяющих условиям (4.62), существует наибольшая (относительно включения). Вполне естественно считать именно эту эквивалентность искомой эквивалентностью.

4.9.2 Определение понятия наблюдаемой конгруэнции

Для определения понятия наблюдаемой конгруэнции мы введём вспомогательное обозначение.

Пусть *P* и *P′* - некоторые процессы. Знакосочетание

$$P \xrightarrow{\tau^+} P'$$

означает, что существует последовательность процессов

$$P_1, \ldots, P_n \quad (n \geq 2)$$

таких, что

- $P_1 = P, \quad P_n = P'$
- для каждого $i = 1, \ldots, n-1$

$$P_i \xrightarrow{\tau} P_{i+1}$$

Мы будем говорить, что процессы P_1 и P_2 находятся в отношении **наблюдаемой конгруэнции,** и обозначать этот факт знакосочетанием

$$P_1 \overset{+}{\approx} P_2$$

если выполнены следующие условия.

(0) $P_1 \approx P_2$.

(1) Если для некоторого процесса P'_1 верно утверждение

$$P_1 \xrightarrow{\tau} P'_1 \quad (4.63)$$

то существует процесс P'_2, такой, что

$$P_2 \xrightarrow{\tau^+} P'_2 \quad (4.64)$$

и

$$P'_1 \approx P'_2 \quad (4.65)$$

(2) Симметричное условие: если для некоторого процесса P'_2 верно утверждение

$$P_2 \xrightarrow{\tau} P'_2 \quad (4.66)$$

то существует процесс P'_1, такой, что

$$P_1 \xrightarrow{\tau^+} P'_1 \quad (4.67)$$

и (4.65).

Нетрудно доказать, что наблюдаемая конгруэнция является отношением эквивалентности.

4.9.3 Логический критерий наблюдаемой конгруэнтности

Логический критерий наблюдаемой конгруэнтности двух процессов получается небольшой модификацией логического критерия наблюдаемой эквивалентности, из параграфа 4.8.2.

Множество формул Fm^+, используемых в данном критерии, является расширением множества формул Fm из параграфа 4.4.2 путём использования дополнительной модальной связки $\langle\tau^+\rangle$:

- каждая формула из Fm принадлежит Fm^+, и
- для каждой формулы $\varphi \in Fm$ знакосочетание

$$\langle\tau^+\rangle\varphi$$

является формулой из множества Fm^+.

Для каждой формулы $\varphi \in Fm^+$ её значение на процессе P определяется следующим образом.

- Если $\varphi \in Fm$, то её значение в P определяется так же, как в параграфе 4.8.2.
- Если $\varphi = \langle\tau^+\rangle\psi$, где $\psi \in Fm$, то

$$P(\varphi) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{если существует процесс } P': \\ & P \xrightarrow{\tau^+} P',\ P'(\psi) = 1 \\ 0, & \text{в противном случае} \end{cases}$$

Для каждого процесса P мы будем обозначать знакосочетанием $Th^+_\tau(P)$ совокупность всех формул из Fm^+, которые имеют на этом процессе значение **1.**

Теорема 16.

Пусть процессы P_1 и P_2 конечны. Тогда

$$P_1 \overset{+}{\approx} P_2 \quad \Leftrightarrow \quad Th_\tau^+(P_1) = Th_\tau^+(P_2)$$

Как и в случаях ~ и ≈, представляет интерес задача нахождения по двум заданным процессам P_1 и P_2 списка формул

$$\varphi_1, \ldots, \varphi_n \in Fm^+$$

как можно меньшего размера, таких, что $P_1 \overset{+}{\approx} P_2$ тогда и только тогда, когда

$$\forall\, i = 1, \ldots, n \qquad P_1(\varphi_i) = P_2(\varphi_i)$$

4.9.4 Критерий наблюдаемой конгруэнтности, основанный на понятии НБМ

Введём вспомогательное обозначение. Пусть

- P – процесс вида (S, s^0, R), и
- s_1, s_2 – пара состояний из S.

Тогда знакосочетание

$$s \xrightarrow{\tau^+} s'$$

означает, что существует последовательность состояний

$$s_1, \ldots, s_n \quad (n \geq 2)$$

такая, что $s_1 = s$, $s_n = s'$, и для каждого $i = 1, \ldots, n-1$

$$(s_i \xrightarrow{\tau} s_{i+1}) \in R$$

Теорема 17.
Пусть заданы два процесса

$$P_i = (S_i, s_i^0, R_i) \qquad (i = 1, 2)$$

$P_1 \overset{+}{\approx} P_2$ тогда и только тогда, когда существует отношение

$$\mu \subseteq S_1 \times S_2$$

удовлетворяющее следующим условиям.
0.μ - НБМ между P_1 и P_2
(понятие НБМ изложено в параграфе 4.8.3).
1. Для каждого перехода из R_1 вида

$$s_1^0 \xrightarrow{\tau} s_1'$$

существует состояние $s_2' \in S_2$,такое, что

$$s_2^0 \xrightarrow{\tau^+} s_2'$$

и

$$(s_1', s_2') \in \mu \quad (4.68)$$

2. Для каждого перехода из R_2 вида

$$s_2^0 \xrightarrow{\tau} s_2'$$

существует состояние $s_1' \in S_1$, такое, что

$$s_1^0 \xrightarrow{\tau^+} s_1'$$

и (4.68).

Ниже знакосочетание НБМ$^+$ является сокращённой записью фразы "НБМ, удовлетворяющее условиям 1 и 2 теоремы 17".

4.9.5 Алгебраические свойства наблюдаемой конгруэнции

Теорема 18.

Наблюдаемая конгруэнция действительно является конгруэнцией, т.е. если $P_1 \overset{+}{\approx} P_2$,то

- для каждого $a \in Act \quad a.P_1 \overset{+}{\approx} a.P_2$
- для каждого процесса $P \quad P_1 + P \overset{+}{\approx} P_2 + P$
- для каждого процесса $P \quad P_1|P \overset{+}{\approx} P_2|P$
- для каждого $L \subseteq Names \quad P_1 \setminus L \overset{+}{\approx} P_2 \setminus L$
- для каждого переименования $f \quad P_1[f] \overset{+}{\approx} P_2[f]$

Доказательство.

Как было установлено в параграфе 4.9.4, соотношение $P_1 \overset{+}{\approx} P_2$ эквивалентно тому, что существует НБМ$^+$ μ между P_1 и P_2. Используя это μ, для обоснования каждого из вышеприведённых соотношений мы построим соответствующее НБМ$^+$.

•Пусть символыобозначают $s_{(1)}^0$ и $s_{(2)}^0$ начальные состояния

$a.P_1$ **и** $a.P_2$ соответственно.

Тогда отношение

$$\{(s^0_{(1)}, s^0_{(2)})\} \cup \mu$$

является НБМ$^+$ между $a.P_1$ **и** $a.P_2$

• Пусть

—символы $s^0_{(1)}$ **и** $s^0_{(2)}$ обозначают начальные состояния

$P_1 + P$ **и** $P_2 + P$ соответственно, и

— символ *S* обозначает множество состояний процесса *P*.

Тогда отношение

$$\{(s^0_{(1)}, s^0_{(2)})\} \cup \mu \cup Id_S$$

является НБМ$^+$ между $P_1 + P$ **и** $P_2 + P$

• Пусть символ *S* обозначает множество состояний процесса *P*. Тогда отношение

$$\{((s_1, s), (s_2, s)) \mid (s_1, s_2) \in \mu,\ q \in S\}$$

является НБМ$^+$ между $P_1|P$ **и** $P_2|P$

• Отношение μ является НБМ$^+$

– между $P_1 \setminus L$ **и** $P_2 \setminus L$, **и**

– между $P_1[f]$ **и** $P_2[f]$.

Теорема 19.

Для любых процессов P_1 и P_2

$$P_1 \approx P_2 \quad \Leftrightarrow \quad \begin{cases} P_1 \overset{+}{\approx} P_2 & \textbf{или} \\ P_1 \overset{+}{\approx} \tau.P_2 & \textbf{или} \\ \tau.P_1 \overset{+}{\approx} P_2 \end{cases}$$

Доказательство.

Импликация "$\Leftarrow$" следует из включения $\overset{+}{\approx} \subseteq \approx$, и того, что

для любого процесса P $\quad P \approx \tau.P$ (4.69)

Докажем импликацию "$\Rightarrow$". Предположим, что

$P_1 \approx P_2$ (4.70)

и

неверно, что $P_1 \overset{+}{\approx} P_2$ (4.71)

(4.71) может иметь место, например, в следующем случае:

$$\text{существует процесс } P_1', \text{ такой, что } P_1 \xrightarrow{\tau} P_1' \tag{4.72}$$

и

$$\text{не существует процесса } P_2' \approx P_1', \text{ такого, что } P_2 \xrightarrow{\tau^+} P_2' \tag{4.73}$$

Докажем, что в этом случае имеет место соотношение

$$P_1 \overset{+}{\approx} \tau.P_2$$

Согласно определению наблюдаемой конгруэнции, надо доказать, что выполнены следующие условия.

$$(0)\ \ P_1 \approx \tau.P_2.$$

Это условие следует из (4.70) и (4.69).

(1) Если для некоторого процесса P'_1 верно утверждение

$$P_1 \xrightarrow{\tau} P_1' \quad (4.74)$$

то для некоторого процесса $P_2' \approx P_1'$ верно утверждение

$$\tau.P_2 \xrightarrow{\tau^+} P_2' \quad (4.75)$$

Из (4.70), (4.74), и из определения наблюдаемой эквивалентности следует, что для некоторого процесса $P_2' \approx P_1'$ верно утверждение

$$P_2 \xrightarrow{\tau^*} P_2' \quad (4.76)$$

(4.75) следует из $\tau.P_2 \xrightarrow{\tau} P_2$ и (4.76).

(2) Если для некоторого процесса P'_2 верно утверждение

$$\tau.P_2 \xrightarrow{\tau} P_2' \quad (4.77)$$

то для некоторого процесса $P_1' \approx P_2'$ верно утверждение

$$P_1 \xrightarrow{\tau^+} P_1'$$

Из определения операции префиксного действия и из (4.77) следует, что

$$P_2' = P_2$$

Таким образом, надо доказать, что

$$\text{для некоторого процесса } P_1' \approx P_2 \text{ верно утверждение } P_1 \xrightarrow{\tau^+} P_1' \tag{4.78}$$

Пусть P'_1 есть в точности тот процесс, который упоминается в предположении (4.72). Из предположения (4.70) следует, что

$$\begin{array}{l}\text{существует процесс } P'_2 \approx P'_1, \\ \text{такой, что } P_2 \xrightarrow{\tau^*} P'_2\end{array} \quad (4.79)$$

Сопоставляя (4.79) и (4.73), получаем, $P'_2 = P_2$, т.е. мы доказали (4.78). Другая причина, по которой может иметь место (4.71) заключается в том, что

- существует процесс P_2, такой, что $P_2 \xrightarrow{\tau} P'_2$, и
- не существует процесса $P'_1 \approx P'_2$, такого, что

$$P_1 \xrightarrow{\tau^+} P'_1$$

В этом случае аналогичными рассуждениями можно доказать, что имеет место соотношение

$$\tau.P_1 \overset{+}{\approx} P_2$$

Теорема 20.

Отношение $\overset{+}{\approx}$ совпадает с отношением

$$\{(P_1, P_2) \mid \forall\, P \quad P_1 + P \approx P_2 + P\} \quad (4.80)$$

Доказательство.

Включение $\overset{+}{\approx} \subseteq (4.80)$ следует из того, что

- $\overset{+}{\approx}$ - конгруэнция (т.е., в частности, $\overset{+}{\approx}$ « сохраняет операцию **+**),

и

- $\overset{+}{\approx} \subseteq \approx$.

Докажем включение (4.80) $\subseteq \overset{+}{\approx}$.

Пусть $(P_1, P_2) \in (4.80)$.

Поскольку для каждого процесса P имеет место соотношение

$$P_1 + P \approx P_2 + P \quad (4.81)$$

то, полагая в (4.81) $P \overset{\text{def}}{=} \mathbf{0}$, имеем:

$$P_1 + \mathbf{0} \approx P_2 + \mathbf{0} \quad (4.82)$$

Поскольку

- для любого процесса P имеет место свойство

$$P + \mathbf{0} \sim P$$

- и, кроме того, $\sim \subseteq \approx$

то из (4.82) следует, что

$$P_1 \approx P_2 \quad (4.83)$$

Если неверно, что $P_1 \overset{+}{\approx} P_2$, то из (4.83) по теореме 19 следует, что

- **либо** $P_1 \overset{+}{\approx} \tau.P_2$
- **либо** $\tau.P_1 \overset{+}{\approx} P_2$

Рассмотрим, например, случай

$$P_1 \overset{+}{\approx} \tau.P_2 \quad (4.84)$$

(другой случай разбирается аналогично).

Так как $\overset{+}{\approx}$ - конгруэнция, то из (4.84) следует, что для любого процесса *P*

$$P_1 + P \overset{+}{\approx} \tau.P_2 + P \quad (4.85)$$

Из (4.81), (4.85) и включения $\overset{+}{\approx} \subseteq \approx$ следует, что для любого процесса *P*

$$P_2 + P \approx \tau.P_2 + P \quad (4.86)$$

Докажем, что имеет место соотношение

$$P_2 \overset{+}{\approx} \tau.P_2 \quad (4.87)$$

(4.87) равносильно существованию процесса $P_2' \approx P_2$, такой, что

$$P_2 \xrightarrow{\tau^+} P_2' \quad (4.88)$$

Выберем произвольное действие $b \in Act \setminus \{\tau\}$, которое не входит в P_2 (здесь мы используем предположение из параграфа 2.3 о том, что множество *Names*, а значит, и множество *Act*, является бесконечным).

Соотношение (4.86) должно быть верно в случае, когда *P* имеет вид b.**0**, т.е. должно быть верно соотношение

$$P_2 + b.\mathbf{0} \approx \tau.P_2 + b.\mathbf{0} \quad (4.89)$$

Так как имеет место соотношение

$$\tau.P_2 + b.\mathbf{0} \xrightarrow{\tau} P_2$$

то из (4.89) по определению отношения ≈ следует, что для некоторого процесса $P_2' \approx P_2$ имеет место соотношение

$$P_2 + b.\mathbf{0} \xrightarrow{\tau^*} P_2' \quad (4.90)$$

Случай $P_2 + b.\mathbf{0} = P_2'$ невозможен, так как процесс в левой части этого равенства содержит действие, которое не содержит процесс в правой части этого равенства. Согласно (4.90), отсюда следует соотношение

$$P_2 + b.\mathbf{0} \xrightarrow{\tau^+} P_2' \quad (4.91)$$

Из определения операции + следует, что (4.91) возможно в том и только в том случае, когда имеет место (4.88).
Таким образом, мы доказали, что для некоторого процесса $P'_2 \approx P_2$ имеет место (4.88), т.е. мы доказали (4.87).
Из (4.84) и (4.87) следует, что $P_1 \overset{+}{\approx} P_2$.

Теорема 21.

$\overset{+}{\approx}$ является наибольшей конгруэнцией, содержащейся в $\approx$, т.е. для каждой конгруэнции ν на множестве всех процессов имеет место импликация:

$$\nu \subseteq \approx \quad \Rightarrow \quad \nu \subseteq \overset{+}{\approx}$$

Доказательство.

Докажем, что если $(P_1, P_2) \in \nu$, то $P_1 \overset{+}{\approx} P_2$.
Пусть $(P_1, P_2) \in \nu$. Так как ν - конгруэнция, то

$$\text{для каждого процесса } P \quad (P_1 + P, P_2 + P) \in \nu \quad (4.92)$$

Если $\nu \subseteq \approx$, то из (4.92) следует, что

$$\text{для каждого процесса } P \quad P_1 + P \approx P_2 + P \quad (4.93)$$

Согласно теореме 20, из (4.93) следует, что $P_1 \overset{+}{\approx} P_2$.

Теорема 22.

Для отношений $\sim$, $\approx$ и $\overset{+}{\approx}$ верны включения

$$\sim \subseteq \overset{+}{\approx} \subseteq \approx \quad (4.94)$$

Доказательство.

Включение $\overset{+}{\approx} \subseteq \approx$ верно по определению $\overset{+}{\approx}$.
Включение $\sim \subseteq \overset{+}{\approx}$ следует

- из включения $\sim \subseteq \approx$, и
- из того, что если $P_1 \sim P_2$, то данная пара удовлетворяет условиям, изложенным в определении отношения $\overset{+}{\approx}$.

Отметим, что оба включения в (4.94) - собственные:

- $a.\tau.\mathbf{0} \not\approx a.\mathbf{0}$, но $a.\tau.\mathbf{0} \overset{+}{\approx} a.\mathbf{0}$
- $\tau.\mathbf{0} \overset{+}{\not\approx} \mathbf{0}$, но $\tau.\mathbf{0} \approx \mathbf{0}$

Теорема 23.

1. Если $P_1 \approx P_2$, то для каждого $a \in Act$

$$a.P_1 \overset{+}{\approx} a.P_2$$

В частности, для каждого процесса *P*

$$a.\tau.P \overset{+}{\approx} a.P \quad (4.95)$$

2. Для любого процесса *P*

$$P + \tau.P \overset{+}{\approx} \tau.P \quad (4.96)$$

3. Для любых процессов P_1 и P_2 и любого $a \in Act$

$$a.(P_1 + \tau.P_2) + a.P_2 \overset{+}{\approx} a.(P_1 + \tau.P_2) \quad (4.97)$$

4. Для любых процессов P_1 и P_2

$$P_1 + \tau.(P_1 + P_2) \overset{+}{\approx} \tau.(P_1 + P_2) \quad (4.98)$$

Доказательство.

Для каждого из доказываемых соотношений мы построим НМБ$^+$ между его левой и правой частями.

1. Как было установлено в теореме 14 из параграфа 4.8.3, соотношение $P_1 \approx P_2$ эквивалентно тому, что существует НБМ μ между P_1 и P_2.

Пусть символы $s^0_{(1)}$ и $s^0_{(2)}$ обозначают начальные состояния $a.P_1$ и $a.P_2$ соответственно.

Тогда отношение

$$\{(s^0_{(1)}, s^0_{(2)})\} \cup \mu$$

является НБМ$^+$ между $a.P_1$ и $a.P_2$.

(4.95) следует

- из вышедоказанного утверждения, и
- из соотношения $\tau.P \approx P$, которое верно согласно (4.57).

2. Пусть *P* имеет вид

$$P = (S, s^0, R)$$

Обозначим символами $S_{(1)}$ и $S_{(2)}$ дубликаты множества *S* в процессах *P* и $\tau.P$ соответственно, входящих в левую часть соотношения (4.96).

Элементы этих дубликатов мы будем обозначать символами $s_{(1)}$ и $s_{(2)}$ соответственно, где s - произвольный элемент множества S. Пусть символы s_l^0 и s_r^0 обозначают начальные состояния процессов в левой и правой частях соотношения (4.96) соответственно. Тогда отношение

$$\{(s_l^0, s_r^0)\} \cup \{(s_{(i)}, s) \mid s \in S,\ i = 1,2\}$$

является НБМ+ между левой и правой частями соотношения (4.96).

3. Пусть $P_i = (S_i, s_i^0, R_i) \quad (i = 1,2)$. Мы можем считать, что $S_1 \cap S_2 = \emptyset$. Обозначим

- символом s_τ^0 начальное состояние процесса

$$P_1 + \tau.P_2 \qquad (4.99)$$

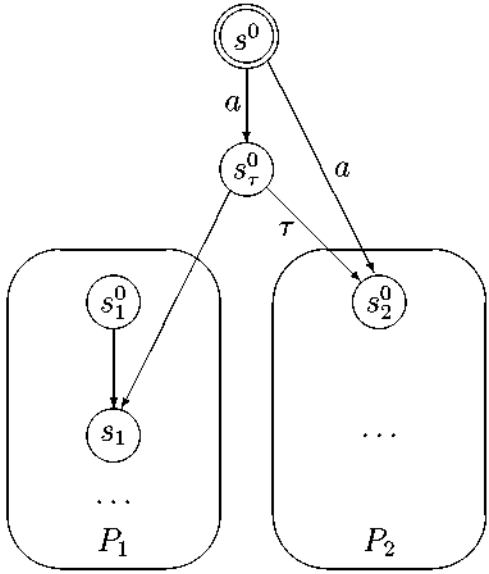
- символом s^0 - начальное состояние процесса

$$a.(P_1 + \tau.P_2) \qquad (4.100)$$

Заметим, что (4.100) совпадает с правой частью (4.97).
Левая часть (4.97) сильно эквивалентна процессу *P′,* который получается из (4.100) добавлением перехода

$$s^0 \xrightarrow{a} s_2^0$$

это легко увидеть при рассмотрении графового представления процесса *P′,* которое имеет вид



Нетрудно доказать, что процесс *P′* наблюдаемо конгруэнтен процессу (4.100). Множества состояний этих процессов можно рассматривать как дубликаты $S_{(1)}$ и $S_{(2)}$ одного и того же множества *S*, и НБМ⁺ между *P′* и (4.100) имеет вид

$$\{(s_{(1)}, s_{(2)}) \mid s \in S\} \quad (4.101)$$

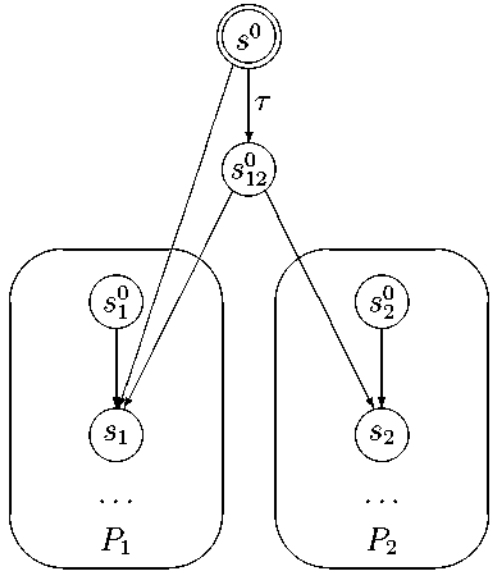
Поскольку

- по теореме 22, имеет место включение $\sim \subseteq \overset{+}{\approx}$, и
- (4.100) совпадает с правой частью (4.97),

то мы доказали наблюдаемую конгруэнтность левой и правой частей соотношения (4.97).

4. Рассуждения в данном случае аналогичны рассуждениям в предыдущем случае. Мы не будем излагать их детально, отметим лишь, что

- левая часть соотношения (4.98) находится в отношении сильной эквивалентности с процессом *P′*, графовое представление которой имеет вид



где

- s^0_1 и s^0_2 - начальные состояния процессов P_1 и P_2,
- s^0_{12} - начальное состояние процесса $P_1 + P_2$

• правая часть соотношения (4.98), которую мы обозначим символом *P″,* получается из *P′* удалением переходов вида

$$s^0 \longrightarrow s_1$$

Нетрудно доказать, что $P' \overset{+}{\approx} P''$. Множества состояний этих процессов можно рассматривать как дубликаты $S_{(1)}$ и $S_{(2)}$ одного и того же множества *S,* и НБМ⁺ между *P′* и *P″* имеет вид (4.101).

4.9.6 Распознавание наблюдаемой конгруэнтности

Для решения задачи распознавания для двух заданных конечных процессов, являются ли они наблюдаемо конгруэнтными, можно использовать следующую теорему.

Теорема 24.

Пусть *P₁ и P₂* - конечные процессы. Соотношение

$$P_1 \overset{+}{\approx} P_2$$

имеет место тогда и только тогда, когда

$$\begin{cases} (s_1^0, s_2^0) \in \mu_\tau(P_1, P_2) \\ \mu_\tau(P_1, P_2) \ - \ \text{НБМ}^+ \end{cases}$$

4.9.7 Минимизация процессов относительно наблюдаемой конгруэнции

Для решения задачи минимизации конечных процессов относительно наблюдаемой конгруэнции можно использовать следующие теоремы.

Теорема 25.

Пусть $P = (S, s^0, R)$ - произвольный процесс.

Обозначим символом P≈ **фактор-процесс** процесса *P* по эквивалентности $\mu_\tau(P, P)$,т.е. процесс, компоненты которого имеют следующий вид.

• Множество состояний процесса P≈ представляет собой совокупность классов эквивалентности множества *S* по отношению $\mu_\tau(P, P)$.

• Начальным состоянием является класс $[s^0]$.

• Переходы процесса P≈ имеют вид

$$[s_1] \xrightarrow{a} [s_2]$$

Где $s_1 \xrightarrow{a} s_2$ - произвольный переход из *R.*

Тогда $P \overset{+}{\approx} (P_\approx)$.

Теорема 26.
Пусть процесс P' получается из процесса P путём удаления недостижимых состояний. Тогда $P'_{\approx}$ имеет наименьшее число состояний среди всех процессов, которые наблюдаемо конгруэнтны P.

5. Рекурсивные определения процессов

В некоторых случаях процесс удобнее задавать не явным описанием множеств его состояний и переходов, а при помощи рекурсивного определения.

5.1 Процессные выражения

Для того, чтобы сформулировать понятие рекурсивного определения процессов, мы введём понятие **процессного выражения.**
Множество *PExpr* **процессных выражений** (ПВ) определяется индуктивно, т.е.
- указываются элементарные ПВ, и
- описываются правила построения новых ПВ из уже имеющихся.

Каждое из правил построения ПВ имеет своё название, которое указывается жирным шрифтом перед описанием этого правила.

процессные константы:
Мы будем предполагать, что задано счётное множество **процессных** констант, причём каждой процессной константе сопоставлен некоторый процесс, называемый **значением** этой константы.
Существует процессная константа, значением которой является пустой процесс **0**, эта константа обозначается тем же символом **0**.
Каждая процессная константа является **ПВ.**

процессные имена:
Мы будем предполагать, что задано счётное множество **процессных имён.**
Каждое процессное имя является **ПВ.**

префиксное действие:
Для каждого $a \in Act$ и каждого ПВ P знакосочетание $a.P$ является **ПВ.**

выбор:

Для любых ПВ Pi, *P2* знакосочетание *P*\ + P_2 является **ПВ.**
параллельная композиция:
Для любых ПВ P_1, P_2 знакосочетание P_1 | P_2 является **ПВ.**

ограничение:
Для каждого подмножества $L \subseteq Names$ и каждого ПВ Р знакосочетание *P**L* является **ПВ.**
переименование:
Для каждого переименования *f* и каждого ПВ Р знакосочетание $P[f]$ является **ПВ.**

5.2 Понятие рекурсивного определения процессов

Рекурсивным определением (РО) процессов называется список формальных равенств вида

$$\begin{cases} A_1 = P_1 \\ \dots \\ A_n = P_n \end{cases} \qquad (5.1)$$

где

- $A_1, \dots, A_n$ - различные процессные имена, и
- $P_1, \dots, P_n$ - ПВ, удовлетворяющие следующему условию: для каждого *i* = 1,..., *n* каждое процессное имя, входящее в P_i, совпадает с одним из имён A_1,..., A_n.

Мы будем предполагать, что каждому процессному имени соответствует единственное РО, в котором это имя является левой частью одного из равенств.
В параграфе 5.5 мы определим соответствие, которое сопоставляет каждому ПВ *P* некоторый процесс $[\![P]\!]$. Для определения этого соответствия мы сначала изложим

- понятие **вложения процессов,** и
- понятие **предела** последовательности вложенных процессов.

а также утверждения, связанные с этими понятиями.

5.3 Вложение процессов

Пусть заданы два процесса

$$P_i = (S_i, s_i^0, R_i) \quad (i = 1, 2)$$

и *f* - инъективное отображение из S_1 в S_2.
Мы будем говорить, что *f* является **вложением** *Pi* в P_2, если

- $f(s_1^0) = s_2^0$, и

• для любых s', $s'' \in S_1$ и любого $a \in Act$

$$(s' \xrightarrow{a} s'') \in R_1 \quad \Leftrightarrow \quad (f(s') \xrightarrow{a} f(s'')) \in R_2$$

Для каждой пары процессов P_1, P_2 знакосочетание

$$P_1 \hookrightarrow P_2$$

является сокращённой записью утверждения о том, что существует вложение P_1 в P_2.

Если процессы P_1 P_2 имеют вид (??), и $P_1 \hookrightarrow P_2$, то мы можем отождествить P_1 с его образом в P_2, т.е. мы можем считать, что

- $S_1 \subseteq S_2$
- $s_1^0 = s_2^0$, и
- $R_1 \subseteq R_2$.

Теорема 27. Пусть $P_1 \hookrightarrow P_2$. Тогда

- $a.P_1 \hookrightarrow a.P_2$
- $P_1 + P \hookrightarrow P_2 + P$
- $P_1 \mid P \hookrightarrow P_2 \mid P$
- $P_1 \setminus L \hookrightarrow P_2 \setminus L$, и
- $P_1[f] \hookrightarrow P_2[f]$.

Ниже мы рассматриваем выражения, построенные из процессов, и символов операций над процессами $(a., +, \mid, \setminus L, [f])$. Понятие такого выражения отличается от понятия ПВ, и мы называем такие выражения **выражениями над процессами.** Для каждого выражения над процессами определён процесс, являющийся значением этого выражения. В нижеследующих рассуждениях мы будем обозначать выражение над процессами и его значение одним и тем же символом.

Теорема 28.

Пусть

• P - выражение над процессами, в которое входят процессы

$$P_1, \ldots, P_n$$

• для каждого $i = 1, \ldots, n \quad P_i \hookrightarrow P'_i$, и

• *P′* - выражение, получаемое из *P* заменой для каждого *i* = 1,...,*n* каждого вхождения процесса P_i на соответствующий процесс P'_i.

Тогда $P \hookrightarrow P'$.

Доказательство.

Данная теорема доказывается индукцией по структуре выражения *P:* мы докажем, что для каждого подвыражения *Q* выражения *P* верно утверждение

$$Q \hookrightarrow Q' \quad \textbf{(5.2)}$$

где *Q′* - подвыражение выражения *P′,* которое соответствует подвыражению *Q* .

базис индукции:

Если $Q = P_i$, то $Q' = P'_i$, и (5.2) верно по предположению.

индуктивный переход:

Из теоремы 27 следует, что для каждого подвыражения *Q* выражения *P* имеет место импликация: если для каждого собственного подвыражения Q_1 выражения *Q* (т.е. $Q_1 \neq Q$)

$$Q_1 \hookrightarrow Q'_1$$

то верно (5.2).

Таким образом, (5.2) верно для каждого подвыражения *Q* выражения *P*. В частности, (5.2) верно для *P*.

5.4 Предел последовательности вложенных процессов

Пусть задана последовательность процессов

$$\{P_k \mid k \geq 0\} \quad (5.3)$$

такая, что

$$\forall k \geq 0 \quad P_k \hookrightarrow P_{k+1} \quad (5.4)$$

Последовательность процессов (5.3), удовлетворяющая условию (5.4), называется **последовательностью** вложенных **процессов.**

Определим процесс

$$\lim_{k \to \infty} P_k,$$

называемый **пределом** последовательности (5.3).

Пусть процессы $P_k \ (k \geq 0)$ имеют вид

$$P_k = (S_k, s_k^0, R_k)$$

Согласно (5.4), мы можем предполагать, что $\forall k \geq 0$

- $S_k \subseteq S_{k+1}$
- $s_k^0 = s_{k+1}^0$
- $R_k \subseteq R_{k+1}$

т.е. компоненты процессов $P_k \ (k \geq 0)$ имеют следующие свойства:

- $S_0 \subseteq S_1 \subseteq S_2 \subseteq \ldots$
- $s_0^0 = s_1^0 = s_2^0 = \ldots$
- $R_0 \subseteq R_1 \subseteq R_2 \subseteq \ldots$

Процесс $\lim\limits_{k\to\infty} P_k$ имеет вид

$$(\bigcup_{k\geq 0} S_k, s_0^0, \bigcup_{k\geq 0} R_k)$$

Нетрудно доказать, что для каждого *k≥0*

$$P_k \hookrightarrow \lim_{k\to\infty} P_k$$

Теорема 29.

Пусть заданы последовательности вложенных процессов

$$\{P_k \mid k \geq 0\} \quad \text{и} \quad \{Q_k \mid k \geq 0\}$$

Тогда

- $\lim\limits_{k\to\infty} (a.P_k) = a.(\lim\limits_{k\to\infty} P_k)$
- $\lim\limits_{k\to\infty} (P_k + Q_k) = (\lim\limits_{k\to\infty} P_k) + (\lim\limits_{k\to\infty} Q_k)$
- $\lim\limits_{k\to\infty} (P_k \mid Q_k) = (\lim\limits_{k\to\infty} P_k) \mid (\lim\limits_{k\to\infty} Q_k)$
- $\lim\limits_{k\to\infty} (P_k \setminus L) = (\lim\limits_{k\to\infty} P_k) \setminus L$
- $\lim\limits_{k\to\infty} (P_k[f]) = (\lim\limits_{k\to\infty} P_k)[f]$

Ниже мы будем использовать следующее обозначение: если
- P - ПВ, в которое входят процессные имена $A_1, ..., A_n$, и
- $P_1,..., P_n$ - некоторые процессы то знакосочетание

$$P(P_1/A_1, \ldots, P_n/A_n)$$

обозначает выражение над процессами (а также его значение), получаемое из P заменой для каждого $i = 1,..., n$ каждого вхождения процессного имени A_i на соответствующий процесс P_i.

Теорема 30.

Пусть заданы
- ПВ P, в которое входят процессные имена $A_1, ..., A_n$, и
- последовательности вложенных процессов

$$\{P_i^{(k)} \mid k \geq 0\} \quad (i = 1, \ldots, n)$$

Тогда

$$P((\lim_{k\to\infty} P_1^{(k)})/A_1, \ldots, (\lim_{k\to\infty} P_n^{(k)})/A_n) =$$
$$= \lim_{k\to\infty} P(P_1^{(k)}/A_1, \ldots, P_n^{(k)}/A_n)$$

Доказательство.

Данная теорема доказывается индукцией по структуре ПВ P, с использованием теоремы 29.

5.5 Процессы, определяемые процессными выражениями

В этом параграфе мы излагаем правило, которое сопоставляет каждому ПВ P процесс $[\![P]\!]$,определяемый этим ПВ.

Процессы, определяемые процессными константами, являются значениями этих констант.

Процессы, определяемые ПВ вида

$$a.P, \quad P_1 + P_2, \quad P_1 \mid P_2, \quad P \setminus L, \quad P[f]$$

являются результатами применения соответствующих операций к процессам определяемым ПВ P, P_1 и P_2, т.е.

$$[\![a.P]\!] \stackrel{\text{def}}{=} a.[\![P]\!]$$
$$[\![P_1 + P_2]\!] \stackrel{\text{def}}{=} [\![P_1]\!] + [\![P_2]\!]$$
$$[\![P_1 \mid P_2]\!] \stackrel{\text{def}}{=} [\![P_1]\!] \mid [\![P_2]\!]$$
$$[\![P \setminus L]\!] \stackrel{\text{def}}{=} [\![P]\!] \setminus L$$
$$[\![P\ [f]\ ]\!] \stackrel{\text{def}}{=} [\![P]\!]\ [f]$$

Опишем теперь правило, сопоставляющее процессы процессным именам.

Пусть задано РО вида (5.1). Определим последовательность списков процессов

$$\{(P_1^{(k)}, \ldots, P_n^{(k)}) \mid k \geq 0\} \quad (5.5)$$

следующим образом:

- $P_1^{(0)} \stackrel{\text{def}}{=} \mathbf{0}, \ \ldots, \ P_n^{(0)} \stackrel{\text{def}}{=} \mathbf{0}$

• если процессы $P_1^{(k)}, \ldots, P_n^{(k)}$ уже определены, то для каждого $i = 1,\ldots, n$

$$P_i^{(k+1)} \stackrel{\text{def}}{=} P_i(P_1^{(k)}/A_1, \ldots, P_n^{(k)}/A_n)$$

Докажем, что для каждого $k \geq 0$ и для каждого $i = 1,\ldots, n$

$$P_i^{(k)} \hookrightarrow P_i^{(k+1)} \quad (5.6)$$

Доказательство будем вести индукцией по k.

базис индукции:

Если $k = 0$, то $P_i^{(0)}$ по определению совпадает с процессом **0**, который можно вложить в любой процесс.

индуктивный переход:

Пусть для каждого $i = 1, \ldots, n \quad P_i^{(k-1)} \hookrightarrow P_i^{(k)}$.

По определению процессов из совокупности (5.5), имеют место соотношения

$$P_i^{(k)} = P_i(P_1^{(k-1)}/A_1, \ldots, P_n^{(k-1)}/A_n)$$
$$P_i^{(k+1)} = P_i(P_1^{(k)}/A_1, \ldots, P_n^{(k)}/A_n)$$

Соотношение $P_i^{(k)} \hookrightarrow P_i^{(k+1)}$ следует из теоремы 28.

Определим для каждого $i = 1,\ldots, n$ процесс $[\![A_i]\!]$ как предел

$$[\![A_i]\!] \stackrel{\text{def}}{=} \lim_{k\to\infty} P_i^{(k)}$$

Из теоремы 30 следует, что для каждого i = 1,... ,n верна цепочка равенств

$$\begin{aligned} & P_i([\![A_1]\!]/A_1,\ldots,[\![A_n]\!]/A_n) = \\ & = P_i((\lim_{k\to\infty} P_1^{(k)})/A_1,\ldots,(\lim_{k\to\infty} P_n^{(k)})/A_n) = \\ & = \lim_{k\to\infty} P_i(P_1^{(k)}/A_1,\ldots,P_n^{(k)}/A_n) = \\ & = \lim_{k\to\infty} (P_i^{(k+1)}) = [\![A_i]\!] \end{aligned}$$

т.е. список процессов

$$[\![A_1]\!],\ldots,[\![A_n]\!]$$

является решением системы уравнений, соответствующей РО (5.1) (переменными в этой системе уравнений являются процессные имена).

5.6 Эквивалентность РО

Пусть заданы два РО вида

$$\left\{\begin{array}{l} A_1^{(1)} = P_1^{(1)} \\ \ldots \\ A_n^{(1)} = P_n^{(1)} \end{array}\right. \quad \text{и} \quad \left\{\begin{array}{l} A_1^{(2)} = P_1^{(2)} \\ \ldots \\ A_n^{(2)} = P_n^{(2)} \end{array}\right. \qquad (5.7)$$

Для каждого списка процессов $Q_1, \ldots, Q_n$ мы будем обозначать выражение над процессами (и его значение)

$$P_i^{(j)}(Q_1/A_1^{(j)},\ldots,Q_n/A_n^{(j)}) \qquad (i=1,\ldots,n;\ j=1,2)$$

сокращённо в виде знакосочетания

$$P_i^{(j)}(Q_1,\ldots,Q_n)$$

Пусть задана некоторая эквивалентность μ, на множестве всех процессов.

Мы будем говорить, что РО (5.7) являются **эквивалентными** относительно μ, если для

- каждого списка процессов Q_1, ..., Q_n, и
- каждого i = 1,..., n имеет место соотношение

$$\left(P_i^{(1)}(Q_1,\ldots,Q_n)\ ,\ P_i^{(2)}(Q_1,\ldots,Q_n) \right) \in \mu$$

Теорема 31.
Пусть заданы

• два РО вида (5.7), и
• конгруэнция μ на множестве процессов.
Если РО (5.7) эквивалентны относительно μ, то процессы, определяемые этими РО, т.е.

$$\{[\![A_i^{(1)}]\!] \mid i = 1, \ldots, n\} \quad \text{и} \quad \{[\![A_i^{(2)}]\!] \mid i = 1, \ldots, n\}$$

тоже эквивалентны относительно μ, т.е. для каждого $i = 1,..., n$ имеет место соотношение

$$\left([\![A_i^{(1)}]\!], [\![A_i^{(2)}]\!] \right) \in \mu$$

5.7 Переходы на *PExpr*

Существует другой способ определения соответствия между ПВ и процессами. Данный способ связан с определением **множества переходов** $\mathcal{R}$ на совокупности *PExpr* всех ПВ. Каждый переход из $\mathcal{R}$ представляет собой тройку

$$(P, a, P') \quad \textbf{(5.8)}$$

где $P, P' \in PExpr$, и $a \in Act$.
Если $(5.8) \in \mathcal{R}$, то мы сокращённо обозначаем этот факт в виде знакосочетания

$$P \xrightarrow{a} P' \quad (5.9)$$

Понятие перехода определяется индуктивно, т.е.
• указываются тройки вида (5.8), которые являются переходами по определению, и
• описываются правила построения новых переходов из уже имеющихся.
В этом параграфе мы предполагаем, что
• значением каждой процессной константы является конечный процесс, и
• каждый конечный процесс является значением некоторой процессной константы.
В нижеследующих правилах, определяющих множество переходов $\mathcal{R}$, символы *P, P′* обозначают произвольные ПВ, и символ *a* обозначает произвольное действие из *Act.*
1. если *P* - процессная константа, то

$$P \xrightarrow{a} P'$$

где *P′* - процессная константа, такая, что
• значения *P* и *P′* имеют вид

(S, s^0, R) и (S, s^1, R)

соответственно, и

- *R* содержит переход $s^0 \xrightarrow{a} s^1$

2. $a.P \xrightarrow{a} P$

3. если $P \xrightarrow{a} P'$,то

- $P + Q \xrightarrow{a} P'$, и
- $Q + P \xrightarrow{a} P'$
- $P \mid Q \xrightarrow{a} P' \mid Q$, и
- $Q \mid P \xrightarrow{a} Q \mid P'$

- если $L \subseteq Names$, $a \neq \tau$, и $name(a) \notin L$, то

$P \setminus L \xrightarrow{a} P' \setminus L$

- для каждого переименования *f*

$P[f] \xrightarrow{f(a)} P'[f]$

4. если $a \neq \tau$, то из

$P_1 \xrightarrow{a} P_1'$ и $P_2 \xrightarrow{\bar{a}} P_2'$

следует, что

$P_1 \mid P_2 \xrightarrow{\tau} P_1' \mid P_2'$

5. для каждого РО (5.1) и каждого $i \in \{1, \ldots, n\}$

если $P_i \xrightarrow{a} P'$

то $A_i \xrightarrow{a} P'$ (5.10)

Можно доказать, что для каждого ПВ *P* существует лишь конечное множество переходов с началом *P,* т.е. имеющих вид

$P \xrightarrow{a} P'$

Для каждого ПВ $P \in PExpr$ процесс $[\![P]\!]$,соответствующий этому ПВ, имеет вид

$(PExpr, P, \mathcal{R})$

При данном определении соответствия между ПВ и процессами имеет место следующая теорема.

Теорема 32.

Для каждого РО (5.1) и каждого *i* = 1,..., *n*

$[\![A_i]\!] \sim P_i([\![A_1]\!]/A_1, \ldots, [\![A_n]\!]/A_n)$

(т.е. список процессов $[\![A_1]\!], \ldots, [\![A_n]\!]$ является решением системы уравнений, соответствующей РО (5.1) **с точностью до** ~).

5.8 Доказательство эквивалентности процессов при помощи РО

Можно доказывать эквивалентность $(\sim$ **или** $\overset{+}{\approx})$ двух процессов путём предъявления РО, такого, что оба этих процесса являются компонентами с одинаковыми номерами некоторых решений системы уравнений, соответствующей этому РО.
Соответствующие эквивалентности обосновываются теоремой 33.
Для формулировки этой теоремы мы введём следующее вспомогательное понятие.
Пусть заданы
- бинарное отношение μ , на множестве всех процессов, и
- РО вида (5.1).

Мы будем говорить, что список процессов, определяемый РО (5.1), единствен с точностью до μ, если для каждой пары списков процессов

$$(Q_1^{(1)}, \ldots, Q_n^{(1)}) \quad \text{и} \quad (Q_1^{(2)}, \ldots, Q_n^{(2)})$$

удовлетворяющей следующему условию: для каждого $i = 1,..., n$

$$\left([\![Q_i^{(1)}]\!] \,,\; P_i(Q_1^{(1)}/A_1, \ldots, Q_n^{(1)}/A_n) \right) \in \mu$$
$$\left([\![Q_i^{(2)}]\!] \,,\; P_i(Q_1^{(2)}/A_1, \ldots, Q_n^{(2)}/A_n) \right) \in \mu$$

имеет место соотношение

$$\forall i = 1, \ldots, n \quad \left([\![Q_i^{(1)}]\!] \,,\; [\![Q_i^{(2)}]\!] \right) \in \mu$$

Теорема 33.
Пусть задано РО вида (5.1).
1. Если каждое вхождение каждого процессного имени A_i в каждое ПВ P_j содержится в подвыражении вида $a.Q$, то список процессов, определяемый РО (5.1), единствен с точностью до ~.
2. Если
- каждое вхождение каждого A_i в каждое P_j содержится в подвыражении вида $a.Q$, где $a \neq \tau$, и
- каждое вхождение каждого A_i в каждое P_j содержится только в подвыражениях вида $a.Q$ и $Q_1 + Q_2$

то список процессов, определяемый РО (5.1), единствен с точностью до $\overset{+}{\approx}$.

5.9 Проблемы, связанные с понятием РО

1. Распознавание существования конечных процессов, эквивалентных (относительно $\sim$, $\approx$, $\overset{+}{\approx}$) процессам вида $[\![A]\!]$.
2. Построение алгоритмов нахождения минимальных процессов, эквивалентных процессам вида $[\![A]\!]$ в том случае, когда эти процессы конечны.
3. Распознавание эквивалентности процессов вида *{A}*
(эти процессы могут быть бесконечными, и методы из главы 4 для них не подходят).
4. Распознавание эквивалентности РО.
5. Нахождение необходимых и достаточных условий единственности списка процессов, определяемого РО (с точностью до $\sim$, $\overset{+}{\approx}$).

6. Примеры доказательства свойств процессов

6.1 Потоковые графы

Если процесс *P* можно представить в виде алгебраического выражения

$$P(P_1, \ldots, P_n) \quad (6.1)$$

в которое входят процессы $P_1,...,P_n$, соединённые символами операций
- параллельной композиции,
- ограничения, и
- переименования

то *P* называется **структурной композицией** процессов $P_1,...,P_n$.
Если процесс *P* является структурной композицией, то ему можно сопоставить некоторый графический объект

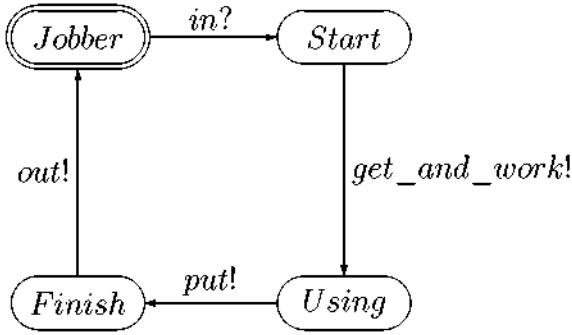
$$G(P)$$

называемый **потоковым графом (ПГ)** процесса *P*.
Представление структурной композиции в виде ПГ повышает наглядность и облегчает понимание взаимосвязи между её компонентами.
Для построения ПГ *G(P)* для процесса *P* вида (6.1) строятся ПГ, соответствующие всем подвыражениям выражения (6.1).
Элементарные ПГ:
Для каждого $i = 1,..., n$, и каждого вхождения P_i в выражение (6.1), ПГ $G(P_i)$, соответствующий этому вхождению, имеет вид овала, внутри которого написано знакосочетание P_i.

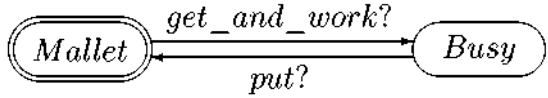
На периметре овала рисуется несколько кружочков, называемых **портами.**
Каждый порт соответствует некоторому действию из множества *Act(Pi),* причём
• если это действие имеет вид $\alpha!$, то соответствующий этому действию порт рисуется чёрным цветом, и
• если это действие имеет вид $\alpha?$, то соответствующий этому действию порт рисуется белым цветом.
Около каждого порта написана его метка, равная тому действию из $Act(P_i)$, которому соответствует этот порт.
Отметим, что если P_i имеет несколько вхождений в выражение (6.1), то для каждого такого вхождения рисуется отдельный элементарный **ПГ** *G(Pi).*
Параллельная композиция:
Если выражение (6.1) содержит подвыражение вида *P' | P",* то *G(P' | P")* получается
• дизъюнктным объединением *G(P')* и *G(P"),* и
• соединением стрелочками в этом дизъюнктном объединении портов ПГ *G(P')* и *G(P")* с комплементарными метками: если
— один из этих ПГ содержит порт с меткой $\alpha!$, и
— другой из этих ПГ содержит порт с меткой $\alpha?$,
то рисуется стрелочка с меткой α от первого порта к второму.
Ограничение:
$G(P'\backslash L)$ получается из *G(P')* удалением меток портов, имена которых принадлежат *L.*
Переименование:
$G(P'[f])$ получается из *G(P')* соответствующим переименованием меток портов.
В излагаемых ниже примерах процессов приводятся ПГ, соответствующие этим процессам.

6.2 Мастерская

Рассмотрим модель мастерской, в которой работают двое рабочих, пользующиеся для работы одним молотком.
Поведение каждого рабочего в мастерской описывается процессом *Jobber*



• Действия $in?$ **и** $out!$ используются для взаимодействия рабочего с заказчиком, и обозначают соответственно
— прием материала, и
— выдачу готового изделия.
• Действия *get_and_work* и *putl* используются для взаимодействия рабочего с молотком, и обозначают соответственно — взятие молотка и выполнение с его помощью некоторых действий, и
— возвращение молотка на место.
Обратим внимание, что действие *get_and_work* состоит из нескольких действий, которые мы не детализируем и агрегируем все их в одно действие.
Согласно графовому представлению процесса *Jobber,* рабочий
• сначала принимает материал,
• затем берет молоток и работает,
• после чего кладет молоток,
• выдает готовое изделие,
• и все повторяется сначала.
Поведение молотка мы представляем при помощи следующего процесса *Mallet:*

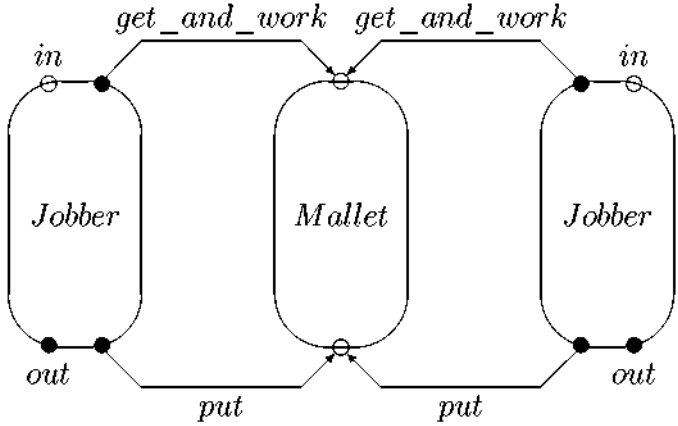


Отметим, что объект "молоток" и процесс "молоток" - это разные понятия.
Функционирование мастерской определяется при помощи следующего процесса *Job_S'hop:*

$$Job\_Shop = (Jobber \mid Jobber \mid Mallet) \setminus L$$

где $L = \{get\_and\_work,\ put\}$.

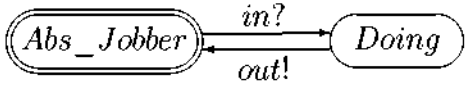
Потоковый граф процесса *Job_Shop* имеет следующий вид.



Введем теперь понятие "абстрактного рабочего", про которого известно, что он циклически

- принимает материал, и
- выдает готовые изделия

но ничего неизвестно о подробностях процесса его работы. Поведение "абстрактного рабочего" мы зададим при помощи следующего процесса *Abs_Jobber:*



Поведение "абстрактной мастерской" мы зададим при помощи следующего процесса *Abs_Job_Shop:*

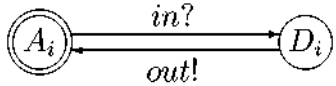
$$Abs\_Job\_Shop = Abs\_Jobber \mid Abs\_Jobber$$

"Абстрактную мастерскую" мы будем использовать как **спецификацию** мастерской. Процесс "абстрактная мастерская" представляет поведение мастерской без учета деталей ее реализации. Докажем соответствие мастерской ее спецификации, то есть наличие наблюдаемой конгруэнции

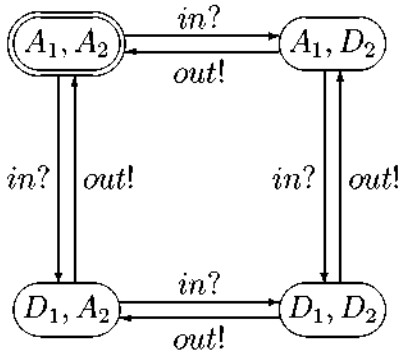
$$Job\_Shop \overset{+}{\approx} Abs\_Job\_Shop \quad \textbf{(6.2)}$$

Процесс *Abs_Job_Shop* является параллельной композицией двух процессов *Abs_ Jobber*. В целях предотвращения коллизии в обозначениях, мы выберем различные идентификаторы для обо-

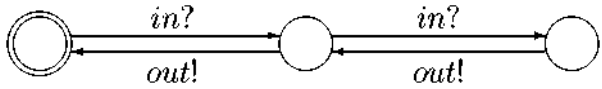
значения состояний этих процессов. Пусть, например, эти процессы имеют вид



где i = 1, 2.
Параллельная композиция этих процессов имеет вид



Применяя к данному процессу процедуру минимизации относительно наблюдаемой эквивалентности, мы получим процесс



(6.3)

Процесс *Job_Shop* имеет 4-4-2 = 32 состояния, и мы не приводим его здесь ввиду его большой громоздкости. Если минимизировать этот процесс относительно наблюдаемой эквивалентности, то получится процесс, изоморфный процессу (6.3).
Это означает, что имеет место соотношение

$$Job\_Shop \approx Abs\_Job\_Shop \quad (6.4)$$

Поскольку из начальных состояний процессов

$$Job\_Shop \text{ и } Abs\_Job\_Shop$$

не выходит рёбер с меткой τ, то отсюда и из (6.4) следует искомое соотношение (6.2).

6.3 Неконфликтное использование ресурса

Предположим, что имеется некоторая фирма, сотрудники которой объединены в несколько групп.

В здании, где работает фирма, выделена одна комната, которую каждая из групп может использовать для проведения своих рабочих совещаний.

Предположим, что необходимо обеспечить неконфликтное использование этой комнаты группами. Это означает, что когда одна из групп проводит в комнате совещание, другой группе должно быть запрещено проводить своё совещание в этой комнате.

Для решения этой задачи создаётся специальный процесс -**диспетчер.** Если какая-либо из групп хочет провести совещание в этой комнате, она должна послать диспетчеру заявку на предоставление ей права пользования комнатой для проведения этого совещания.

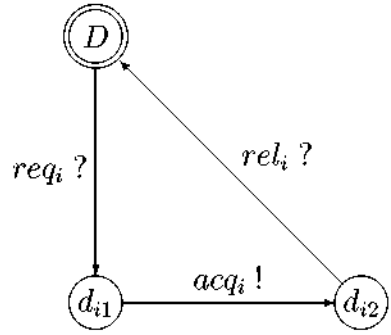
Когда диспетчер разрешает какой-либо группе использовать комнату, он посылает ей уведомление об этом.

Окончив совещание, группа должна сообщить об этом диспетчеру, чтобы диспетчер знал, что комната стала свободной и доступна для других групп.

Рассмотрим описание работы указанной системы при помощи теории процессов.

Пусть число групп равно n ($n \geq 2$).

Диспетчер мы опишем как процесс с именем *D,* графовое представление которого содержит для каждого $i = 1,..., n$ подграф



т.е.

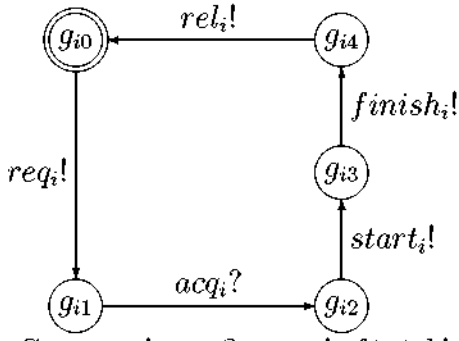
$$D \sim \sum_{i=1}^{n} req_i?.\ acq_i!.\ rel_i?.\ D$$

Действия, входящие в *Act(D),* имеют следующий смысл:

- $req_i?$ - получение заявки от *i-й* группы

• $acq_i!$ - уведомление *i-й* группы о том, что она имеет право пользоваться комнатой
• $rel_i?$ - получение сообщения от *i-й* группы об освобождении ею комнаты.

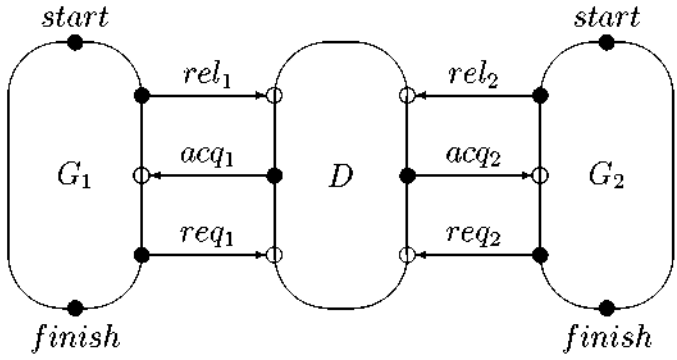
Опишем теперь поведение каждой группы.
(Мы будем описывать только взаимодействие групп с диспетчером и с комнатой, и не будем касаться прочих их функций).
Мы будем представлять
• начало проведения совещания в комнате действием *start!*,
и
• окончание совещания - действием *finish!*.
Поведение *i-й* группы мы опишем в виде процесса G_i, который имеет следующее графовое представление:



т.е. $G_i \sim req_i!.\ acq_i?.\ start!.\ finish!.\ rel_i!.\ G_i.$
Совместное поведение диспетчера и групп можно описать следующим процессом:

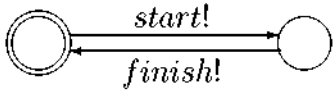
$$Sys = (D \mid G_1 \mid \ldots \mid G_n) \setminus L$$

где $L = \{req_i, acq_i, rel_i \mid i = 1, \ldots, n\}.$
Потоковый граф системы, состоящей из диспетчера и групп для $n = 2$ показан на рисунке



Покажем теперь, что алгоритмы работы диспетчера и групп действительно обеспечивают неконфликтный режим использования комнаты, который заключается в том, что после начала проведения совещания в комнате какой-либо группой (то есть после выполнения этой группой действия *startl)* никакая другая группа не может начать проводить в этой комнате своё совещание (т.е. тоже выполнить действие *startl),* до тех пор первая группа не закончит своё совещание (т.е. пока не будет выполнено действие *finishl).*

Определим процесс *Spec* следующим образом:



т.е. $Spec \sim start!.\ finish!.\ Spec.$

Наличие неконфликтного режима использования комнаты эквивалентно истинности соотношения

$$Sys \approx Spec \quad \textbf{(6.5)}$$

Это соотношение можно рассматривать как требование к системе, состоящей из диспетчера и групп.

Докажем соотношение (6.5).

Преобразуем процесс *Sys,* применив несколько раз теорему о разложении:

$$Sys \sim$$

$$\sim \sum_{i=1}^{n} \tau. \left(\begin{array}{l} acq_i!.\ rel_i?.\ D \,|\, G_1 \,|\, \ldots \\ \ldots \,|\, acq_i?.\ start!.\ finish!.\ rel_i!.\ G_i \,|\, \ldots \\ \ldots \,|\, G_n \end{array} \right) \setminus L \sim$$

$$\sim \sum_{i=1}^{n} \tau.\tau. \left(\begin{array}{l} rel_i?.\ D \,|\, G_1 \,|\, \ldots \\ \ldots \,|\, start!.\ finish!.\ rel_i!.\ G_i \,|\, \ldots \\ \ldots \,|\, G_n \end{array} \right) \setminus L \sim$$

$$\sim \sum_{i=1}^{n} \tau.\tau.start!. \left(\begin{array}{l} rel_i?.\ D \,|\, G_1 \,|\, \ldots \\ \ldots \,|\, finish!.\ rel_i!.\ G_i \,|\, \ldots \\ \ldots \,|\, G_n \end{array} \right) \setminus L \sim$$

$$\sim \sum_{i=1}^{n} \tau.\tau.start!.\ finish!. \left(\begin{array}{l} rel_i?.\ D \,|\, G_1 \,|\, \ldots \\ \ldots \,|\, rel_i!.\ G_i \,|\, \ldots \\ \ldots \,|\, G_n \end{array} \right) \setminus L \sim$$

$$\sim \sum_{i=1}^{n} \tau.\tau.start!.\ finish!.\ \tau. \underbrace{\left(\begin{array}{l} D \,|\, G_1 \,|\, \ldots \\ \ldots \,|\, G_i \,|\, \ldots \\ \ldots \,|\, G_n \end{array} \right) \setminus L}_{Sys} =$$

$$= \sum_{i=1}^{n} \tau.\tau.start!.\ finish!.\ \tau.Sys$$

Воспользовавшись тождествами

$$P + P \sim P \quad \text{и} \quad \alpha.\tau.P \overset{+}{\approx} \alpha.P$$

получаем отсюда, что

$$Sys \overset{+}{\approx} \tau.start!.\ finish!.\ Sys$$

Рассмотрим теперь уравнение

$$X = \tau.start!.\ finish!.\ X \qquad (6.6)$$

Согласно теореме 33 из параграфа 5.8, решение этого уравнения с точностью до $\overset{+}{\approx}$ единственно.

Как было показано выше, процесс *Sys* является решением уравнения (6.6) с точностью до $\overset{+}{\approx}$.

Процесс τ.*Spec* тоже является решением уравнения (6.6) с точностью до $\overset{+}{\approx}$, так как

$$\tau.Spec \sim \tau.start!.\ finish!.\ Spec \overset{+}{\approx}$$

$$\overset{+}{\approx} \tau.start!.\ finish!.\ (\tau.Spec)$$

Следовательно, имеет место соотношение

$$Sys \overset{+}{\approx} \tau.Spec$$

из которого следует (6.5).

6.4 Планировщик

Предположим, что имеется *n* процессов

$$P_1, \ldots, P_n \quad (6.7)$$

и для каждого *i* = 1,..., *n* среди действий, которые может выполнить P_i, есть два служебных действия:

• действие $\alpha_i?$, которое представляет собой сигнал

$$P_i \ начинает \ работу \quad (6.8)$$

• действие $\beta_i?$, которое представляет собой сигнал

$$P_i \ заканчивает \ работу \quad (6.9)$$

Мы предполагаем, что все имена

$$\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n \quad (6.10)$$

различны, и для каждого *i* = 1,..., *n* имена из

$$names(Act(P_i)) \setminus \{\alpha_i, \beta_i\}$$

не совпадают ни с одним именем из (6.10). Обозначим множество имён из (6.10) символом *L*.

Для каждого *i* = 1,..., *n* действия из множества

$$Act(P_i) \setminus \{\alpha_i?, \beta_i?\}$$

называются **собственными действиями** процесса P_i.

Произвольная трасса каждого процесса P_i может содержать действия $\alpha_i?$ **и** $\beta_i?$ в любом количестве и в любом порядке.

Мы хотели бы создать новый процесс *P,* в котором все процессы P_1, ..., P_n работали бы совместно, причём эта совместная работа должна подчиняться определённому режиму. Процесс *P* должен иметь вид

$$P = (P_1 \mid \ldots \mid P_n \mid Sch) \setminus L$$

где процесс *Sch* называется **планировщиком** и предназначен для задания требуемого режима работы процессов P_1 ..., P_n. Процесс *Sch* должен выполнять только действия из множества

$$\{\alpha_1!, \ldots, \alpha_n!, \beta_1!, \ldots, \beta_n!\} \quad (6.11)$$

В силу определения процесса *P,* для каждого *i* = 1,..., *n*

• каждое исполнение процессомдействия $P_i \in (6.7)$ или $\alpha_i?$

$\beta_i?$ в составе процесса *P* может произойти только одновременно с исполнением комплементарного действия процессом *Sch,* и
• исполнение этих действий будет невидимо за пределами процесса P.
Говоря неформально, каждый процесс P_i, работающий в составе процесса P, может начать или закончить какой-либо сеанс своей работы тогда и только тогда, когда планировщик *Sch* разрешит ему это сделать.
Режим, которому должна подчиняться работа процессов $P_1, ..., P_n$, заключается в следующих двух условиях.
1. Для каждого $i = 1,..., n$ произвольная трасса процесса P_i, работающего в составе процесса *P,* должна иметь вид

$$\alpha_i? \ \ldots \ \beta_i? \ \ldots \alpha_i? \ \ldots \ \beta_i? \ \ldots$$

где точки изображают собственные действия процесса P_i, т.е. функционирование процесса P_i должно представлять собой последовательность сеансов вида

$$\alpha_i? \ \ldots \ \beta_i? \ \ldots$$

где каждый сеанс
• начинается с сигнала $\alpha_i?$ о начале сеанса
• далее выполняются некоторые собственные действия процесса P_i
• затем производится сигнал $\beta_i?$ о завершении сеанса,
• после чего процесс P_i может производить некоторые собственные действия (например, связанные с подготовкой к следующему сеансу).
2. Процессы P_i должны начинать новые сеансы только по очереди в циклическом порядке, т.е.
• сначала может начать свой первый сеанс только процесс P_1
• потом может начать свой первый сеанс процесс P_2
• ...
• затем может начать свой первый сеанс процесс P_n
• после этого может начать свой второй сеанс процесс P_1
• затем может начать свой второй сеанс процесс P_2
• и т.д.
Отметим, что мы не требуем, чтобы каждый процесс P_i получал разрешение начать свой *k-й* сеанс только после того, как предыдущий процесс P_{i-1} завершит свой *k-й* сеанс. Однако мы требуем, чтобы каждый процесс P_i получал разрешение начать новый сеанс, только если он выполнил действие $\beta_i?$, сигнализирующее о завершении своего предыдущего сеанса.
Исполнение собственных действий процессами P_i может производиться в произвольном порядке, в том числе допускается и

взаимодействие между этими процессами во время их работы в составе процесса *P*.

Описанный режим можно равносильным образом выразить в виде следующих двух условий на произвольную трассу

$$tr \in Tr(Sch)$$

В формулировке этих условий мы будем использовать следующее обозначение: если

$$tr \in Tr(Sch) \quad \text{и} \quad M \subseteq Act$$

то знакосочетание $tr \mid_M$ обозначает последовательность действий, получаемую из *tr* удалением всех действий, не принадлежащих *M*.

Условия, выражающие описанный выше режим, имеют следующий вид:

$$\begin{array}{l} \forall\, tr \in Tr(Sch),\ \forall\, i = 1, \ldots, n \\ tr \mid_{\{\alpha_i, \beta_i\}} \ = \ (\alpha_i!\ \beta_i!\ \alpha_i!\ \beta_i!\ \alpha_i!\ \beta_i!\ \ldots) \end{array} \qquad (6.12)$$

и

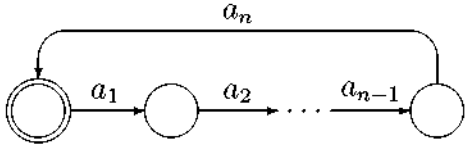
$$\begin{array}{l} \forall\, tr \in Tr(Sch) \\ tr \mid_{\{\alpha_1, \ldots, \alpha_n\}} \ = \ (\alpha_1!\ \ldots\ \alpha_n!\ \alpha_1!\ \ldots\ \alpha_n!\ \ldots) \end{array} \qquad (6.13)$$

Данные условия можно выразить равносильным образом в виде условия наличия наблюдаемой эквивалентности между некоторыми процессами. Чтобы определить эти процессы, мы введём вспомогательные обозначения.

1. Пусть $a_1 \ldots a_n$ - некоторая последовательность действий из *Act.* Тогда знакосочетание

$$(a_1 \ldots a_n)^*$$

обозначает процесс, графовое представление которого имеет следующий вид:



2. Пусть заданы

- некоторый процесс *P,* и
- некоторое множество действий

$$\{a_1, \ldots, a_k\} \subseteq Act \setminus \{\tau\} \qquad (6.14)$$

Знакосочетание

$$hide\ (P, a_1, \ldots, a_k) \quad (6.15)$$

обозначает процесс

$$(\ P \mid (\overline{a_1})^* \mid \ldots \mid (\overline{a_k})^*\ ) \setminus names(\{a_1, \ldots, a_k\})$$

Процесс (6.15) можно рассматривать как процесс, получаемый из *P* заменой меток переходов: все метки переходов в *P,* принадлежащие множеству (6.14), заменяются на τ.

Используя введённые обозначения, условие (6.12) можно выразить следующим образом: для каждого *i* = 1,... *n*

$$hide\left(\begin{array}{ll} Sch, & \alpha_1!, \ldots, \alpha_{i-1}!, \alpha_{i+1}!, \ldots, \alpha_n! \\ & \beta_1!, \ldots, \beta_{i-1}!, \beta_{i+1}!, \ldots, \beta_n! \end{array}\right) \approx \quad (6.16)$$
$$\approx (\alpha_i!.\ \beta_i!)^*$$

Условие (6.13) можно выразить следующим образом:

$$hide\ (Sch, \beta_1!, \ldots, \beta_n!) \approx (\alpha_1!.\ \ldots \alpha_n!)^* \quad (6.17)$$

Нетрудно заметить, что существует несколько планировщиков, удовлетворяющих данным условиям. Например, данным условиям удовлетворяют следующие планировщики:

- $Sch\ =\ (\alpha_1!\ \beta_1!\ \ldots\ \alpha_n!\ \beta_n!)^*$
- $Sch\ =\ (\alpha_1!\ \ldots\ \alpha_n!\ \beta_1!\ \ldots\ \beta_n!)^*$

Однако такие планировщики налагают слишком жёсткие ограничения на работу процессов P_1,..., P_n.

Нам хотелось бы, чтобы планировщик допускал максимальную свободу в поведении процессов P_1,..., P_n в составе процесса *P.* Это означает, что если в какой-либо момент времени

- какой-либо процесс P_i имеет намерение выполнить действие $a \in \{\alpha_i?, \beta_i?\}$, и
- это намерение процесса P_i не противоречит описанному выше режиму

то планировщик не должен отказывать процессу P_i в возможности выполнить это действие в текущий момент времени, т.е. среди действий планировщика, которые он может выполнить в текущий момент времени, должно присутствовать действие $\overline{a}$ (не факт, что именно это действие будет выполнено в текущий момент времени, но среди возможных действий оно должно присутствовать) .

Сформулированное выше неформальное описание максимальной свободы в поведении планировщика можно формально уточнить следующим образом:

• каждому состоянию *s* планировщика можно сопоставить метку *label(s),* имеющую вид пары *(i,X)* где

– $i \in \{1,\ldots,n\}$, $i =$ номер того процесса, который имеет право начать очередной сеанс в текущий момент времени

– $X \subseteq \{1,\ldots,n\} \setminus \{i\}$, $X =$ список активных процессов на текущий момент времени (т.е. таких процессов, которые начали очередной сеанс, но пока его не закончили)

• начальное состояние планировщика имеет метку $(1, \varnothing)$

• множество переходов состоит из

— переходов вида

$$s \xrightarrow{\alpha_i!} s'$$

где

$$* \ label(s) = (i, X)$$

$$* \ label(s') = (next(i), X \cup \{i\}), \text{ где}$$

$$next(i) \stackrel{\text{def}}{=} \begin{cases} i+1, & \text{если } i < n, \text{ и} \\ 1, & \text{если } i = n \end{cases}$$

— и переходов вида

$$s \xrightarrow{\beta_j!} s'$$

где

$$* \ label(s) = (i, X),$$

$$* \ label(s') = (i, X \setminus \{j\}), \text{ причём } j \in X$$

Сформулированное описание свойств требуемого планировщика можно рассматривать как его определение:

• в качестве множества состояний планировщика мы можем взять просто множество пар вида

$$\{(i, X) \in \{1,\ldots,n\} \times \mathcal{P}(\{1,\ldots,n\}) \mid i \notin X\}$$

(т.е. каждое состояние совпадает со своей меткой)

• и определить множество переходов так, как описано выше.

Обозначим такой планировщик знакосочетанием Sch_0.

Описание планировщика Sch_0 содержит существенный недостаток: размер такого описания экспоненциально зависит от числа процессов (6.7), которыми нужно управлять (число состояний Sch_0 равно $n \cdot 2^{n-1}$),

что не позволяет быстро модифицировать такой планировщик в том случае, когда множество процессов (6.7) изменяется (например, к нему добавляются новые процессы, или удаляются старые).
Мы можем использовать Sch_0 только как эталон, с которым мы каким-либо способом будем сравнивать другие планировщики.
Для решения поставленной задачи мы определим другой планировщик *Sch*. Мы будем определять его

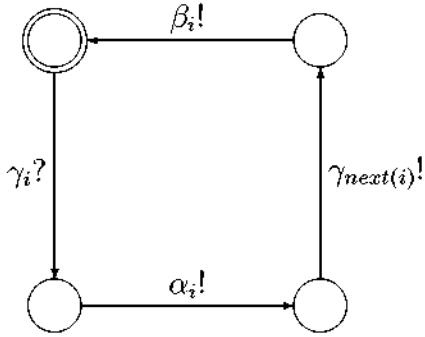
- не путём явного описания состояний и переходов,
- а путём задания некоторого выражения, которое определяет его в терминах композиции процессов достаточно простого вида.

Данное описание будет лишено сформулированного выше недостатка.
В описании планировщика *Sch* мы будем использовать новые вспомогательные имена $\gamma_1, \ldots, \gamma_n$. Обозначим множество этих имён символом Г.
Процесс *Sch* определяется следующим образом:

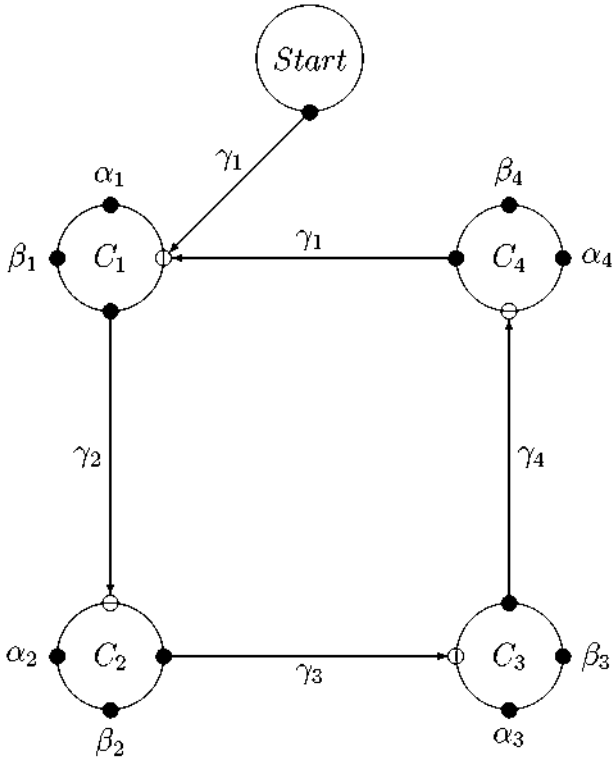
$$Sch \stackrel{\text{def}}{=} (Start \mid C_1 \mid \ldots \mid C_n) \setminus \Gamma \quad (6.18)$$

где

- $Start \stackrel{\text{def}}{=} \gamma_1!.\, \mathbf{0}$
- для каждого $i = 1,..., n$ процесс C_i имеет вид



Потоковый граф процесса *Sch* в случае $n = 4$ имеет следующий вид:



Приведём неформальное пояснение функционирования процесса *Sch.* Назовём процессы $C_1,...,\ C_n$, участвующие в определении планировщика *Sch,* **циклерами.** Циклер C_i называется

- **выключенным,** если он находится в своём начальном состоянии, и
- **включённым,** если он находится не в начальном состоянии.

Процесс *Start* включает первый циклер C_i и после этого "умирает". Каждый циклер C_i отвечает за работу процесса P_i. Циклер C_i

- включает следующий циклер $C_{next(i)}$ после того, как он дал разрешение процессу P_i начать очередной сеанс работы, и
- выключается после того, как он дал разрешение процессу P_i закончить очередной сеанс работы.

Докажем, что процесс (6.18) удовлетворяет условию (6.17) (проверку условия (6.16) мы опустим).

Согласно определению процесса (6.15), условие (6.17) имеет вид

$$(Sch \mid (\beta_1?)^* \mid \ldots \mid (\beta_n?)^*) \setminus B \approx (\alpha_1! \ldots \alpha_n!)^* \quad (6.19)$$

где $B = \{\beta_1, \ldots, \beta_n\}$.

Обозначим $Sch' \stackrel{\text{def}}{=}$ левая часть (6.19).

Докажем, что

$$Sch' \stackrel{+}{\approx} \tau.\alpha_1!. \ \ldots \alpha_n!. \ Sch' \quad (6.20)$$

Отсюда, по свойству единственности (с точностью до $\stackrel{+}{\approx}$) решения уравнения

$$X \ = \ \tau.\alpha_1!. \ \ldots \alpha_n!. \ X$$

будет следовать соотношение

$$Sch' \ \stackrel{+}{\approx} (\tau \ \alpha_1! \ \ldots \alpha_n! \ )^*$$

из которого, в свою очередь, следует (6.19).

Мы будем преобразовывать левую часть соотношения (6.20) так, чтобы получилась правая часть этого соотношения. Для этого мы будем использовать свойства 8, 11 и 12 операций на процессах, сформулированные в параграфе 3.7. Напомним эти свойства:

- $P \setminus L = P$, **если** $L \cap names(Act(P)) = \emptyset$
- $(P_1 \mid P_2) \setminus L = (P_1 \setminus L) \mid (P_2 \setminus L)$, **если**

$$L \cap names(Act(P_1) \cap \overline{Act(P_2)}) = \emptyset$$

- $(P \setminus L_1) \setminus L_2 = P \setminus (L_1 \cup L_2) = (P \setminus L_2) \setminus L_1$

Используя данные свойства, можно преобразовать левую часть соотношения (6.20) следующим образом.

$$\begin{array}{l} Sch' = \\ = (Sch \mid (\beta_1?)^* \mid \ldots \mid (\beta_n?)^*) \setminus B = \\ = \left(\begin{array}{l} ((Start \mid C_1 \mid \ldots \mid C_n) \setminus \Gamma) \mid \\ \mid (\beta_1?)^* \mid \ldots \mid (\beta_n?)^* \end{array} \right) \setminus B = \\ = (Start \mid C_1' \mid \ldots \mid C_n') \setminus \Gamma \end{array} \quad (6.21)$$

где

$$C_i' = (C_i \mid (\beta_i?)^*) \setminus \{\beta_i\}$$

Заметим, что для каждого *i* = 1,..., *n* имеет место соотношение

$$C_i' \ \stackrel{+}{\approx} \ \gamma_i?. \ \alpha_i!. \ \gamma_{next(i)}!. \ C_i' \quad (6.22)$$

Действительно, по теореме о разложении

$$C'_i = ((\gamma_i?.\ \alpha_i!.\ \gamma_{next(i)}!.\ \beta_i!.\ C_i) \mid (\beta_i?)^*) \setminus \{\beta_i\} \sim$$
$$\sim \gamma_i?.\ \alpha_i!.\ \gamma_{next(i)}!.\ \tau.C'_i \overset{+}{\approx} \textbf{правая часть (6.22)}$$

Используя данное замечание и теорему о разложении, цепочку равенств (6.21) можно продолжить следующим образом:

$$(Start \mid C'_1 \mid C'_2 \mid \ldots \mid C'_n) \setminus \Gamma \overset{+}{\approx}$$
$$\overset{+}{\approx}(\underbrace{\gamma_1!.\ \mathbf{0}}_{=Start} \mid \underbrace{\gamma_1?.\ \alpha_1!.\ \gamma_2!.\ C'_1}_{\overset{+}{\approx}C'_1} \mid C'_2 \mid \ldots \mid C'_n) \setminus \Gamma \sim$$
$$\sim \tau.\ (\mathbf{0} \mid \alpha_1!.\ \gamma_2!.\ C'_1 \mid C'_2 \mid \ldots \mid C'_n) \setminus \Gamma =$$
$$= \tau.\ (\alpha_1!.\ \gamma_2!.\ C'_1 \mid C'_2 \mid \ldots \mid C'_n) \setminus \Gamma \sim$$
$$\sim \tau.\ \alpha_1!.\ (\gamma_2!.\ C'_1 \mid C'_2 \mid \ldots \mid C'_n) \setminus \Gamma \overset{+}{\approx}$$
$$\overset{+}{\approx} \tau.\ \alpha_1!.\ (\gamma_2!.\ C'_1 \mid \underbrace{\gamma_2?.\ \alpha_2!.\ \gamma_3!.\ C'_2}_{\overset{+}{\approx}C'_2} \mid \ldots \mid C'_n) \setminus \Gamma \sim \qquad (6.23)$$
$$\sim \tau.\ \alpha_1!.\ \tau.\ (C'_1 \mid \alpha_2!.\ \gamma_3!.\ C'_2 \mid \ldots \mid C'_n) \setminus \Gamma \sim \ldots \sim$$
$$\sim \tau.\ \alpha_1!.\ \tau.\ \alpha_2!.\ \ldots \tau.\ \alpha_n!.\ (C'_1 \mid \ldots \mid \gamma_1!.\ C'_n) \setminus \Gamma \overset{+}{\approx}$$
$$\overset{+}{\approx} \tau.\ \alpha_1!.\ \ldots \alpha_n!.\ (C'_1 \mid \ldots \mid \gamma_1!.\ C'_n) \setminus \Gamma \overset{+}{\approx}$$
$$\overset{+}{\approx} \tau.\ \alpha_1!.\ \ldots \alpha_n!.\ (\underbrace{\gamma_1?.\ \alpha_1!.\ \gamma_2!.\ C'_1}_{\overset{+}{\approx}C'_1} \mid \ldots \mid \gamma_1!.\ C''_n) \setminus \Gamma \sim$$
$$\sim \tau.\ \alpha_1!.\ \ldots \alpha_n!.\ \underbrace{\tau.\ (\alpha_1!.\ \gamma_2!.\ C'_1 \mid \ldots \mid C'_n) \setminus \Gamma}$$

Выражение, подчёркнутое фигурной скобкой в последней строке данной цепочки, совпадает с выражением в четвёртой строке этой цепочки, которое, в свою очередь, находится в отношении $\overset{+}{\approx}$ с Sch'.

Мы получили, что последнее выражение в цепочке (6.23)

- находится в отношении $\overset{+}{\approx}$ с правой частью соотношения (6.20), и
- с другой стороны, данное выражение находится в отношении $\overset{+}{\approx}$ с левой частью соотношения (6.20)

Таким образом, соотношение (6.20) доказано.

Читателю предоставляется в качестве упражнения доказать

- истинность условия (6.16), и
- соотношение $Sch \approx Sch_0$.

Читателю предлагается самостоятельно определить и доказать корректность планировщика, управляющего совокупностью $P_1, \ldots, P_n$ **процессов с приоритетами,** в которой каждому процессу P_i

сопоставлен некоторый **приоритет,** представляющий собой число $p_i \in [0, 1]$,причём $\sum_{i=1}^{n} p_i = 1$.

Планировщик должен реализовать такой режим работы процессов P_1, ..., P_n, при котором

• для каждого i = 1,..., n доля количества сеансов, выполненных процессом P_i, относительно общего количества сеансов, выполненных всеми процессами P_1,... ,P_n, асимптотически стремилась бы к p_i, при неограниченном увеличении времени работы процессов P_1 ..., P_n, причём

• данный планировщик должен обеспечивать максимальную свободу функционирования процессов P_1 ..., P_n.

6.5 Семафор

В этом примере, как и в примере из предыдущего параграфа, предполагается, что заданы n процессов

$$P_1, \ldots, P_n \quad (6.24)$$

причём для каждого i = 1,..., n процесс P_i имеет вид

$$P_i = (\alpha_i? \; a_{i1} \; \ldots \; a_{ik_i} \; \beta_i?)^*$$

где

• $\alpha_i?$ **и** $\beta_i?$ - служебные действия, представляющие собой сигналы о начале и о конце очередного сеанса работы процесса P_i, и

• $a_{i1}, \ldots, a_{ik_i}$ - собственные действия процесса P_i.

Мы хотели бы создать такой процесс P, в котором все процессы P_1, ..., P_n работали бы совместно, причём эта совместная работа должна подчиняться следующему режиму:

• если в некоторый момент времени какой-либо из процессов P_i начал очередной сеанс своей работы исполнением действия $\alpha_i?$,

• то этот сеанс должен быть **непрерываемым,** т.е. все последующие действия процесса P должны быть действиями процесса P_i, до тех пор, пока P_i не завершит этот сеанс.

Данное требование можно выразить в терминах трасс: каждая трасса процесса P должна иметь вид

$$\alpha_i? \; a_{i1} \; \ldots \; a_{ik_i} \; \beta_i? \; \alpha_j? \; a_{j1} \; \ldots \; a_{jk_j} \; \beta_i? \; \ldots$$

т.е. каждая трасса tr процесса P должна представлять собой конкатенацию трасс

$$tr_1 \cdot tr_2 \cdot tr_3 \ldots$$

каждая из которых представляет собой сеанс работы какого-либо процесса из (6.24).
Искомый процесс *P* мы определим следующим образом:

$$P := \ (\ P_1[f_1] \ | \ \ldots \ | \ P_n[f_n] \ | \ Sem \ ) \setminus \{\pi, \varphi\}$$

где
• *Sem* - процесс, предназначенный для задания требуемого режима работы процессов P_1 ..., P_n, данный процесс называется **семафором** и имеет вид

$$Sem \ = \ (\ \pi! \ \varphi! \ )^*$$

• $f_i : \alpha_i \mapsto \pi, \ \beta_i \mapsto \varphi$

Спецификация процесса *P* представляется следующим соотношением:

$$\begin{array}{l} P \ \overset{+}{\approx} \ \tau.a_{11}.\ldots.a_{1k_1}.\ P \ + \ldots + \\ \quad + \ \ \tau.a_{n1}.\ldots.a_{nk_n}.\ P \end{array} \qquad (6.25)$$

Доказательство того, что процесс *P* удовлетворяет этой спецификации, производится при помощи теоремы о разложении:

$$\begin{array}{l} P \ = \ (\ P_1[f_1] \ | \ \ldots \ | \ P_n[f_n] \ | \ Sem \ ) \setminus \{\pi, \varphi\} \sim \\ \sim \ \left(\begin{array}{l} \pi?.a_{11}.\ldots.a_{1k_1}.\varphi?.P_1[f_1] \ | \ \ldots \ | \\ \ | \ \pi?.a_{n1}.\ldots.a_{nk_n}\varphi?.P_n[f_n] \ | \\ \ | \ \pi!.\ \varphi!.\ Sem \end{array} \right) \setminus \{\pi, \varphi\} \sim \\ \sim \ \tau.\left(\begin{array}{l} a_{11}.\ldots.a_{1k_1}.\varphi?.P_1[f_1] \ | \ \ldots \ | \\ \ | \ \pi?.a_{n1}.\ldots.a_{nk_n}\varphi?.P_n[f_n] \ | \\ \ | \ \varphi!.\ Sem \end{array} \right) \setminus \{\pi, \varphi\} + \\ + \ldots + \\ + \tau.\left(\begin{array}{l} \pi?.a_{11}.\ldots.a_{1k_1}.\varphi?.P_1[f_1] \ | \ \ldots \ | \\ \ | \ a_{n1}.\ldots.a_{nk_n}\varphi?.P_n[f_n] \ | \\ \ | \ \varphi!.\ Sem \end{array} \right) \setminus \{\pi, \varphi\} \sim \\ \sim \ldots \sim \\ \sim \tau.a_{11}.\ldots.a_{1k_1}.\tau.\ P \ + \ldots + \tau.a_{n1}.\ldots.a_{nk_n}.\tau.\ P \overset{+}{\approx} \\ \overset{+}{\approx} \tau.a_{11}.\ldots.a_{1k_1}.\ P \ + \ldots + \tau.a_{n1}.\ldots.a_{nk_n}.\ P \end{array}$$

В заключение обратим внимание на следующий момент. Наличие префикса "τ." в каждом слагаемом в правой части соотношения (6.25) означает, что выбор альтернативы в начальный момент функционирования процесса *P* определяется

- не в окружающей среде процесса *P,*
- а внутри процесса *P.*

Если бы этого префикса не было, то это означало бы, что выбор альтернативы в начальный момент функционирования процесса *P* определяется окружающей средой процесса *P.*

7. Процессы с передачей сообщений

7.1 Действия с передачей сообщений

Введённое и изученное в предыдущих главах понятие процесса допускает различные обобщения.

Одно из таких обобщений заключается в добавлении к действиям из *Act* некоторых **параметров,** т.е. рассматриваются такие процессы, у которых выполняемые действия имеют вид

(a, p)

где $a \in Act$, и *p* - параметр, который может иметь, например следующий смысл:

- сложность (или стоимость) выполнения действия *a*
- приоритет действия *a* по отношению к другим действиям
- момент времени, в который произошло действие *a*
- длительность действия *a*
- вероятность совершения действия *a*
- или что-либо другое.

В настоящей главе мы рассматриваем один из вариантов такого обобщения, который связан с добавлением к действиям из *Act* параметров, представляющих собой **сообщения,** передаваемые при выполнении этих действий.

Напомним, что, согласно нашей неформальной интерпретации понятия действия,

- выполнение процессом действия вида $\alpha!$ заключается в передаче другому процессу объекта с именем *a,* и
- выполнение процессом действия вида $\alpha?$ заключается в получении от другого процесса объекта с именем *a.*

Обобщим данную интерпретацию следующим образом. Будем считать, что к объектам, которыми обмениваются процессы, можно добавлять **сообщения,** т.е. действия процессов могут иметь вид

$\alpha!v$ **и** $\alpha?v$ (7.1)

где $a \in Names$, и *v* - **сообщение,** которое может представлять собой

- число,

- символьную строку,
- банкноту,
- материальный ресурс,
- и т.п.

Выполняя действие вида $\alpha!v$ **или** $\alpha?v$, процесс передаёт или получает вместе с объектом *a* сообщение *v*.

Напомним, что понятие передаваемого объекта, как и понятия приёма и передачи, могут иметь виртуальный характер (более подробно см. параграф 2.3).

Для формального описания процессов, которые могут выполнять действия вида (7.1), мы обобщим понятие процесса.

7.2 Вспомогательные понятия

7.2.1 Типы, переменные, значения и константы

Мы будем предполагать, что задано множество *Types* **типов,** причём каждому типу $t \in Types$ сопоставлено множество D_t **значений** типа *t.*

Типы могут обозначаться идентификаторами. Для наиболее часто используемых типов существуют общепринятые идентификаторы, например,

- тип целых чисел обозначается int
- тип булевых значений (0 и 1) обозначается bool
- тип "символы" обозначается char
- тип "символьные строки" обозначается string

Также мы будем предполагать, что заданы следующие множества.

- Множество *Var,* элементы которого называются **переменными,** причём каждой переменной $x \in Var$ сопоставлен тип $t(x) \in Types.$

Каждая переменная $x \in Var$ может принимать **значения** в множестве $D_{t(x)}$, т.е. в различные моменты времени переменная *x* может быть связана с различными элементами множества $D_{t(x)}$.

- Множество *Con,* элементы которого называются **константами.**

Каждой константе $c \in Con$ сопоставлены

— тип $t(c) \in Types$, и

— значение $[\![c]\!] \in D_{t(c)}$, называемое **интерпретацией** константы *c*.

7.2.2 Функциональные символы

Мы будем предполагать, что задано множество **функциональных символов (ФС),** причём каждому ФС *f* сопоставлены

• **функциональный тип** *t*(*f*) (называемый ниже просто **типом),** представляющий собой знакосочетание

$$(t_1, \ldots, t_n) \to t \quad (7.2)$$

где $t_1, \ldots, t_n, t \in Types$, и

• частичная функция

$$[\![f]\!] : D_{t_1} \times \ldots \times D_{t_n} \to D_t$$

называемая **интерпретацией** ФС *f*.

Например, ниже мы будем рассматривать следующие ФС:

$$+, \quad -, \quad \cdot, \quad \mathbf{head}, \quad \mathbf{tail}, \quad [\,]$$

где

• ФС + и — имеют тип

$$(\mathtt{int}, \mathtt{int}) \to \mathtt{int}$$

функции $[\![+]\!]$ и $[\![-]\!]$ представляют собой соответствующие арифметические операции

• ФС • имеет тип

$$(\mathtt{string}, \mathtt{string}) \to \mathtt{string}$$

функция $[\![\cdot]\!]$ сопоставляет каждой паре строк *(u, v)* строку, получаемую приписыванием *v* к *u* справа (т.е. конкатенацию *u* и *v)*.

• **ФС head** имеет тип

$$\mathtt{string} \to \mathtt{char}$$

функция $[\![\mathbf{head}]\!]$ сопоставляет каждой непустой строке её первый символ

(значение функции $[\![\mathbf{head}]\!]$ на пустой строке не определено)

• **ФС tail** имеет тип

$$\mathtt{string} \to \mathtt{string}$$

функция $[\![\mathbf{tail}]\!]$ сопоставляет каждой непустой строке *u* строку, получаемую из *u* удалением её первого символа (значение функции $[\![\mathbf{tail}]\!]$ на пустой строке не определено)

• ФС [ ] имеет тип

$$\mathtt{char} \to \mathtt{string}$$

функция $[\![\,[\,]\,]\!]$ сопоставляет каждому символу строку, состоящую из одного этого символа

• ФС **length** имеет тип

$\mathtt{string} \to \mathtt{int}$

функция $[\![\mathbf{length}]\!]$ сопоставляет каждой строке её длину (т.е. количество символов в этой строке).

7.2.3 Выражения

Выражения строятся стандартным образом из переменных, констант и ФС. Каждое выражение *e* имеет тип $t(e) \in Types$, определяемый структурой этого выражения.

Правила построения выражений имеют следующий вид.

• Каждая переменная или константа является выражением того типа, который сопоставлен этой переменной или константе.

• Если

— f - ФС, имеющий тип (7.2), и

— $e_1,\ldots,e_n$ - выражения типов $t_1,\ldots,t_n$ соответственно

то знакосочетание $f(e_1,\ldots,e_n)$ является выражением типа ***t.***

Если каждой переменной *x,* входящей в выражение *e*, сопоставлено значение $\xi(x)$, то выражению *e* можно сопоставить значение, обозначаемое знакосочетанием $\xi(e)$, и определяемое стандартным образом:

• если $e = x$ (переменная), то $\xi(e) \stackrel{\text{def}}{=} \xi(x)$

(значение $\xi(x)$ предполагается заданным)

• если $e = c$ (константа), то $\xi(e) \stackrel{\text{def}}{=} [\![c]\!]$

• если $e = f(e_1,\ldots,e_n)$, то значение $\xi(e)$ выражения *e* определено, если

— все значения $\xi(e_1),\ldots,\xi(e_n)$ определены, и

— значение функции $[\![f]\!]$ определено на списке $(\xi(e_1),\ldots,\xi(e_n))$

в этом случае

$$\xi(e) \stackrel{\text{def}}{=} [\![f]\!](\xi(e_1),\ldots,\xi(e_n))$$

Ниже мы будем использовать следующие обозначения.

• Знакосочетание *Expr* обозначает совокупность всех выражений.

• Знакосочетание *Fm* обозначает совокупность тех выражений, которые имеют тип bool.

Выражения из *Fm* называются **формулами.**

При построении формул могут использоваться обычные булевские связки $(\wedge, \vee, \to$ **и т.д.**$)$, интерпретируемые стандартным образом.

Символ $\top$ обозначает тождественно истинную формулу, а символ $\bot$-тождественно ложную формулу.

Формулы вида $\wedge(b_1, b_2)$, $\vee(b_1, b_2)$, и т.п. мы будем записывать в более привычном виде $b_1 \wedge b_2$, $b_1 \vee b_2$, и т.д.

В некоторых случаях формулы вида

$$b_1 \wedge \ldots \wedge b_n \quad \text{и} \quad b_1 \vee \ldots \vee b_n$$

будут записываться в виде

$$\left\{ \begin{array}{c} b_1 \\ \ldots \\ b_n \end{array} \right\} \quad \text{и} \quad \left[\begin{array}{c} b_1 \\ \ldots \\ b_n \end{array} \right]$$

соответственно.

• Выражения вида $+(e_1, e_2)$, $-(e_1, e_2)$ и $\cdot(e_1, e_2)$ будут записываться в более привычном виде $e_1 + e_2$, $e_1 - e_2$ и $e_1 \cdot e_2$.

• Выражения вида $\mathbf{head}(e)$, $\mathbf{tail}(e)$, $|\ |(e)$, и $\mathbf{length}(e)$ будут записываться в виде $\hat{e}$, e', $[e]$ и $|e|$ соответственно.

• Костанта, интерпретацией которой является пустая строка, будет обозначаться символом *e.*

7.3 Понятие процесса с передачей сообщений

В этом параграфе мы излагаем понятие процесса с передачей сообщений. Данное понятие получается из исходного понятия процесса, изложенного в параграфе 2.4, следующей модификацией.

• К числу компонентов процесса добавляются

— компонента X_p, называемая **множеством переменных** этого процесса, и

— компонента I_p, называемая **начальным условием.**

• Метки переходов представляют собой не действия, а **операторы.**

Прежде чем излагать формальное определение понятия процесса с передачей сообщений, мы объясним смысл вышеперечисленных понятий.

Для краткости, мы будем в этой главе называть процессы с передачей сообщений просто **процессами.**

7.3.1 Множество переменных процесса

Мы будем предполагать, что с каждым процессом *P* связано множество переменных

$X_P \subseteq Var$

В каждый момент времени *i* работы процесса P $(i = 0, 1, 2, \ldots)$ каждой переменной $x \in X_P$ сопоставлено **значение** $\xi_i(x) \in D_{t(x)}$. Значения переменных могут изменяться во время работы процесса.

Означиванием переменных из X_p называется произвольный набор ξ значений, сопоставленных этим переменным, т.е. каждое означивание ξ имеет вид

$$\xi = \{\xi(x) \in D_{t(x)} \mid x \in X_P\}$$

Таким образом, в каждый момент времени *i* работы процесса *P* определено некоторое означивание ξ_i переменных из X_p.

Для каждого процесса *P* знакосочетание $Eval(X_p)$ обозначает совокупность всевозможных означиваний переменных из X_p.

Ниже мы будем предполагать, что для каждого процесса *P* все выражения, относящиеся к процессу *P,* содержат переменные только из множества X_p.

7.3.2 Начальное условие

Другой новой компонентой процесса *P* является формула $I_P \in Fm$, называемая **начальным условием.** Данная формула выражает условие на означивание ξ_0 переменных процесса *P* в начальный момент его работы: ξ_0 должно удовлетворять условию

$$\xi_0(I_P) = 1$$

7.3.3 Операторы

Главное отличие нового определения понятия процесса от старого заключается в том, что

• в старом определении метка каждого перехода является действием, которое совершает процесс при выполнении этого перехода, а

• в новом определении метка каждого перехода является **оператором,** который представляет собой **схему действия,** приобретающую вид конкретного действия лишь при конкретном выполнении этого оператора.

В определении понятия оператора мы будем использовать то же самое множество *Names,* которое было введено в параграфе 2.3.

Обозначим символом $\mathcal{O}$ множество, элементы которого называются **операторами,** и подразделяются на следующие четыре класса.

1. Операторы ввода, которые представляют собой знакосочетания вида

$\alpha ? x$ (7.3)

где $\alpha \in Names$ и $x \in Var.$

Действие, соответствующее оператору (7.3), выполняется путём ввода в процесс объекта, который имеет

- имя α. и
- дополнительный параметр, представляющий собой сообщение.

Введённое сообщение *v* записывается в переменную *x,* т.е. после исполнения данного действия значение переменной *x* становится равным *v.*

2. **Операторы вывода,** которые представляют собой знакосочетания вида

$\alpha ! e$ (7.4)

где $\alpha \in Names$ и $e \in Expr.$

Действие, соответствующее оператору (7.4), исполняется путём вывода из процесса объекта, который имеет

- имя α. и
- дополнительный параметр, представляющий собой сообщение вида *v,* которое равно значению выражения e на текущих значениях переменных процесса.

3. Операторы **присваивания** (первый вид внутренних операторов), которые представляют собой знакосочетания вида

$x := e$ (7.5)

где

- $x \in Var$, и
- $e \in Expr$, причём $t(e) = t(x)$

Действие, соответствующее оператору (7.5), исполняется путём обновления значения переменной *x:* после исполнения этого оператора её значение становится равным значению выражения e на текущих значениях переменных процесса *P*

4. Операторы **проверки условия** (второй вид внутренних операторов), которые представляют собой знакосочетания вида

$b ?$

где $b \in Fm.$

Действие, соответствующее такому оператору, исполняется путём вычисления значения формулы *b* на текущих значениях переменных процесса *P,* и

- если оно равно 0, то выполнение всего действия считается невозможным,
- иначе - выполнение действия считается завершённым.

7.3.4 Определение процесса

Процессом называется пятёрка *P* вида

$$P = (X_P, I_P, S_P, s_P^0, R_P) \quad (7.6)$$

компоненты которой имеют следующий смысл:

1. X_p - множество переменных процесса *P*
2. I_p - формула, называемая **начальным условием** процесса *P*
3. S_p - множество **состояний** процесса *P*
4. $s_P^0 \in S_P$ **начальное состояние**
5. R_p - подмножество вида

$$R_P \subseteq S_P \times \mathcal{O} \times S_P$$

Элементы множества R_p называются **переходами.**
Если переход из R_p имеет вид (s_1, op, s_2), то мы будем обозначать его знакосочетанием

$$s_1 \xrightarrow{op} s_2$$

и говорить, что

- состояние s_1 является **началом** этого перехода,
- состояние s_2 - его **концом,**
- оператор *op* является **меткой** этого перехода.

Также мы будем предполагать, что для каждого процесса *P* множество его переменных X_p содержит специальную переменную at_P, множеством значений которой является множество S_p состояний процесса *P*.

7.3.5 Функционирование процесса

Функционирование процесса *P* заключается в обходе множества его состояний (начиная с начального состояния s_P^0), с выполнением операторов, являющихся метками проходимых переходов.
Более подробно: на каждом шаге функционирования $i \geq 0$

- процесс находится в некотором состоянии s_i $(s_0 = s_P^0)$
- определено некоторое означивание ξ_i переменных процесса *P* $(\xi_0(I_P)$ должно быть равно 1)
- если есть хотя бы один переход из R_p с началом в s_i, то процесс

— недетерминированно выбирает переход с началом в s_i, помеченный таким оператором op_i, который можно выполнить в текущий момент времени,
(если таких переходов нет, то процесс временно приостанавливает свою работу до того момента, когда появится хотя бы один такой переход)
— выполняет оператор op_i, и после этого
— переходит в состоние s_{i+1}, которое является концом выбранного перехода
• если в R_p нет переходов с началом в s_i, то процесс заканчивает свою работу.
Для каждого $i \geq 0$ означивание ξ_{i+1} определяется
• по означиванию ξ_i , и
• по оператору op_i, который исполняется на i-м шаге функционирования процесса P.
Связь между означиваниями ξ_i, ξ_{i+1} и оператором op_i имеет следующий вид:
1. если $op_i = \alpha\,?\,x$, и при выполнении этого оператора в процесс было введено сообщение v, то

$$\xi_{i+1}(x) = v$$
$$\forall y \in X_P \setminus \{x, at_P\} \qquad \xi_{i+1}(y) = \xi_i(y)$$

2. если $op_i = \alpha\,!\,e$, то при выполнении этого оператора процесс выводит сообщение

$$\xi_i(e)$$

а значения переменных из $X_P \setminus \{at_P\}$ не изменяются:

$$\forall x \in X_P \setminus \{at_P\} \qquad \xi_{i+1}(x) = \xi_i(x)$$

3. если $op_i = (x := e)$, то

$$\xi_{i+1}(x) = \xi_i(e)$$
$$\forall x \in X_P \setminus \{x, at_P\} \qquad \xi_{i+1}(x) = \xi_i(x)$$

4. если $op_i = b\,?$ и $\xi_i(b) = 1$, то

$$\forall x \in X_P \setminus \{at_P\} \qquad \xi_{i+1}(x) = \xi_i(x)$$

Мы будем предполагать, что для каждого $i \geq 0$ значение переменной at_p при означивании ξ_i равно тому состоянию $s \in S_P$, в котором процесс P находится в момент времени i, т.е.

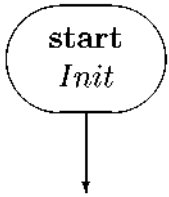
- $\xi_0(at_P) = s_P^0$
- $\xi_1(at_P) = s_1$, где s_1 – **конец первого перехода**
- $\xi_2(at_P) = s_2$, где s_2 – **конец второго перехода**
- **и т.д.**

7.4 Изображение процессов в виде блок-схем

В целях повышения наглядности, процессы иногда изображают в виде **блок-схем.**
Отметим, что язык блок-схем возник в программировании, где использование этого языка позволяет существенно облегчить описание и понимание функционирования алгоритмов и программ.

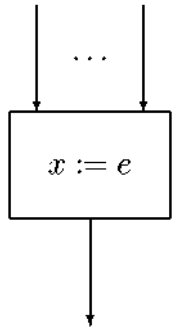
7.4.1 Понятие блок-схемы

Блок-схема представляет собой ориентированный граф, каждому узлу *n* которого сопоставлен **оператор** *op(n)* одного из перечисляемых ниже видов.
Каждый узел *n* блок-схемы изображается в виде геометрической фигуры (прямоугольника, овала или кружочка). Если *op(n)* не является оператором выбора или оператором соединения, то он изображается внутри этой фигуры.
оператор начала:



(7.7)

где *Init* - формула, называемая **начальным условием, оператор присваивания:**

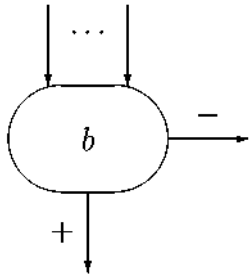


(7.8)

где

- $x \in Var$,
- $e \in Expr$, **причём** $t(e) = t(x)$

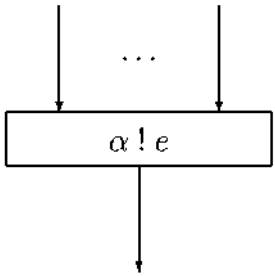
оператор условного перехода:



(7.9)

где $b \in Fm$.

оператор посылки сообщения:



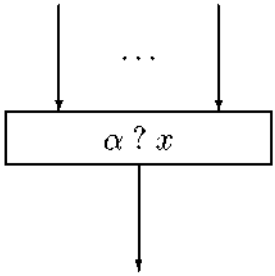
(7.10)

где

• α - имя (например, имя процесса, которому посылается сообщение), и

• *e* - выражение, значенем которого является посылаемое сообщение.

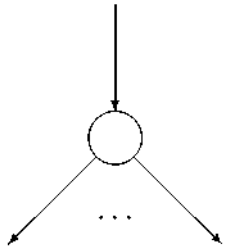
оператор получения сообщения:



(7.11)

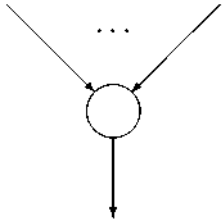
где
• α - имя (например, имя процесса, от которого приходит сообщение), и
• *x* - переменная, в которую следует записать получаемое сообщение.

оператор выбора:



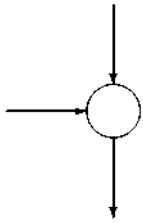
(7.12)

оператор соединения:

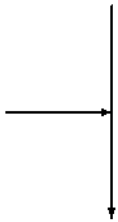


(7.13)

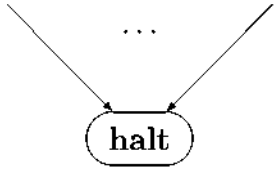
Иногда
• кружочек, изображающий оператор соединения, не рисуют,
• и также не рисуют концы некоторых стрелок, ведущих в этот оператор
т.е., например, фрагмент блок-схемы вида



может быть изображён следующим образом:



оператор остановки:



(7.14)

Блок-схемы должны удовлетворять следующим условиям:
• узел вида (7.7) может быть только один
• из узлов вида (7.7), (7.8), (7.10), (7.11), (7.13) выходит только одно ребро
• из узлов вида (7.9) выходят одно или два ребра, причём
— если из узла вида (7.9) выходит одно ребро, оно имеет метку "+", и
— если из узла вида (7.9) выходят два ребра, то одно из них имеет метку "+", а другое - метку "—"
• в узлы вида (7.12) входит только одно ребро
• из узлов вида (7.14) не выходит ни одного ребра

7.4.2 Функционирование блок-схемы

Функционирование блок-схемы представляет собой последовательность переходов от одного узла к другому по рёбрам, начиная с

узла n_0 вида (7.7) (называемого **начальным узлом),** с выполнением операторов, сопоставленных проходимым узлам.
Более подробно: каждый шаг функционирования $i \geq 0$ связан с некоторым узлом n_i, который называется **текущим узлом,** и
• если n_i не имеет вид (7.14), то после выполнения оператора, соответствующего узлу n_i , происходит переход по ребру, выходящему из n_i , к следующему узлу n_{i+1}, который будет текущим узлом на следующем шаге функционирования
• если же n_i имеет вид (7.14), то функционирование блок-схемы завершается.
Обозначим символом X совокупность всех переменных, входящих в блок-схему. На каждом шаге i функционирования ($i = 0,1,...$) каждой переменной $x \in X$ сопоставлено значение $\xi_i(x)$.
Совокупность значений переменных

$$\{\xi_i(x) \mid x \in X\}$$

обозначается символом ξ_i и называется **означиванием** переменных из X на i-м шаге функционирования блок-схемы. Значения переменных в начальный момент времени должны удовлетворять начальному условию, т.е. должно быть верно соотношение

$$\xi_0(Init) = 1$$

Операторы выполняются следующим образом.
• Оператор (7.8)
— вычисляет значение выражения e на текущем означивании ξ_i переменных из X, и
— заносит это значение в переменную x
т.е.

$$\xi_{i+1}(x) \stackrel{\text{def}}{=} \xi_i(e)$$
$$\forall\, y \in X \setminus \{x\} \quad \xi_{i+1}(y) \stackrel{\text{def}}{=} \xi_i(y)$$

• Оператор (7.9) вычисляет значение формулы b на текущем означивании ξ_i переменных из X, и
— если $\xi_i(b) = 1$, то происходит переход к следующему узлу по ребру с меткой "+",
— иначе -
* если из текущего узла выходят два ребра, то происходит переход к следующему узлу по ребру с меткой "—", и
* если из текущего узла выходит одно ребро, то в данный момент времени выполнение оператора, соответствующего текущему узлу, считается невозможным.

• Оператор (7.10) может выполниться в текущий момент времени только в том случае, когда в этот момент времени процесс может послать объект с именем α.
Если это возможно, то выполняется посылка

$$\alpha!\xi_i(e)$$

• Оператор (7.11) может выполниться в текущий момент времени только в том случае, когда в этот момент времени процесс может принять объект с именем *α*.
Если это возможно, то
— этот объект принимается,
— сообщение *v,* содержащееся в этом объекте, записывается в переменную *x,* т.е.

$$\xi_{i+1}(x) \stackrel{\text{def}}{=} v$$
$$\forall y \in X \setminus \{y\} \quad \xi_{i+1}(y) \stackrel{\text{def}}{=} \xi_i(y)$$

• Если текущий узел помечен оператором (7.12), то
— из рёбер, которые из него выходят, выбирается ребро, ведущее в узел, помеченный таким оператором, который возможно выполнить в текущий момент времени, и
— происходит переход в этот узел.
(если таких операторов, которые возможно выполнить в текущий момент времени, несколько, то выбор производится недетерминированно)
• Оператор (7.14) завершает функционирование блок-схемы.

7.4.3 Построение процесса, определяемого блок-схемой

Алгоритм построения процесса по блок-схеме имеет следующий вид.
1. На каждом ребре блок-схемы рисуется точка.
2. Для
• каждого узла *n* блок-схемы, который не имеет вида (7.12) или (7.13), и
• каждой пары F_1, F_2 рёбер блок-схемы, таких что F_1 входит в *n,* а F_2 - выходит из *n* выполняются следующие действия:
(a) рисуется стрелка *f*, соединяющая точку на F_1 с точкой на F_2,
(b) на этой стрелке *f* рисуется метка (*f*), определяемая следующим образом:
i. если *op(n)* имеет вид (7.8), то

$$\langle f \rangle \stackrel{\text{def}}{=} (x := e)$$

ii. если *op(n)* имеет вид (7.9), и ребро, выходящее из *n,* помечено символом "+", то

$$\langle f \rangle \stackrel{\text{def}}{=} b\,?$$

iii. если *op(n)* имеет вид (7.9), и ребро, выходящее из *n,* помечено символом "—", то

$$\langle f \rangle \stackrel{\text{def}}{=} \neg b\,?$$

iv. если *op(n)* имеет вид (7.10) или (7.11), то (*f*) совпадает с *op(n).*

3. Для каждого узла *n* вида (7.12) и каждого ребра *F,* выходящего из *n,* выполняются следующие действия:

• стрелка, соответствующая тому узлу, в который входит *F,* заменяется на стрелку

— с тем же концом и с той же меткой,

— но с началом - на ребре, входящем в *n*

• точка на ребре *F* удаляется.

4. Для каждого узла *n* вида (7.13) и каждого ребра *F,* входящего в *n,* выполняются следующие действия:

• стрелка, соответствующая тому узлу, из которого выходит *F,* заменяется на стрелку

— с тем же началом и с той же меткой,

— но с концом - на ребре, выходящем из *n*

• точка на ребре *F* удаляется.

5. Состояниями искомого процесса являются оставшиеся нарисованные точки.

6. Начальное состояние s^0_P определяется следующим образом.

• Если точка, нарисованная на том ребре блок-схемы, которое выходит из её начального узла, не была удалена, то эта точка является начальным состоянием s°_P.

• Если же эта точка была удалена, то нетрудно заметить, что это могло произойти только в том случае, когда концом ребра, выходящего из начального узла блок-схемы, является узел *n* вида (7.13). В этом случае начальным состоянием s^0_P является точка, нарисованная на ребре, выходящем из *n.*

7. Переходы процесса соответствуют нарисованным стрелкам: для каждой такой стрелки *f* процесс содержит переход

$$s_1 \xrightarrow{\langle f \rangle} s_2$$

где s_1 и s_2 - начало и конец стрелки *f* соответственно.

8. Множество переменных процесса содержит

• все переменные, входящие в какой-либо из операторов блок-схемы,

• а также переменную at_p.

9. Начальное условие процесса совпадает с начальным условием *Init* блок-схемы.

7.5 Пример процесса с передачей сообщений

В этом параграфе мы рассмотрим в качестве примера процесс "буфер":
• сначала мы определим этот процесс в виде блок-схемы, и
• затем мы построим по этой блок-схеме представление процесса "буфер" в стандартной форме.

7.5.1 Понятие буфера

Пусть *n* - некоторое положительное целое число.
Под **буфером размера** *n* мы в этом параграфе будем понимать систему (называемую ниже просто **буфером),** которая обладает следующими свойствами.
• В буфер можно вводить символы. Символы, введённые в буфер, можно выводить из буфера.
Если некоторый символ с введён в буфер, то мы будем говорить, что символ с **содержится** в буфере (до тех пор пока этот символ не будет выведен из буфера).
В буфере может одновременно содержаться не более *n* символов.
• В каждый момент времени совокупность символов, содержащихся в буфере, представляет собой упорядоченную последовательность

$$c_1, \ldots, c_k \qquad (0 \le k \le n) \quad (7.15)$$

которая называется **содержимым буфера.** Число *к* называется **размером** содержимого буфера.
Случай $k = 0$ соответствует ситуации, когда в буфере не содержится ни одного символа, в этом случае мы будем говорить, что **буфер пуст.**
• Если в текущий момент времени содержимое буфера имеет вид (7.15), и $k < n$, то в буфер можно ввести произвольный символ *c*, и после выполнения этой операции содержимое буфера примет вид

$$c_1, \ldots, c_k, c$$

• Если в текущий момент времени содержимое буфера имеет вид (7.15), и $k > 0$, то из буфера можно вывести символ, расположенный в начале его содержимого (т.е. c_1), после выполнения этой операции содержимое буфера примет вид

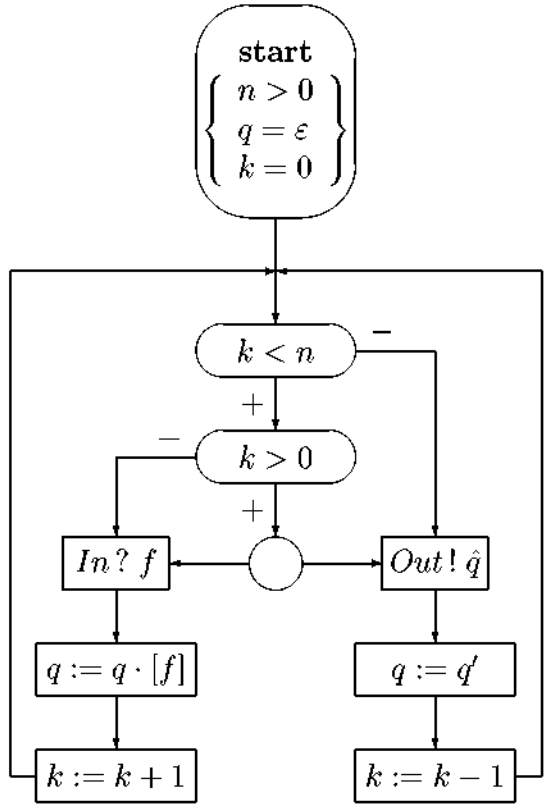
$$c_2, \ldots, c_k$$

Таким образом, в каждый момент времени содержимое буфера представляет собой очередь символов, причём

• каждая операция ввода символа в буфер добавляет вводимый символ в конец этой очереди, и
• каждая операция вывода символа из буфера
— выводит из буфера первый элемент этой очереди, и
— удаляет этот элемент из очереди
Очереди, операции с которыми выглядят описанным выше образом, называются **очередями типа FIFO** (First Input - First Output).

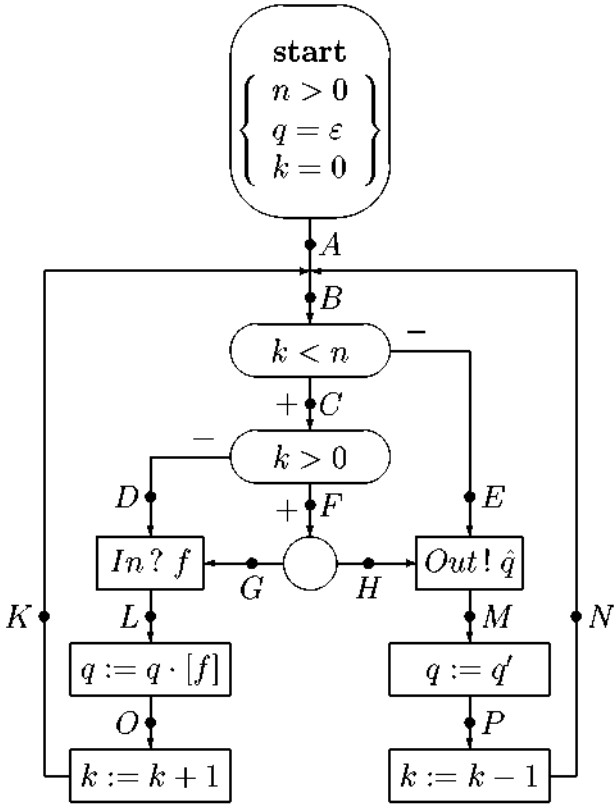
7.5.2 Представление поведения буфера в виде блок-схемы

Одним из возможных формальных уточнений понятия буфера является процесс $Buffer_n$, описание которого приводится в этом параграфе в виде блок-схемы. В этом процессе
• операция ввода символа в буфер представляется действием с именем *In*, и
• операция вывода символа из буфера представляется действием с именем *Out*.
Процесс $Buffer_n$ имеет следующие переменные:
• переменная *n* типа int, её значение не изменяется, оно равно максимальному размеру содержимого буфера
• переменная *k* типа int, её значение равно размеру содержимого буфера в текущий момент времени
• переменная *q* типа string, её значение равно содержимому буфера в текущий момент времени
• переменная *f* типа char, в эту переменную будут записываться вводимые символы при выполнении операции ввода.
Блок-схема, представляющая процесс $Bujfer_n$, имеет следующий вид (используемые в этой блок-схеме обозначения были определены в параграфе 7.2.3):



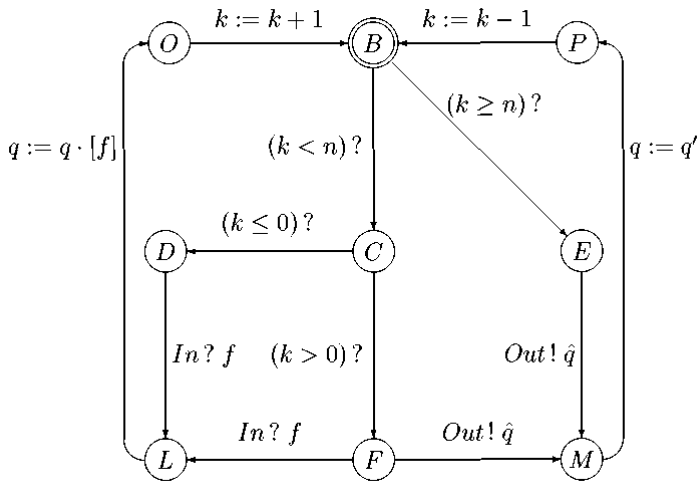
7.5.3 Представление поведения буфера в виде процесса

Для построения процесса $Buffer_n$, который соответствует определённой выше блок-схеме, мы нарисуем точки на её рёбрах:



При построении процесса по этой блок-схеме будут удалены точки *A*, *G*, *H*, *K* и *N*.

Искомый процесс $Buffer_n$ имеет следующий вид:



7.6 Операции на процессах с передачей сообщений

Операции на процессах с передачей сообщений аналогичны операциям на процессах в исходном смысле данного понятия (см. главу 3).

7.6.1 Префиксное действие

Пусть заданы процесс *P* и оператор *op.*
Процесс *op.P* получается из процесса *P* добавлением
• к множеству его состояний - нового состояния *s*, которое является начальным в *op.P,*
• к множеству переходов - нового перехода с меткой *op* из *s* в s^0_P
• к множеству переменных - всех переменных, входящих в *op.*

7.6.2 Альтернативная композиция

Пусть процессы P_1 и P_2 таковы, что $S_{P_1} \cap S_{P_2} = \emptyset$.
Тогда можно определить процесс $P_1 + P_2$, называемый **альтернативной композицией** процессов P_1 и P_2:
• множества его состояний, переходов, и начальное состояние определяются так же, как определяются соответствующие компоненты процесса $P_1 + P_2$ в главе 3

- $X_{P_1+P_2} \stackrel{\text{def}}{=} X_{P_1} \cup X_{P_2}$
- $I_{P_1+P_2} \stackrel{\text{def}}{=} I_{P_1} \wedge I_{P_2}$

Если же $S_{P_1} \cap S_{P_2} \neq \emptyset$, то для определения процесса $P_1 + P_2$ сначала надо заменить в P_2 те состояния, которые входят также и в P_1, на новые элементы, и модифицировать соответствующим образом другие компоненты P_2.

7.6.3 Параллельная композиция

Пусть процессы P_1 и P_2 таковы, что $X_{P_1} \cap X_{P_2} = \emptyset$.
Тогда можно определить процесс $P_1 \mid P_2$, называемый **параллельной композицией** процессов P_1 и P_2:
• множество его состояний и начальное состояние определяются так же, как определяются соответствующие компоненты процесса $P_1 \mid P_2$ в главе 3

- $X_{P_1+P_2} \stackrel{\text{def}}{=} X_{P_1} \cup X_{P_2}$
- $I_{P_1+P_2} \stackrel{\text{def}}{=} I_{P_1} \wedge I_{P_2}$

• множество переходов процесса $P_1 \mid P_2$ определяется следующим образом:
- для
* каждого перехода $s_1 \xrightarrow{op} s_1'$ процесса P_1, и
* каждого состояния s процесса P_2
процесс $P_1 \mid P_2$ содержит переход

$$(s_1, s) \xrightarrow{op} (s_1', s)$$

- для
* каждого перехода $s_2 \xrightarrow{op} s_2'$ процесса P_2, и
* каждого состояния s процесса P_1
процесс $P_1 \mid P_2$ содержит переход

$$(s, s_2) \xrightarrow{op} (s, s_2')$$

— для каждой пары переходов вида

$$s_1 \xrightarrow{op_1} s_1' \quad \in R_{P_1}$$
$$s_2 \xrightarrow{op_2} s_2' \quad \in R_{P_2}$$

где
* один из операторов *op1*, *op2* имеет вид $\alpha ? x$,
* а другой – $\alpha ! e$, причём $t(x) = t(e)$

(имя в обоих операторах - одно и то же)
процесс P1 | P2 содержит переход

$$(s_1, s_2) \xrightarrow{x := e} (s_1', s_2')$$

Если же $X_{P_1} \cap X_{P_2} \neq \emptyset$, то для определения процесса P1 | *P2* сначала надо заменить в одном из процессов те переменные, которые входят также и в другой процесс, на новые переменные.

7.6.4 Ограничение

Пусть заданы процесс P и подмножество $L \subseteq Names.$
Ограничением P по *L* называется процесс

$$P \setminus L$$

который получается из P удалением тех переходов, метки которых содержат имена из *L*.

7.6.5 Переименование

Пусть заданы процесс *P* и функция $f : Names \to Names.$
Действие операции переименования [*f*] на процесс *P* заключается в изменении имён в метках переходов: каждое имя α в какой-либо из этих меток заменяется на $f(\alpha)$.

Получившийся процесс обозначается знакосочетанием $P[f]$. Если *f* действует нетождественно лишь на имена из списка

$$\alpha_1, \ldots, \alpha_n$$

и отображает их в имена

$$\beta_1, \ldots, \beta_n$$

соответственно, то для $P[f]$ мы будем иногда использовать эквивалентное обозначение

$$P[\beta_1/\alpha_1, \ldots, \beta_n/\alpha_n]$$

7.7 Эквивалентность процессов

7.7.1 Понятие конкретизации процесса

Пусть *P* - произвольный процесс. Обозначим знакосочетанием *Conc(P)* процесс в исходном смысле данного понятия (см. параграф 2.4), который называется **конкретизацией** процесса *P,* и имеет следующие компоненты.

1. Состояниями *Conc(P)* являются
- всевозможные означивания из $Eval(X_p)$,
- а также дополнительное состояние s^0, которое является начальным в *Conc(P)*

2. Для
- каждого перехода $s_1 \xrightarrow{op} s_2$ процесса *P,* и
- каждого означивания $\xi \in Eval(X_P)$, такого, что

$$\xi(at_P) = s_1$$

Conc(P) содержит переход

$$\xi \xrightarrow{a} \xi'$$

если $\xi'(at_P) = s_2$, и имеет место один из следующих случаев:

- $op = \alpha\,?\,x, \quad a = \alpha\,?\,v$, где v – произвольное значение из $D_{t(x)}$
  $\xi'(x) = v, \quad \forall y \in X_P \setminus \{x, at_P\} \qquad \xi'(y) = \xi(y)$
- $op = \alpha\,!\,e, \quad a = \alpha\,!\,\xi(e)$
  $\forall x \in X_P \setminus \{at_P\} \qquad \xi'(x) = \xi(x)$
- $op = \ (x := e), \quad a = \tau$
  $\xi'(x) = \xi(e), \quad \forall y \in X_P \setminus \{x, at_P\} \qquad \xi'(y) = \xi(y)$
- $op = \ b\,?, \quad \xi(b) = 1, \quad a = \tau$
  $\forall x \in X_P \setminus \{at_P\} \qquad \xi'(x) = \xi(x)$

3. Для
- каждого означивания $\xi \in Eval(X_P)$, такого, что

$$\xi(I_P) = 1$$

- и каждого перехода в *Conc(P)* вида $\xi \xrightarrow{a} \xi'$

Conc(P) содержит переход $s^0 \xrightarrow{a} \xi'$.

Из определений

• понятия функционирования процесса с передачей сообщений (см. пункт 7.3.5), и

• понятия функционирования процесса в исходном смысле (см. пункт 2.4)

следует, что имеется взаимно однозначное соответствие между

• множеством всех вариантов функционирования процесса *P,* и

• множеством всех вариантов функционирования его конкретизации *Conc(P).*

Читателю предлагается самостоятельно исследовать переста-новочность функции *Conc* с операциями на процессах, т.е. установить истинность или ложность соотношений вида

$$Conc(P_1 \mid P_2) = Conc(P_1) \mid Conc(P_2)$$

и т.п.

7.7.2 Понятие эквивалентности процессов

На множестве процессов с передачей сообщений можно определить те же отношения эквивалентности, которые можно определить для процессов в исходном смысле данного понятия (см. главу 4).

Мы будем считать, что любые два процесса с передачей сообщений P_1, P_2 по определению находятся в том же самом отношении эквивалентности $(\sim,\ \approx,\ \overset{+}{\approx},\ \ldots)$,в котором находятся их конкретизации, т.е.

$$P_1 \sim P_2 \quad \Leftrightarrow \quad Conc(P_1) \sim Conc(P_2), \quad \text{и т.д.}$$

Читателю предлагается самостоятельно

• исследовать связь операций на процессах с различными от-ношениями эквивалентности, т.е. установить свойства, аналогичные свойствам, изложенным в параграфах 3.7, 4.5, 4.8.4, 4.9.5

• сформулировать и обосновать необходимые и достаточные условия эквивалентности $(\approx, \overset{+}{\approx}, \ldots)$ процессов, не использующие понятие конкретизации процесса.

7.8 Процессы с составными операторами

7.8.1 Мотивировка понятия процесса с составными операторами

Сложность задачи анализа процесса существенно зависит от размера его описания (в частности, от количества состояний). Поэтому для построения эффективных алгоритмов анализа процессов необходим поиск методов понижения сложности описания анализируемых процессов. В этом параграфе мы рассматриваем один из таких методов. **Мы обобщаем понятие процесса до понятия процесса с составными операторами. Составной оператор является комбинацией нескольких обычных операторов. За счёт того, что мы объединяем последовательность обычных операторов в один составной, у нас появляется возможность исключить из описания процесса те состояния, в которых он находится на промежуточных шагах выполнения этой последовательности операторов.**
Мы определяем понятие редукции процессов с составными операторами, с таким расчётом, чтобы при выполнении редукции получался процесс,

- имеющий менее сложное описание, и
- эквивалентный (в некотором смысле) исходному процессу.

С использованием описанных выше понятий **задача анализа процесса может решаться следующим образом.**
1. Сначала мы сопоставляем исходному процессу *P* процесс *P′* с составными операторами, в некотором смысле совпадающий с *P* (говоря неформально, мы просто рассматриваем каждый оператор, входящий в *P,* как составной оператор).
2. Затем мы редуцируем *P′,* получая процесс *P″,* сложность которого может быть существенно меньше сложности исходного процесса *P.*
3. После этого мы выполняем анализ процесса *P″,* и по результатам этого анализа мы составляем заключение о свойствах исходного процесса *P.*

7.8.2 Понятие составного оператора

Составным оператором (СО) называется конечная последовательность *Op* операторов

$$Op = (op_1, \ldots, op_n) \qquad (n \geq 1) \quad (7.16)$$

обладающая следующими свойствами:

1. op_1 является оператором проверки условия,
формулу, входящую в этот оператор, мы будем обозначать знакосочетанием

$$cond\,(Op)$$

2. последовательность $(op_2, \ldots, op_n)$

- не содержит операторов проверки условия, и
- содержит не более одного оператора ввода или вывода.

Пусть *Op* - некоторый СО.

- *Op* называется **СО ввода** (или **вывода),** если среди операторов, входящих в *Op,* есть оператор ввода (или вывода).
- *Op* называется **внутренним,** если все операторы, входящие в *Op* - внутренние.
- Если *Op* является СО ввода или вывода, то знакосочетание

$$name\,(Op)$$

обозначает имя, входящее в *Op.*

- Если ξ - некоторое означивание переменных, входящих в *cond(Op),* то мы будем говорить, что *Op* **открыт на** ξ, если
ξ(cond(Op)) = 1

7.8.3 Понятие процесса с СО

Понятие **процесса с СО** отличается от понятия процесса из параграфа 7.3.4 только тем, что метки переходов у процесса с СО представляют собой СО.

7.8.4 Функционирование процесса с СО

Функционирование процесса с СО

- определяется почти так же, как определяется функционирование процесса в параграфе 7.3.5, и
- тоже представляет собой обход множества его состояний (начиная с начального состояния), с выполнением СО, являющихся метками проходимых переходов.

Пусть $P = (X_P, I_P, S_P, s_P^0, R_P)$ - процесс с СО.

На каждом шаге функционирования *i≥0*

- процесс *P* находится в некотором состоянии s_i $(s_0 = s_P^0)$
- определено некоторое означивание ξ_i переменных из X_p
$(\xi_0(I_P) = 1, \quad \xi_i(at_P) = s_i)$
- если есть хотя бы один переход из R_p с началом в s_i, то процесс

— недетерминированно выбирает переход с началом в s_i, помеченный таким СО Op_i, который обладает следующими свойствами:
* Op_i открыт на ξ_i
* если среди операторов, входящих в Op_i есть оператор вида

$$\alpha ? x \quad \textbf{или} \quad \alpha ! e$$

то процесс P может в текущий момент времени выполнить действие вида

$$\alpha ? v \quad \textbf{или} \quad \alpha ! v$$

соответственно
(если таких переходов нет, то процесс временно приостанавливает свою работу до того момента, когда появится хотя бы один такой переход)
— выполняет последовательно все операторы, входящие в Op_i, изменяя соответствующим образом текущее означивание после выполнения каждого оператора, входящего в Op_i, и после этого
— переходит в состояние s_{i+1}, которое является концом выбранного перехода
• если в R_P нет переходов с началом в s_i , то процесс заканчивает свою работу.

7.8.5 Операции на процессах с СО

Определения операций на процессах с СО почти совпадают с соответствующими определениями из параграфа 7.6, поэтому мы лишь укажем отличия в этих определениях.
• В определениях всех операций на процессах с СО вместо операторов участвуют СО.
• Определение операции | отличается только в том пункте, в котором определяются "диагональные" переходы. Для процессов с СО данный пункт выглядит следующим образом:
для каждой пары переходов вида

$$\begin{array}{ll} s_1 \xrightarrow{Op_1} s_1' & \in R_{P_1} \\ s_2 \xrightarrow{Op_2} s_2' & \in R_{P_2} \end{array}$$

где один из СО Op_1, Op_2 имеет вид

$$(op_1, \ldots, op_i, \alpha ? x, op_{i+1}, \ldots, op_n)$$

а другой -

$$(op_1', \ldots, op_j', \alpha ! e, op_{j+1}', \ldots, op_m')$$

где

$- \ t(x) = t(e),$

— подпоследовательности

$(op_{i+1}, \ldots, op_n)$ **и** $(op'_{j+1}, \ldots, op'_m)$

могут быть пустыми процесс $P_1 \mid P2$ содержит переход

$$(s_1, s_2) \xrightarrow{Op} (s'_1, s'_2)$$

где *Op* имеет вид

$$\left(\begin{array}{l} (cond\,(Op_1) \wedge cond\,(Op_2))\,?, \\ op_2, \ldots, op_i, \\ op'_2, \ldots, op'_j, \\ (x := e), \\ op_{i+1}, \ldots, op_n, \\ op'_{j+1}, \ldots, op'_m \end{array} \right)$$

7.8.6 Преобразование процессов с передачей сообщений в процессы с СО

Каждый процесс с передачей сообщений можно преобразовать в процесс с СО путём замены меток его переходов: для каждого перехода

$$s_1 \xrightarrow{op} s_2$$

его метка *op* заменяется на СО *Op,* определяемый следующим образом.

- Если *op* - оператор проверки условия, то

$$Op \stackrel{\text{def}}{=} (op)$$

- Если *op* - оператор присваивания, ввода или вывода, то

$$Op \stackrel{\text{def}}{=} (\top?, op)$$

где $\top$ - тождественно истинная формула.

Для каждого процесса с передачей сообщений *P* мы будем обозначать соответствующий ему процесс с СО тем же символом *P.*

7.8.7 Конкатенация СО

В этом параграфе мы вводим понятие **конкатенации** СО: для некоторых пар СО (Op_1, Op_2) мы определяем СО, обозначаемый знакосочетанием

$$Op_1 \cdot Op_2 \quad (7.17)$$

и называемый **конкатенацией** *Op₁* и *Op₂*.
СО (7.17) отражает идею последовательного выполнения операторов из *Op₁* и *Op₂:* в (7.17)
• сначала выполняются операторы, входящие в *Op₁,*
• а затем - операторы, входящие в *Op₂*.
Необходимым условием для того, чтобы можно было определить конкатенацию (7.17), является условие того, чтобы хотя бы один из СО *Op₁, Op2* был внутренним.
Ниже мы будем использовать следующие обозначения.
1. Для каждого
• $\text{СО } Op = (op_1, \ldots, op_n)$, и
• оператора присваивания *op*
знакосочетание *Op·op* обозначает СО

$$(op_1, \ldots, op_n, op) \quad (7.18)$$

2. Для каждого
• внутреннего $\text{СО } Op = (op_1, \ldots, op_n)$,и
• оператора ввода или вывода *op*
знакосочетание *Op ·op* обозначает СО (7.18)
3. Для каждого
• $\text{СО } Op = (op_1, \ldots, op_n)$, и
• оператора проверки условия *op* = *b*?
знакосочетание $Op \cdot op$ обозначает объект, который
• либо является СО,
• либо не определён.
Данный объект определяется рекурсивно следующим образом.
Если *n* = 1, то

$$Op \cdot op \stackrel{\text{def}}{=} ((cond(Op) \wedge b)\,?)$$

иначе -
• если op_n - оператор присваивания вида $(x := e)$,то

$$Op \cdot op \stackrel{\text{def}}{=} \underbrace{((op_1, \ldots, op_{n-1}) \cdot op_n(op))}_{(*)} \cdot op_n$$

где
— $op_n(op)$ - оператор проверки условия, получаемый из *op* заменой всех вхождений в него переменной *x* на выражение *e*
— если объект (*) не определён, то $Op \cdot op$ тоже не определён
• если op_n - оператор вывода, то $Op \cdot op$ есть СО

$$((op_1, \ldots, op_{n-1}) \cdot op) \cdot op_n \quad (7.19)$$

• если op_n - оператор ввода, и имеет вид $\alpha\,?\,x$, то $Op \cdot op$

— не определён, если *op* зависит от *x*

— равен СО (7.19), в противном случае.

Теперь можно сформулировать определение конкатенации СО. Пусть заданы два СО Op_1, Op_2, причём Op_2 имеет вид

$$Op_2 = (op_1, \ldots, op_n)$$

Мы будем говорить, что **определена конкатенация** СО Op_1 и Op_2, если выполнены следующие условия:

• хотя бы один из СО Op_1, Op_2 является внутренним

• определены все объекты в скобках в выражении

$$(\ldots((Op_1 \cdot op_1) \cdot op_2) \cdot \ldots) \cdot op_n \quad (7.20)$$

Если эти условия выполнены, то конкатенацией Op_1 и Op_2 называется СО

$$Op_1 \cdot Op_2$$

который равен значению выражения (7.20).

7.8.8 Редукция процессов с СО

Пусть *P* - процесс с СО.

Редукция процесса *P* представляет собой последовательность

$$P = P_0 \longrightarrow P_1 \longrightarrow \ldots \longrightarrow P_n \quad (7.21)$$

преобразований этого процесса, каждое из которых производится согласно какому-либо из излагаемых ниже правил. Каждое из этих преобразований (кроме первого) производится над результатом предыдущего преобразования.

Результатом редукции (7.21) является результат последнего из преобразований (т.е. процесс P_n).

Правила редукции имеют следующий вид.

Правило 1 (конкатенация).

Пусть *s* - некоторое состояние процесса с СО, которое не является начальным, и

• совокупность всех переходов этого процесса с концом *s* имеет вид

$$s_1 \xrightarrow{Op_1} s, \quad \ldots, s_n \xrightarrow{Op_n} s$$

• совокупность всех переходов этого процесса с началом *s* имеет вид

$$s \xrightarrow{Op'_1} s'_1, \quad \ldots, s \xrightarrow{Op'_m} s'_m$$

- $s \notin \{s_1, \ldots, s_n, s'_1, \ldots, s'_m\}$

• для каждого *i* = 1,..., *n* и каждого *j* = 1,..., *m* определена конкатенация $Op_i \cdot Op_j$

Тогда данный процесс можно преобразовать в процесс,
• состояниями которого являются состояния исходного процесса, за исключением *s*
• переходами которого являются
— те переходы исходного процесса, началом или концом которых не является s,
— а также переходы вида

$$s_i \xrightarrow{Op_i \cdot Op'_j} s'_j$$

для каждого *i* = 1,..., *n* и каждого *j* = 1,..., *m*
• — начальное состояние которого, а также
— множество переменных, и
— начальное условие
совпадают с соответствующими компонентами исходного процесса.

Правило 2 (склейка).
Пусть *P* - процесс с СО, который содержит два перехода с общим началом и общим концом

$$s_1 \xrightarrow{Op} s_2, \qquad s_1 \xrightarrow{Op'} s_2 \quad (7.22)$$

причём метки этих переходов различаются только в первой компоненте, т.е. *Op* и *Op'* имеют вид

$$Op = (op_1, op_2, \ldots, op_n)$$
$$Op' = (op'_1, op_2, \ldots, op_n)$$

Правило 2 заключается в замене пары переходов (7.22) на один переход вида

$$s_1 \xrightarrow{Op} s_2$$

где $Op = ((cond(Op) \vee cond(Op'))\,?, op_2, \ldots, op_n)$

Правило 3 (удаление несущественных присваиваний).
Пусть *P* - некоторый процесс с СО.
Обозначим знакосочетанием *op(P)* совокупность всех операторов, входящих в какой-либо из СО процесса *P*.
Будем называть переменную $x \in X_P$ **несущественной,**
если
• *x* не входит ни в один из
— операторов проверки условия, и

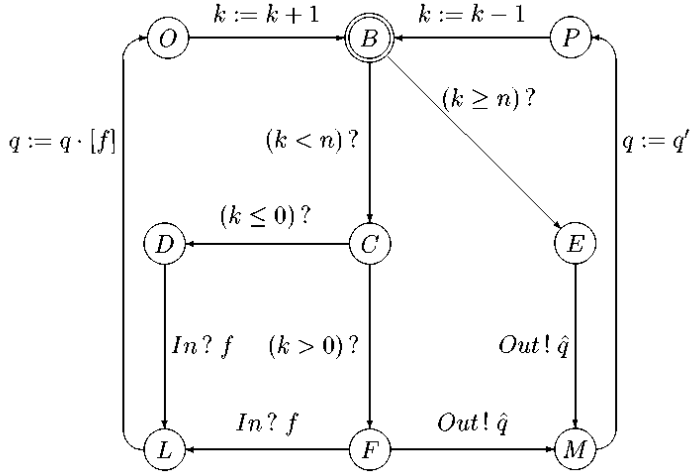
— операторов вывода
из *op(P)*
• если *x* входит в правую часть какого-либо оператора присваивания из *op(P)* вида $(y := e)$, то переменная *y* - несущественная.
Правило 3 заключается в удалении из всех СО редуцируемого процесса операторов присваивания вида $(x := e)$,где переменная *x* - несущественная.

7.8.9 Пример редукции

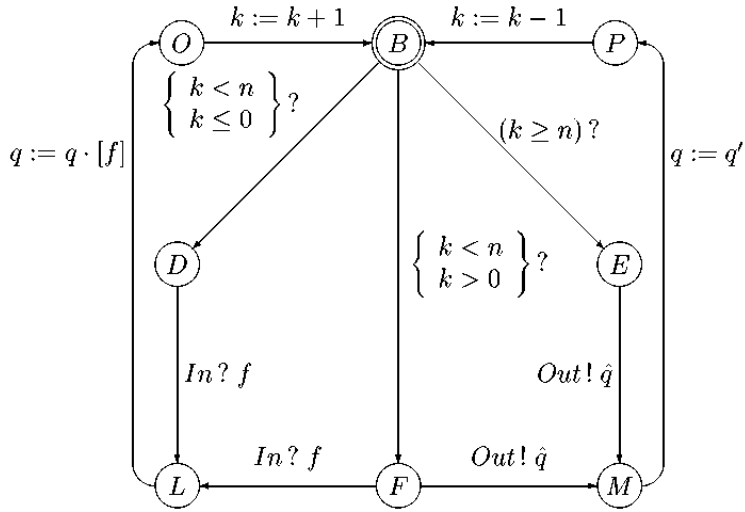
Рассмотрим в качестве примера редукцию процесса $Buffer_n$ (графовое представление которого приведено в параграфе 7.5.3). Ниже мы будем использовать следующее соглашение:
• если в СО *Op*

$$cond\,(Op) = \top$$

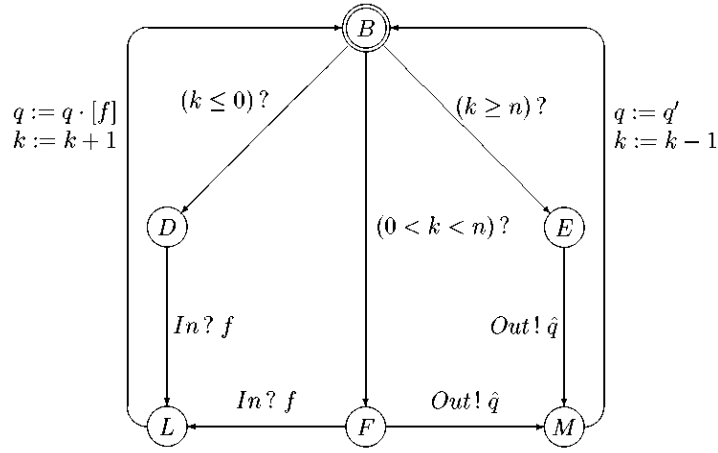
то первый оператор в таком СО мы писать не будем
• операторы, входящие в СО, можно располагать не только по горизонтали, но и по вертикали
• скобки, в которые заключена последовательность операторов, из которых состоит СО, можно опускать.
Исходный процесс $Buffer_n$ имеет следующий вид:



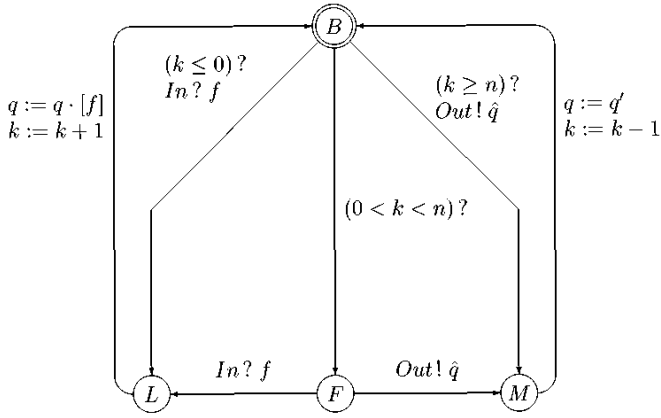
Первый шаг редукции заключается в удалении состояния *C* (применяется правило 1 для *s* = C):



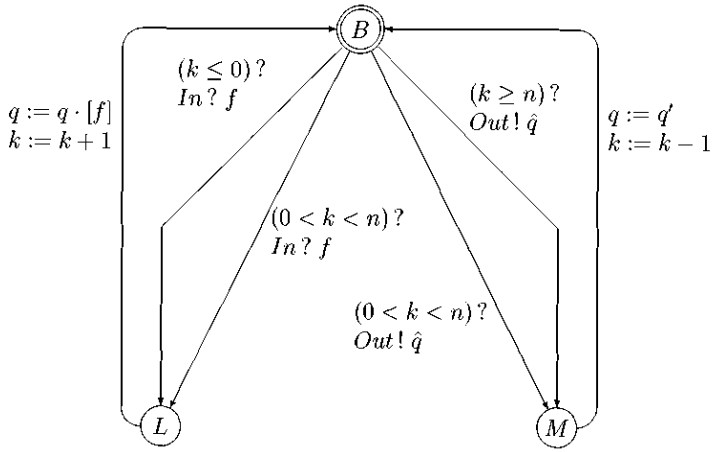
Поскольку $n > 0$, то формулу $(k < n) \wedge (k \le 0)$ в метке перехода из B в D можно заменить на равносильную формулу $k \le 0$. Второй и третий шаги редукции - удаление состояний O и P:



Четвёртый и пятый шаги редукции - удаление состояний D и E:

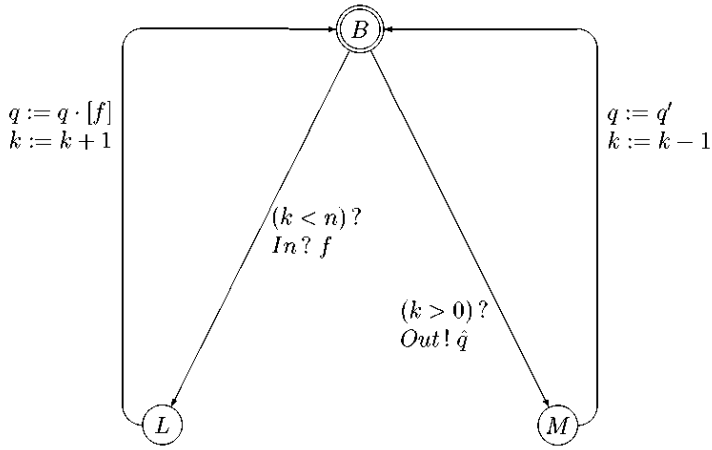


Шестой шаг редукции - удаление состояния *F:*

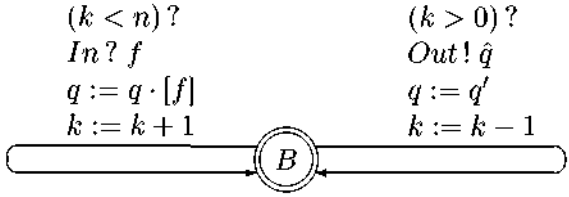


Седьмой и восьмой шаги редукции - применение правила 2 к переходам вида $B \to L$ и $B \to M$. В получившемся процессе мы заменяем

- формулу $(0 < k < n) \vee (k \le 0)$ - на равносильную ей формулу *(k <n)*
- формулу $(0 < k < n) \vee (k \ge n)$ - на равносильную ей формулу *(k* > 0)



Девятый и десятый шаги редукции - удаление состояний *L* и *M.* В результате мы получаем не редуцируемый далее процесс с СО с одним состоянием:



(7.23)

7.8.10 Понятие конкретизации процесса с СО

Для процессов с СО можно определить понятие конкретизации, аналогично тому, как это было сделано для обычных процессов с передачей сообщений в параграфе 7.7.1.

Для каждого процесса с СО *P* знакосочетание *Conc(P)* обозначает процесс в исходном смысле данного понятия (см. параграф 2.4), который называется **конкретизацией** процесса *P,* и имеет следующие компоненты.

1. Состояниями *Conc(P)* являются

• всевозможные означивания из $Eval(X_P)$,

• а также дополнительное состояние s^0, которое является начальным в *Conc(P)*

2. Для

• каждого перехода $s_1 \xrightarrow{Op} s_2$ процесса *P,* и

- каждого означивания $\xi \in Eval(X_P)$, такого, что
  - $\xi(at_P) = s_1$, и
  - Op открыт на ξ

Conc(P) содержит переход

$$\xi \xrightarrow{a} \xi'$$

если $\xi'(at_P) = s_2$, и имеет место один из следующих случаев:

(a) *Op* - внутренний, $a = \tau$, и имеет место соотношение

$$\xi \xrightarrow{Op} \xi'$$

которое означает следующее: если *Op* имеет вид

$$(op_1, \ldots, op_n)$$

то существует последовательность ξ_1 ,...,ξ_n означиваний из *Eval*(X_p), такая, что

- $\forall x \in X_P \setminus \{at_P\} \quad \xi(x) = \xi_1(x), \quad \xi'(x) = \xi_n(x)$, и
- $\forall i = 2, \ldots, n$, если op_i имеет вид $(x := e)$, то

$$\xi_i(x) = \xi_{i-1}(e), \quad \forall y \in X_P \setminus \{x, at_P\} \quad \xi_i(y) = \xi_{i-1}(y)$$

(b)
- $Op = Op_1 \cdot (\alpha\,?\,x) \cdot Op_2$,
- $a = \alpha\,?\,v$, где v – произвольное значение из $D_{t(x)}$, и
- существуют означивания ξ_1 и ξ_2 из $Eval(X_P)$, такие, что

$$\xi \xrightarrow{Op_1} \xi_1\,, \qquad \xi_2 \xrightarrow{Op_2} \xi'$$
$$\xi_2(x) = v, \quad \forall y \in X_P \setminus \{x, at_P\} \qquad \xi_2(y) = \xi_1(y)$$

(c)
- $Op = Op_1 \cdot (\alpha\,!\,e) \cdot Op_2$,
- существует означивание ξ_1 из $Eval(X_P)$, такое, что

$$\xi \xrightarrow{Op_1} \xi_1\,, \qquad \xi_1 \xrightarrow{Op_2} \xi'\,, \qquad a = \alpha\,!\,\xi_1(e)$$

3. Для

- каждого означивания $\xi \in Eval(X_P)$, такого, что

$$\xi(I_P) = 1$$

• и каждого перехода в *Conc(P)* вида $\xi \xrightarrow{a} \xi'$ *Conc(P)* содержит переход $s^0 \xrightarrow{a} \xi'$.

Читателю предлагается самостоятельно исследовать взаимовязь между
• конкретизацией произвольного процесса с передачей сообщений *P,* в том смысле, в каком это понятие было определено в параграфе 7.7.1, и
• конкретизацией процесса с СО, полученного в результате редукции процесса *P.*

7.8.11 Отношения эквивалентности на процессах с СО

Пусть P_1 и P_2 - процессы с СО.
Мы будем говорить, что P_1 и P_2 **наблюдаемо эквивалентны,** и обозначать этот факт знакосочетанием

$$P_1 \approx P_2$$

если их конкретизации наблюдаемо эквивалентны (в исходном смысле данного понятия, см. параграф 4.8).

Аналогичным образом определяется отношение $\overset{+}{\approx}$ наблюдаемой конгруэнции на процессах с СО.
Используя понятие редукции процессов с СО, можно определить ещё одно отношение эквивалентности на множестве процессов с СО. Данное отношение
• обозначается символом $\overset{r}{\approx}$,и
• представляет собой минимальную конгруэнцию на множестве процессов с СО, обладающую следующим свойством: если P' получается из P в результате применения какого-либо правила редукции, то $P \overset{r}{\approx} P'$
(т.е. $\overset{r}{\approx}$ является пересечением всех конгруэнции на множестве процессов с СО, обладающих вышеуказанным свойством). Читателю предлагается самостоятельно
• исследовать связь операций на процессах с СО с отношениями $\approx$ **и** $\overset{+}{\approx}$, т.е. установить свойства, аналогичные свойствам, изложенным в параграфах 3.7, 4.5, 4.8.4, 4.9.5
• сформулировать и обосновать необходимые и достаточные условия наблюдаемой эквивалентности процессов с СО, не использующие понятие конкретизации

- исследовать взаимосвязь между отношениями $\approx$, $\overset{+}{\approx}$ **и** $\overset{r}{\approx}$
- найти такие правила редукции, чтобы было верно включение $\overset{r}{\approx} \subseteq \overset{+}{\approx}$

7.8.12 Метод доказательства наблюдаемой эквивалентности процессов с СО

Один из возможных методов доказательства наблюдаемой эквивалентности процессов с СО основан на излагаемой ниже теореме 34. Для формулировки этой теоремы мы введём вспомогательные понятия и обозначения.

1. Пусть *P* - процесс с СО.

Составной переход (СП) в *P* - это (возможно пустая) последовательность *CT* переходов процесса *P* вида

$$CT = \quad s_0 \xrightarrow{Op_1} s_1 \xrightarrow{Op_2} \ldots \xrightarrow{Op_n} s_n \qquad (n \geq 0) \quad (7.24)$$

такая, что

- среди $Op_1, \ldots, Op_n$ - не более одного СО ввода или вывода
- определена конкатенация

$$(\ldots(Op_1 \cdot Op_2) \cdot \ldots) \cdot Op_n$$

которую мы будем обозначать тем же символом *CT*.

Если последовательность (7.24) пуста, то её конкатенацией *CT* по определению является СО$(\top?)$.

Состояние s_0 называется **началом** СП (7.24), а состояние s_n - его **концом.**

Знакосочетание $s_0 \xrightarrow{CT} s_n$ является сокращённой записью утверждения о том, что *CT* - это

- СП с началом s_0 и концом s_n
- а также - СО, соответствующий этому СП.

2. Пусть φ **и** ψ - формулы.

Знакосочетание $\varphi \leq \psi$ является сокращённой записью утверждения о том, что формула $\varphi \to \psi$ является тождественно истинной.

3. Пусть $Op = (op_1, \ldots, op_n)$ - внутренний *СО, и φ* - формула.

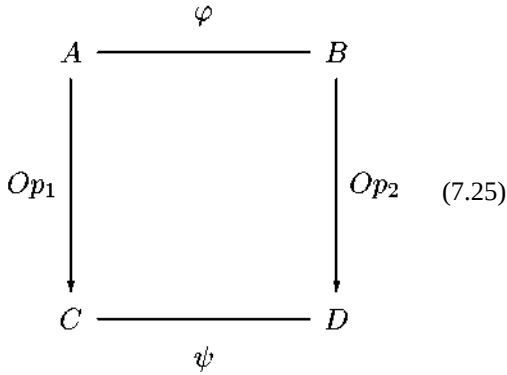
Знакосочетание *Op(φ)* обозначает формулу, определяемую рекурсивно:

$$Op(\varphi) \overset{\text{def}}{=} \begin{cases} cond\,(Op) \to \varphi, & \text{если } n = 1 \\ (op_1, \ldots, op_{n-1})\,(op_n(\varphi)), & \text{если } n > 1 \end{cases}$$

где $op_n(\varphi)$ обозначает формулу, определяемую следующим образом: если op_n имеет вид $(x := e)$, то $op_n(\varphi)$ получается из φ заменой каждого вхождения в неё переменной x на выражение e.

4. Пусть φ,ψ - формулы, и Op_1, Op_2 - СО.

Мы будем говорить, что верна диаграмма



(7.25)

если выполнено одно из следующих условий.

(a) Op_1 и Op_2 - внутренние СО, и верно неравенство

$$\varphi \leq (Op_1 \cdot Op_2)(\psi)$$

(b) Op_1 и Op_2 можно представить в виде конкатенации

$$Op_1 = Op_3 \cdot (\alpha\,?\,x) \cdot Op_4$$
$$Op_2 = Op_5 \cdot (\alpha\,?\,y) \cdot Op_6$$

где Op_3, Op_4, Op_5, Op_6- внутренние СО, и верно неравенство

$$\varphi \leq (Op_1' \cdot Op_2')(\psi)$$

где

- $Op_1' = Op_3 \cdot (x := z) \cdot Op_4$
- $Op_2' = Op_5 \cdot (y := z) \cdot Op_6$

• z - новая переменная (т.е. не входящая в $(\varphi,\psi, Op_1, Op_{P2})$

(c) Op_1 и Op_2 можно представить в виде конкатенации

$$Op_1 = Op_3 \cdot (\alpha\,!\,e_1) \cdot Op_4$$
$$Op_2 = Op_5 \cdot (\alpha\,!\,e_2) \cdot Op_6$$

где Op_3, Op_4, Op_5, Op_6 - внутренние СО, и верно неравенство

$$\varphi \leq \left\{ \begin{array}{l} (Op_3 \cdot Op_5)(e_1 = e_2) \\ (Op_3 \cdot Op_4 \cdot Op_5 \cdot Op_6)(\psi) \end{array} \right\}$$

Теорема 34.
Пусть P_1 и P_2 - два процесса с СО:

$$P_i = (X_{P_i}, I_{P_i}, S_{P_i}, s^0_{P_i}, R_{P_i}) \qquad (i = 1, 2)$$

не имеющие общих состояний и общих переменных.
P_1 и P_2 находятся в отношении $\approx$, если существует функция μ вида

$$\mu : S_{P_1} \times S_{P_2} \to Fm$$

обладающая следующими свойствами.

1. $I_{P_1} \wedge I_{P_2} \leq \mu(s^0_{P_1}, s^0_{P_2})$.

2. Для

- каждой пары $(A_1, A_2) \in S_{P_1} \times S_{P_2}$, и
- каждого перехода $A_1 \xrightarrow{Op} A'_1$ процесса P_1, такого, что

$$cond\,(Op) \ \wedge \ \mu(A_1, A_2) \neq \bot \quad (7.26)$$

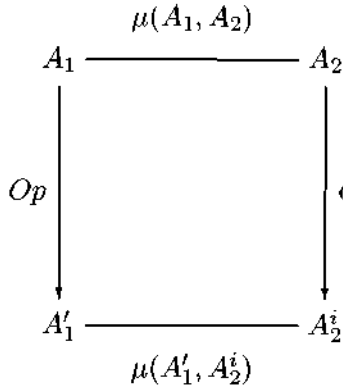
существует совокупность СП процесса P_2 с началом A_2

$$\{ A_2 \xrightarrow{CT_i} A_2^i \mid i \in \Im \} \quad (7.27)$$

удовлетворяющая следующим условиям:
(a) имеет место неравенство

$$cond\,(Op) \ \wedge \ \mu(A_1, A_2) \ \leq \ \bigvee_{i \in \Im} cond\,(CT_i) \qquad (7.28)$$

(b) для каждого $i \in \Im$ верна диаграмма



(7.29)

3. Свойство, симметричное предыдущему: для
- каждой пары $(A_1, A_2) \in S_{P_1} \times S_{P_2}$, и
- каждого перехода $A_2 \xrightarrow{Op} A_2'$ процесса P_2, такого, что верно (7.26)

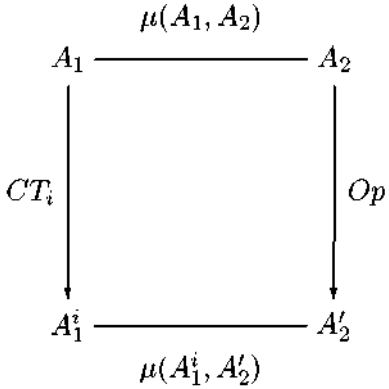
существует совокупность СП процесса P_1 с началом A_1

$$\{ A_1 \xrightarrow{CT_i} A_1^i \mid i \in \Im \} \quad (7.30)$$

удовлетворяющая следующим условиям:

(a) имеет место неравенство (7.28)

(b) для каждого $i \in \Im$ верна диаграмма



(7.31)

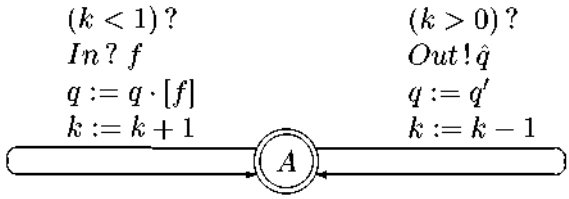
7.8.13 Пример доказательства наблюдаемой эквивалентности процессов с СО

В качестве примера использования теоремы 34 докажем наблюдаемую эквивалентность

$$Buffer_1 \approx Buf$$

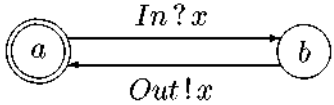
где

• *Buffer₁* - это рассмотренный выше процесс $Buffer_n$ (см. (7.23)) при $n = 1$, т.е. процесс, имеющий вид



его начальное условие -- $(k = 0) \wedge (q = \varepsilon)$, и

• *Buf* - процесс, имеющий вид



начальное условие этого процесса $= \top$.

Определим функцию $\mu : \{A\} \times \{a, b\} \to Fm$ следующим образом:

$$\mu(A, a) \stackrel{\text{def}}{=} (k = 0) \wedge (q = \varepsilon)$$
$$\mu(A, b) \stackrel{\text{def}}{=} (k = 1) \wedge (q = [x])$$

Проверим свойства 1, 2, и 3 для функции μ.

1. Свойство 1 в данном случае представляет собой неравенство

$$((k = 0) \wedge (q = \varepsilon)) \wedge \top \leq ((k = 0) \wedge (q = \varepsilon))$$

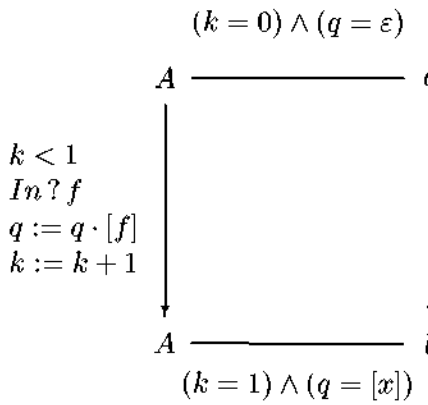
которое, очевидно, верно.

2. Проверим свойство 2.

• Для пары (*A. a*) нам надо рассмотреть лишь левый переход в процессе *Buffer₁* (так как для правого перехода не выполнено условие (7.26)).

В качестве (7.27) мы возьмем совокупность, состоящую из единственного перехода из *a* в *b*.

Диаграмма (7.29) в данном случае имеет вид



(7.32)

Пользуясь тем, что

$$\forall\,\varphi,\psi,\theta\in Fm \qquad (\varphi\le\psi\to\theta \quad\Leftrightarrow\quad \varphi\wedge\psi\le\theta) \tag{7.33}$$

запишем неравенство, соответствующее этой диаграмме, в виде

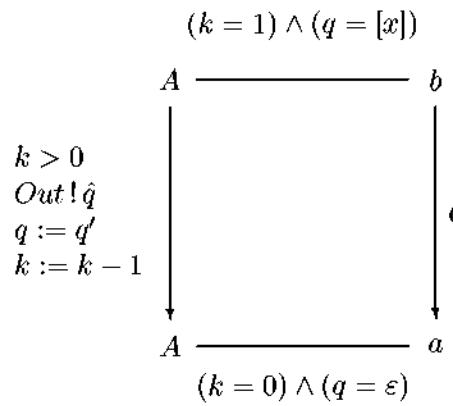
$$\left\{\begin{array}{l} k=0 \\ q=\varepsilon \\ k<1 \end{array}\right\} \le \left\{\begin{array}{l} k+1=1 \\ q\cdot[z]=[z] \end{array}\right\} \tag{7.34}$$

Очевидно, что данное неравенство верно.

• Для пары *(A, b)* нам надо рассмотреть лишь правый переход в процессе *Buffer*$_1$ (так как для левого перехода не выполнено условие (7.26)).

В качестве (7.27) мы возьмем совокупность, состоящую из единственного перехода из *b* в *a.*

Диаграмма (7.29) в данном случае имеет вид



(7.35)

Пользуясь (7.33), запишем неравенство, соответствующее этой диаграмме, в виде

$$\left\{\begin{array}{l} k=1 \\ q=[x] \\ k>0 \end{array}\right\} \leq \left\{\begin{array}{l} \hat{q}=x \\ k-1=0 \\ q'=\varepsilon \end{array}\right\} \qquad (7.36)$$

Очевидно, что это неравенство верно.

3. Проверим свойство 3.

• Для пары *(A, a)* и для единственного перехода из *a* в *b* в качестве (7.30) мы возьмем совокупность, состоящую из левого перехода из *A* в *A*.

Диаграмма (7.31) в данном случае имеет вид (7.32). Как уже было установлено, данная диаграмма верна.

• Для пары *(A, b)* и для единственного перехода из *b* в *a* в качестве (7.30) мы возьмем совокупность, состоящую из правого перехода из *A* в *A*.

Диаграмма (7.31) в данном случае имеет вид (7.35).

Как уже было установлено, данная диаграмма верна.

7.8.14 Дополнительные замечания

Для повышения удобства применения теоремы 34 можно использовать следующие понятия и утверждения.

Инварианты процессов

Пусть *P* - процесс с СО.

Формула *Inv* с переменными из X_p называется **инвариантом** процесса *P,* если она обладает следующими свойствами.

- $I_P \leq Inv$

• для каждого перехода $s \xrightarrow{Op} s'$ процесса *P*

— если *Op* - внутренний, то $Inv \leq Op(Inv)$

— если *Op* - СО ввода, и имеет вид $Op_1 \cdot (\alpha\,?\,x) \cdot Op_2$, то

$$Inv \leq (Op_1 \cdot (x := z) \cdot Op_2)(Inv)$$

где z - переменная, не входящая в X_p

— если *Op* - СО вывода, и имеет вид $Op_1 \cdot (\alpha\,!\,e) \cdot Op_2$, то

$$Inv \leq (Op_1 \cdot Op_2)(Inv)$$

С использованием понятия инварианта процесса теорему 34 можно модифицировать следующим образом.

Теорема 35.

Пусть

• P_1 и P_2 - два процесса с СО:

$$P_i = (X_{P_i}, I_{P_i}, S_{P_i}, s^0_{P_i}, R_{P_i}) \qquad (i = 1, 2)$$

не имеющие общих состояний и общих переменных, и

• формулы Inv_1 и Inv_2 являются инвариантами процессов P_1 и P_2 соответственно.

P_1 и P_2 находятся в отношении ≈, если существует функция μ вида

$$\mu : S_{P_1} \times S_{P_2} \to Fm$$

обладающая следующими свойствами.

1. $I_{P_1} \wedge I_{P_2} \leq \mu(s^0_{P_1}, s^0_{P_2})$.

2. Для

• каждой пары $(A_1, A_2) \in S_{P_1} \times S_{P_2}$, и

• каждого перехода $A_1 \xrightarrow{Op} A'_1$ процесса P_1, такого, что

$$\left\{ \begin{array}{l} cond\,(Op) \\ \mu(A_1, A_2) \\ Inv_1 \\ Inv_2 \end{array} \right\} \neq \perp \quad (7.37)$$

существует совокупность СП процесса P_2 с началом A_2

$$\{\, A_2 \xrightarrow{CT_i} A_2^i \mid i \in \Im\} \quad (7.38)$$

удовлетворяющая следующим условиям:

(a) имеет место неравенство

$$\left\{ \begin{array}{l} cond\,(Op) \\ \mu(A_1, A_2) \\ Inv_1 \\ Inv_2 \end{array} \right\} \leq \bigvee_{i \in \Im} cond\,(CT_i) \quad (7.39)$$

(b) для каждого $i \in \Im$ верна диаграмма

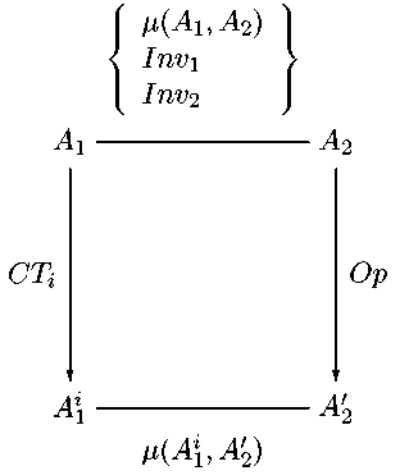
$$\begin{array}{ccc} & \left\{ \begin{array}{l} \mu(A_1, A_2) \\ Inv_1 \\ Inv_2 \end{array} \right\} & \\ A_1 & \longrightarrow & A_2 \\ \Big\downarrow Op & & \Big\downarrow CT_i \\ A_1' & \underset{\mu(A_1', A_2^i)}{\longrightarrow} & A_2^i \end{array} \quad (7.40)$$

3. Свойство, симметричное предыдущему: для
- каждой пары $(A_1, A_2) \in S_{P_1} \times S_{P_2}$, и
- каждого перехода $A_2 \xrightarrow{Op} A_2'$ процесса P_2, такого, что верно (7.37)

существует совокупность СП процесса P_1 с началом A_1

$$\{ A_1 \xrightarrow{CT_i} A_1^i \mid i \in \Im \} \quad (7.41)$$

удовлетворяющая следующим условиям:
(a) имеет место неравенство (7.39)
(b) для каждого $i \in \Im$ верна диаграмма

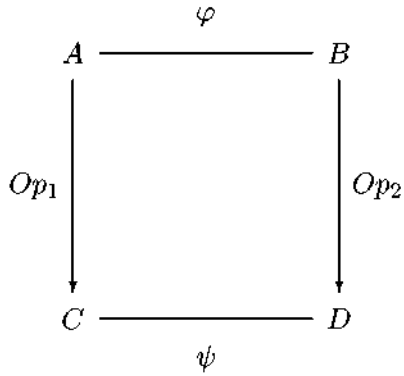


(7.42)

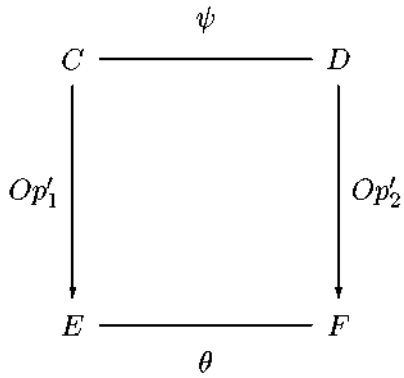
Композиция диаграмм
Теорема 36.
Пусть

- φ, ψ, θ - формулы

• Op_1, Op_2 - внутренние СО, такие, что верна диаграмма

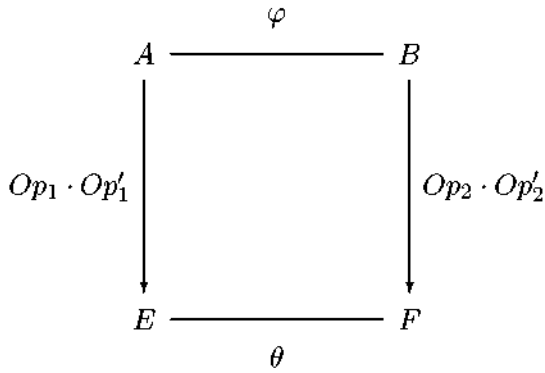


• Op'_1, Op'_2 - СО, такие, что верна диаграмма



• *{Op₁, Op'₁}* и *{Op₂, Op'₂}* не имеют общих переменных.

Тогда верна диаграмма



7.8.15 Другой пример доказательства наблюдаемой эквивалентности процессов с СО

В качестве примера использования теорем из параграфа 7.8.14 докажем наблюдаемую эквивалентность

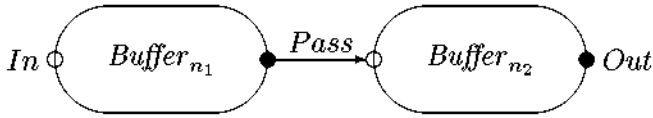
• процесса

$$(Buffer_{n_1}[Pass/Out] \mid Buffer_{n_2}[Pass/In]) \setminus \{Pass\} \quad (7.43)$$

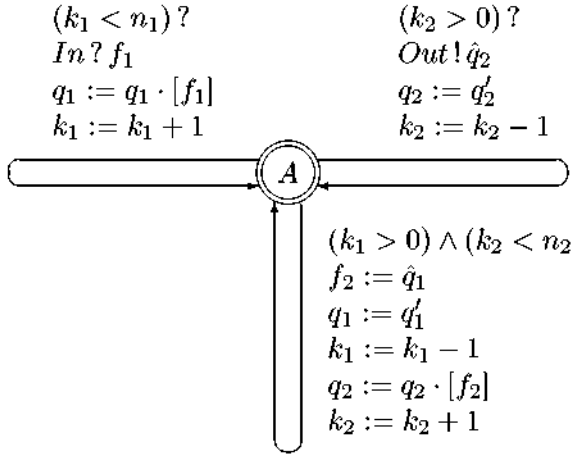
где $Pass \notin \{In, Out\}$,и

• процесса $Buffer_{n_1+n_2}$.

Процесс (7.43) представляет собой последовательную композицию двух буферов размеров n_1 и n_2. Его потоковый граф имеет вид



Согласно определению операций на процессах с СО (см. параграф 7.8.5), графовое представление процесса (7.43) выглядит следующим образом:

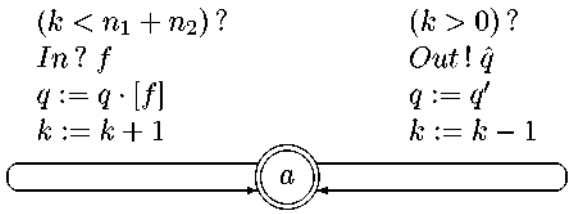


(7.44)

Начальным условием процесса (7.44) является формула

$$\left\{ \begin{array}{l} (n_1 > 0) \ \wedge \ (k_1 = 0) \ \wedge \ (q_1 = \varepsilon) \\ (n_2 > 0) \ \wedge \ (k_2 = 0) \ \wedge \ (q_2 = \varepsilon) \end{array} \right\}$$

Графовое представление процесса $Buffer_{n_1+n_2}$ имеет вид



Начальным условием процесса $Buffer_{n_1+n_2}$ является формула

$$(n_1 + n_2 > 0) \ \wedge \ (k = 0) \ \wedge \ (q = \varepsilon)$$

Нетрудно проверить, что формула

$$Inv \stackrel{\text{def}}{=} \left\{ \begin{array}{l} 0 \le k_1 \le n_1 \\ |q_1| = k_1 \\ 0 \le k_2 \le n_2 \\ |q_2| = k_2 \\ n_1 > 0 \\ n_2 > 0 \end{array} \right\}$$

является инвариантом процесса (7.44). Этот факт следует, в частности, из истинности для каждой строки *u* и каждого символа *a* соотношений

$$\left\{ \begin{array}{l} |u| > 0 \quad \Rightarrow \quad |u'| = |u| - 1 \\ |u \cdot [a]| = |[a] \cdot u| = |u| + 1 \end{array} \right.$$

В качестве инварианта второго процесса мы возьмём формулу $\top$.
Определим функцию $\mu : \{A\} \times \{a\} \to Fm$ следующим образом:

$$\mu(A, a) \stackrel{\text{def}}{=} \left\{ \begin{array}{l} q = q_2 \cdot q_1 \\ k = k_2 + k_1 \end{array} \right\}$$

Проверим свойства 1, 2, и 3 для функции μ.
1. Свойство 1 в данном случае представляет собой неравенство

$$\left\{ \begin{array}{l} (n_1 > 0) \ \wedge \ (k_1 = 0) \ \wedge \ (q_1 = \varepsilon) \\ (n_2 > 0) \ \wedge \ (k_2 = 0) \ \wedge \ (q_2 = \varepsilon) \\ (n_1 + n_2 > 0) \ \wedge \ (k = 0) \ \wedge \ (q = \varepsilon) \end{array} \right\} \le \left\{ \begin{array}{l} q = q_2 \cdot q_1 \\ k = k_2 + k_1 \end{array} \right\}$$

которое, очевидно, верно.
2. Проверим свойство 2.
• Для левого перехода процесса (7.44) неравенство (7.37) верно. В качестве (7.38) мы возьмем совокупность, единственным элементом которой является левый переход процесса $Buffer_{n_1+n_2}$.

Неравенство (7.39) в данном случае имеет вид

$$\left\{ \begin{array}{l} k_1 < n_1 \\ q = q_2 \cdot q_1 \\ k = k_2 + k_1 \\ Inv \end{array} \right\} \le (k < n_1 + n_2)$$

что, очевидно, верно.

Пользуясь (7.33), запишем неравенство, соответствующее диаграмме (7.40) для данного случая в виде

$$\left\{\begin{array}{l} q = q_2 \cdot q_1 \\ k = k_2 + k_1 \\ Inv \\ k_1 < n_1 \\ k < n_1 + n_2 \end{array}\right\} \leq \left\{\begin{array}{l} q \cdot [z] = q_2 \cdot q_1 \cdot [z] \\ k + 1 = k_2 + k_1 + 1 \end{array}\right\} \quad (7.45)$$

Нетрудно установить, что последнее неравенство верно.

• Для среднего (внутреннего) перехода процесса (7.44) неравенство (7.37) верно. В качестве (7.38) мы возьмем совокупность, единственным элементом которой является пустой СП процесса $Buffer_{n_1+n_2}$.

Неравенство (7.39) в данном случае верно по тривиальной причине: правая часть в нём имеет вид $\top$. Пользуясь соотношением (7.33), запишем неравенство, соответствующее диаграмме (7.40) для данного случая, в виде

$$\left\{\begin{array}{l} q = q_2 \cdot q_1 \\ k = k_2 + k_1 \\ Inv \\ k_1 > 0 \\ k_2 < n_2 \end{array}\right\} \leq \left\{\begin{array}{l} q = (q_2 \cdot [\hat{q}_1]) \cdot q_1' \\ k = k_2 + 1 + k_1 - 1 \end{array}\right\} \quad (7.46)$$

Данное неравенство следует из

— ассоциативности операции конкатенации, и

— истинности для каждой строки u соотношения

$$|u| > 0 \quad \Rightarrow \quad u = [\hat{u}] \cdot u'$$

• Для правого перехода процесса (7.44) неравенство (7.37) верно. В качестве (7.38) мы возьмем совокупность, единственным элементом которой является правый переход процесса $Buffer_{n_1+n_2}$.

Неравенство (7.39) в данном случае имеет вид

$$\left\{\begin{array}{l} k_2 > 0 \\ q = q_2 \cdot q_1 \\ k = k_2 + k_1 \\ Inv \end{array}\right\} \leq (k > 0)$$

что, очевидно, верно.

Пользуясь соотношением (7.33), запишем неравенство, соответствующее диаграмме (7.40) для данного случая, в виде

$$\left\{\begin{array}{l} q = q_2 \cdot q_1 \\ k = k_2 + k_1 \\ Inv \\ k_2 > 0 \\ k > 0 \end{array}\right\} \leq \left\{\begin{array}{l} \hat{q}_2 = \hat{q} \\ q' = q_2' \cdot q_1 \\ k - 1 = k_2 - 1 + k_1 \end{array}\right\} \quad (7.47)$$

Данное неравенство следует из истинности для каждой пары строк *u, v* соотношения

$$|u| > 0 \quad \Rightarrow \quad \left\{\begin{array}{l} (u \cdot v)\hat{} = \hat{u} \\ (u \cdot v)' = u' \cdot v \end{array}\right\}$$

3. Проверим свойство 3.

• Для левого перехода процесса $Buffer_{n_1+n_2}$ неравенство (7.37) верно. В качестве (7.41) мы возьмем совокупность, состоящую из двух СП:

— левого перехода процесса (7.44), и

— последовательности из пары переходов,

* первым элементов которой является средний (внутренний) переход процесса (7.44),

* а вторым - левый переход процесса (7.44)

Неравенство (7.39) в данном случае имеет вид

$$\left\{\begin{array}{l} k < n_1 + n_2 \\ q = q_2 \cdot q_1 \\ k = k_2 + k_1 \\ Inv \end{array}\right\} \leq (k_1 < n_1) \ \vee \ \left\{\begin{array}{l} k_1 > 0 \\ k_2 < n_2 \\ k_1 - 1 < n_1 \end{array}\right\}$$

Это неравенство верно (при его обосновании используется содержащийся в *Inv* конъюнктивный член $n_1 > 0$).

Истинность неравенств, соответствующих диаграммам (7.42) для обоих элементов совокупности (7.41), следует из (7.45), (7.46) и теоремы 36.

• Для правого перехода процесса $Buffer_{n_1+n_2}$ неравенство (7.37) верно. В качестве (7.41Jмы возьмем совокупность, состоящую из двух СП:

— правого перехода процесса (7.44), и

— последовательности из пары переходов,
* первым элементов которой является средний (внутренний) переход процесса (7.44),
* а вторым - правый переход процесса (7.44)
Неравенство (7.39) в данном случае имеет вид

$$\left\{\begin{array}{l} k>0 \\ q=q_2\cdot q_1 \\ k=k_2+k_1 \\ Inv \end{array}\right\} \le (k_2>0)\ \vee\ \left\{\begin{array}{l} k_1>0 \\ k_2<n_2 \\ k_2+1>0 \end{array}\right\}$$

Это неравенство верно (при его обосновании используется содержащийся в *Inv* конъюнктивный член $n_2>0$).
Истинность неравенств, соответствующих диаграммам (7.42) для обоих элементов совокупности (7.41), следует из (7.46), (7.47) и теоремы 36.

7.9 Рекурсивные определения процессов

Для процессов с передачей сообщений можно ввести понятие **рекурсивного определения,** которое аналогично понятию РО, изложенному в главе 5.
РО для процессов с передачей сообщений очень похожи на функциональные программы.
Понятие РО определяется на основе понятия **процессного выражения (ПВ),** которое почти совпадает с соответствующим понятием из параграфа 5.1, поэтому мы лишь укажем отличия в определениях этих понятий.
• Во всех ПВ вместо действий используются операторы.
• Каждое процессное имя *A* имеет **тип** *t(A),* который представляет собой последовательность обычных типов из *Types:*

$$t(A)=(t_1,\ldots,t_n) \qquad (n\ge 0)$$

• Каждое процессное имя *A* входит в каждое ПВ только вместе со списком выражений соответствующих типов, т.е. каждое вхождение процессного имени *A* в произвольное ПВ содержится в подвыражении вида

$$A(e_1,\ldots,e_n), \qquad \textbf{где } (t(e_1),\ldots,t(e_n))=t(A)$$

Каждому ПВ *P* соответствует совокупность *fv(P)* **свободных переменных** этого ПВ, которая состоит из всех переменных, имеющих свободные вхождения в *P.*

Понятие свободного и связанного вхождения переменной в ПВ определяется почти так же, как определяется аналогичное понятие в логике предикатов, только в случае ПВ роль кванторов играют операторы ввода и присваивания: каждое свободное вхождение переменной x в ПВ P становится связанным в ПВ $(\alpha?x).P$ и в ПВ $(x := e).P$.

Рекурсивное определение (РО) процессов представляет собой список формальных равенств вида

$$\begin{cases} A_1(x_{11}, \ldots, x_{1k_1}) = P_1 \\ \ldots \\ A_n(x_{n1}, \ldots, x_{nk_n}) = P_n \end{cases} \qquad (7.48)$$

где

- A_1,..., A_n - процессные имена,
- для каждого $i = 1,..., n$ список $(x_{i1}, \ldots, x_{ik_i})$ в левой части i-го равенства представляет собой список различных переменных
- P_1,..., P_n - ПВ, которые удовлетворяют

— условиям, изложенным в определении РО в параграфе 5.2,

— а также следующему условию: для каждого $i = 1,..., n$ совокупность $\{x_{i1}, \ldots, x_{ik_i}\}$ совпадает с совокупностью $fv(P_i)$ свободных переменных ПВ P_i.

РО (7.48) можно интерпретировать как функциональную программу, состоящую из рекурсивных определений функций

$$A_1(x_{11}, \ldots, x_{1k_1}), \ldots, A_n(x_{n1}, \ldots, x_{nk_n})$$

Для каждого $i = 1,..., n$ переменные $x_{i1}, \ldots, x_{ik_i}$ можно рассматривать как формальные параметры функции $A_i(x_{i1}, \ldots, x_{ik_i})$.

Читателю предлагается самостоятельно определить соответствие, которое сопоставляет каждому ПВ вида $A(x_1, \ldots, x_n)$, где

- A - процессное имя, и
- $x_1,...,x_n$ - список различных переменных соответствующих типов

процесс $[\![A(x_1, \ldots, x_n)]\!]$.

Примеры процессов, задаваемых в виде РО, будут изложены в главе 9.

Проблемы, связанные с РО в случае процессов с передачей сообщений, имеют тот же вид, что и для обычных РО:

- распознавание существования конечных процессов, эквивалентных процессам вида $[\![A(x_1, \ldots, x_n)]\!]$

• построение алгоритмов нахождения минимальных процессов, эквивалентных процессам вида $[\![A(x_1, \ldots, x_n)]\!]$ в том случае, когда эти процессы конечны

• распознавание эквивалентности процессов вида $[\![A(x_1, \ldots, x_n)]\!]$

• распознавание эквивалентности РО

• нахождение необходимых и достаточных условий единственности списка процессов, определяемого РО.

8. Примеры процессов с передачей сообщений

8.1 Разделение множеств

8.1.1 Задача разделения множеств

Пусть задана пара *U, V* конечных непересекающихся множеств, причём каждому элементу $x \in U \cup V$ сопоставлено некоторое число *w(x),* называемое **весом** этого элемента.
Требуется преобразовать эту пару в такую пару множеств *U', V',* чтобы

- $|U| = |U'|, \quad |V| = |V'|$

(для каждого конечного множества *M* знакосочетание |*M*| обозначает количество элементов в *M)*

• для каждого $u \in U'$ и каждого $v \in V'$ выполнялось неравенство $w(u) \leq w(v)$

8.1.2 Распределённый алгоритм решения задачи разделения множеств

Задачу разделения множеств можно решить путём выполнения нескольких сеансов обмена элементами между этими множествами. Каждый сеанс состоит из следующих действий:

• нахождение элемента *mx* с максимальным весом в левом множестве

• нахождение элемента *mn* с минимальным весом в правом множестве

• перенесение

— *mx* из левого множества в правое, и

— *mn* из правого множества в левое.

Для реализации этой идеи предлагается использовать распределённый алгоритм, определяемый как процесс вида

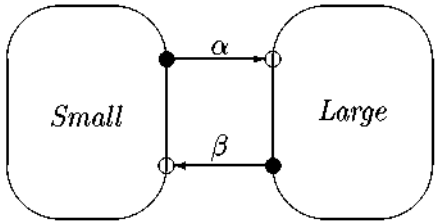
$$(Small \mid Large) \setminus \{\alpha, \beta\} \quad (8.1)$$

где

• процесс *Small* выполняет операции, связанные с левым множеством, и

• процесс *Large* выполняет операции, связанные с правым множеством.

Потоковый граф, соответствующий этому алгоритму, имеет вид



Ниже мы будем использовать следующие обозначения:

• для каждого подмножества $W \subseteq U \cup V$ знакосочетания max*(W)* и min(W) обозначают элемент множества *W* с максимальным и минимальным весом соответственно,

• для

— любых подмножеств $W_1, W_2 \subseteq U \cup V$, и

— любого $u \in U \cup V$

выражения

$$W_1 \leq u, \quad u \leq W_1, \quad W_1 \leq W_2$$

являются сокращённой записью выражений

$$\forall x \in W_1 \quad w(x) \leq w(u)$$
$$\forall x \in W_1 \quad w(u) \leq w(x)$$
$$\forall x \in W_1, \ \forall y \in W_2 \quad w(x) \leq w(y)$$

соответственно.

Аналогичный смысл имеют выражения

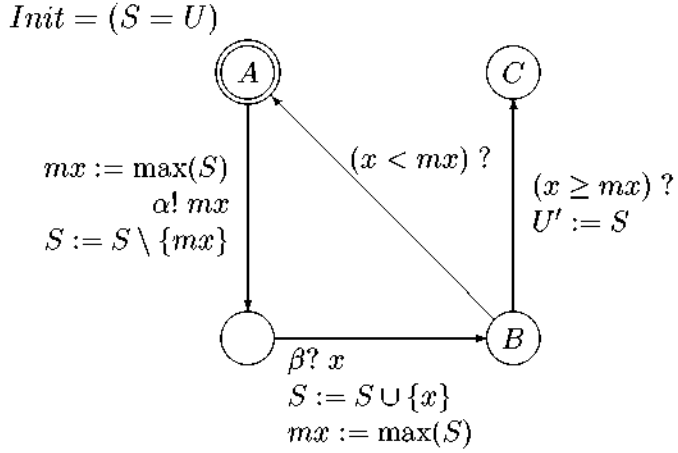
$$\max(W), \quad \min(W), \quad W \leq u, \quad u \leq W, \quad W_1 \leq W_2$$

в которых символы *W*, W_i и *u* обозначают переменные, значениями которых являются

- помножества множества $U \cup V$, и
- элементы множества $U \cup V$ соответственно.

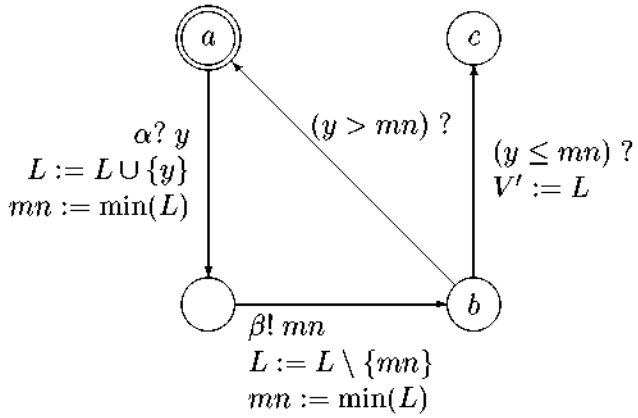
8.1.3 Процессы *Small* и *Large*

Процессы *Small* и *Large* могут быть определены в виде блок-схем, которые затем можно преобразовать в процессы с СО и редуцировать. Мы не будем давать описания этих блок-схем и излагать их преобразование в процессы с СО и этапы их редукции, приведём лишь соответствующие редуцированные процессы с СО.
Процесс *Small* имеет следующий вид:



(8.2)

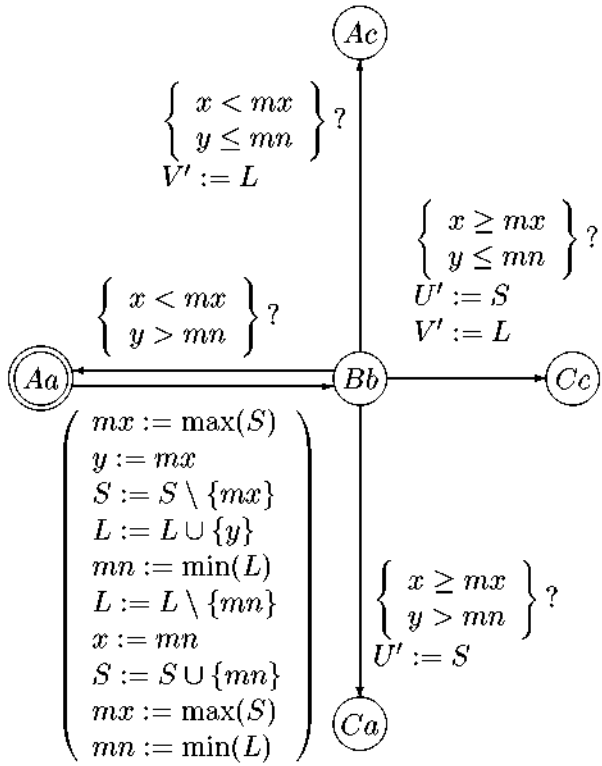
Процесс *Large* имеет следующий вид:
Init =(L = V)



(8.3)

8.1.4 Анализ алгоритма разделения множеств

Процесс, описываемый выражением (8.1), получается путём
• выполнения над процессами (8.2) и (8.3) операций параллельной композиции и ограничения, в соответствии с определением (8.1), и
• выполнения редукции получившегося процесса.
Редуцированный процесс выглядит следующим образом:



(8.4)

На данной диаграмме видно, что у процесса (8.4) имеются такие состояния *(Ac* и *Ca),*

- из которых не выходит ни одного перехода (такие состояния называются **терминальными),** т.е., попав в которые, процесс не может продолжать свою работу,
- но попадание в эти состояния не является нормальным завершением работы процесса.

Попадание процесса в такие состояния называется **тупиком,** или **дедлоком.**

Процесс (8.1) действительно может попасть в одно из таких состояний, например, при

$$U = \{3\} \quad \text{и} \quad V = \{1, 2\}$$

(где вес каждого числа совпадает с его значением).

Тем не менее, процесс (8.1) обладает следующими свойствами:

• данный процесс всегда завершает свою работу (т.е. попадает в одно из терминальных состояний - *Ac, Cc* или *Ca)*
• после завершения работы процесса выполняются соотношения

$$\left.\begin{array}{l} S \cup L = U \cup V \\ |S| = |U|, \;\; |L| = |V| \\ S \le L \end{array}\right\} \quad \textbf{(8.5)}$$

Для обоснования этих свойств мы будем использовать функцию

$$f(S,L) \stackrel{\text{def}}{=} |\,\{(s,l) \in S \times L \mid w(s) > w(l)\}\,|$$

Кроме того, при анализе последовательности операторов присваивания, выполняемых при переходе от состояния *Aa* к состоянию *Bb*, удобно представлять эту последовательность схематически как последовательность следующих действий:

1. $S \xrightarrow{y:=\max(S)} L$
   (**перенесение элемента** $y := \max(S)$ **из** S **в** L)
2. $L \xrightarrow{x:=\min(L)} S$
3. $mx := \max(S)$
4. $mn := \min(L)$

Нетрудно установить, что имеют место следующие утверждения.
1. Если в текущий момент времени *i*
• процесс находится в состоянии *Aa,* и
• значения S_i, L_i переменных *S* и *L* удовлетворяют равенству

$$f(S_i, L_i) = 0$$

т.е. имеет место соотношение

$$S_i \le L_i$$

то $S_{i+1} = S_i$ **и** $L_{i+1} = L_i$. Кроме того, после выполнения перехода от состояния *Aa* к состоянию *Bb* значения переменных *x, y, mx* и *mn* будут удовлетворять соотношениям

$$y = x = mx \le mn$$

и, таким образом, следующим переходом будет переход от состояния *Bb* к состоянию *Cc,* т.е. процесс нормально завершит свою работу. При этом
• значения переменных *U'* и *V'* будут равны S_i и L_i соответственно,

• и, следовательно, значения переменных *U'* и *V'* будут удовлетворять требуемым соотношениям.

2. Если в текущий момент времени *i*

• процесс находится в состоянии *Aa,* и

• значения $S_i|L_i$ переменных *S* и *L* в этот момент удовлетворяют неравенству

$$f(S_i, L_i) > 0$$

то после выполнения перехода от состояния *Aa* к состоянию *Bb* (т.е. в момент *i* + 1) новые значения S_{i+1}, L_{i+1} переменных *S* и *L* будут удовлетворять неравенству

$$f(S_{i+1}, L_{i+1}) < f(S_i, L_i) \quad \textbf{(8.6)}$$

Кроме того, значения переменных *x, y, mx, mn* в момент *i*+1 будут удовлетворять соотношениям

$$y = \max(S_i), \quad x = \min(L_i)$$
$$mx = \max(S_{i+1}), \quad mn = \min(L_{i+1})$$
$$x < y, \quad x \le mx, \quad mn \le y$$

Отсюда следует, что если в момент *i* +1 процесс будет переходить из состояния *Bb* в одно из терминальных состояний *(Ac, Cc* или *Ca),* то это возможно

(a) либо если *x* = *mx*

(b) либо если *y* = *mn*

В случае (a) имеют место соотношения

$$S_{i+1} \le mx = x \le L_i$$

откуда, используя

$$x < y \quad \text{и} \quad L_{i+1} \subseteq L_i \cup \{y\}$$

получаем:

$$S_{i+1} \le L_{i+1} \quad (8.7)$$

В случае (b) имеют место соотношения

$$S_i \le y = mn \le L_{i+1}$$

откуда, используя

$$x < y \quad \text{и} \quad S_{i+1} \subseteq S_i \cup \{x\}$$

получаем соотношение (8.7).

Таким образом, во всех возможных случаях попадания в какое-либо терминальное состояние имеет место соотношение

$$S \leq L$$

Истинность других соотношений, перечисленных в (8.5), усматривается непосредственно.

Из первого и второго утверждения следует, что данный процесс не может зациклиться, так как зацикливание возможно только в том случае, когда

- процесс бесконечно часто попадает в состояние *Aa,* и
- при каждом попадании в состояние *Aa* значение функции *f* на текущих значениях переменных *S, T* положительно.

Невозможность данной ситуации следует из

- неравенства (8.6), и
- свойства фундированности множества натуральных чисел (т.е. из того, что в данном множестве нет бесконечных убывающих цепей).

Читателю предлагается самостоятельно

- найти необходимые и достаточные условия, которым должны удовлетворять разделяемые множества *U* и *V,* чтобы приведённый выше алгоритм завершал свою работу с этими *U* и *V* нормально, и
- разработать такой алгоритм разделения множеств, который бы работал корректно на любых разделяемых множествах *U и V.*

8.2 Вычисление квадрата

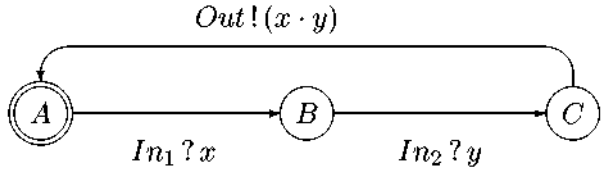
Предположим, что мы имеем систему "умножитель", у которой есть

- два входных порта с именами In_1 и In_2, и
- один выходной порт с именем *Out.*

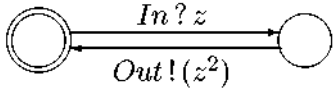
Работа умножителя заключается в том, что он периодически

- получает на свои входные порты два значения, и
- выдаёт на выходном порте их произведение.

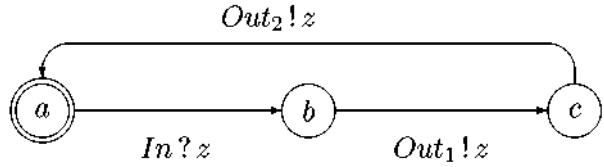
Поведение умножителя описывается процессом *Mul:*



Используя этот умножитель, мы хотим построить систему "вычислитель квадрата", поведение которой описывается процессом *Square _Spec:*



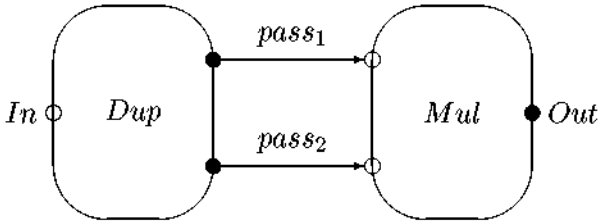
Искомую систему мы построим как композицию
• вспомогательной системы "дубликатор", имеющей
— входной порт *In*, и
— выходные порты Out_1 и Out_2
поведение которой описывается процессом *Dup:*



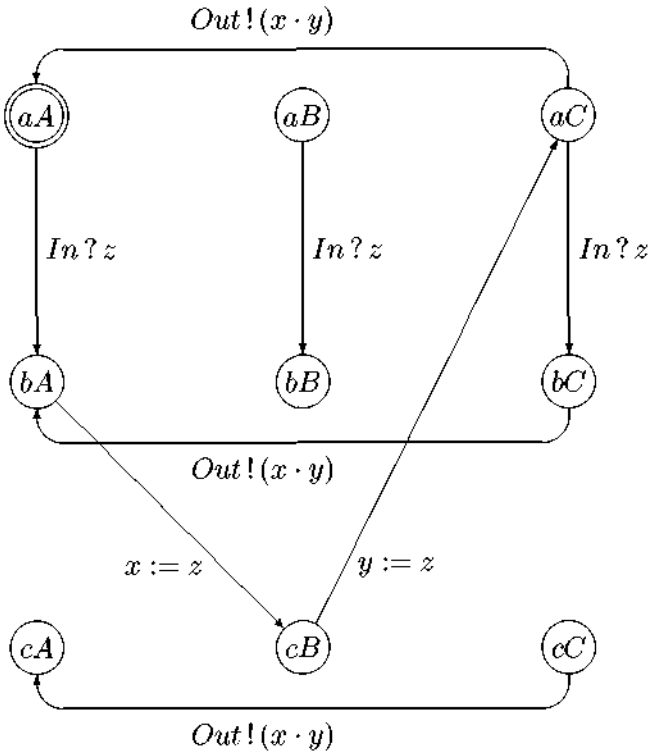
т.е. дубликатор копирует свой вход на два выхода, и
• умножителя, который будет получать на свои входные порты те значения, которые будет выдавать дубликатор.
Процесс *Square,* соответствующий такой композиции, определяется следующим образом:

$$Square \stackrel{\text{def}}{=} \stackrel{\text{def}}{=} \left(\begin{array}{l} Dup[pass_1/Out_1, pass_2/Out_2] \mid \\ \mid Mul[pass_1/In_1, pass_2/In_2] \end{array} \right) \setminus \{pass_1, pass_2\}$$

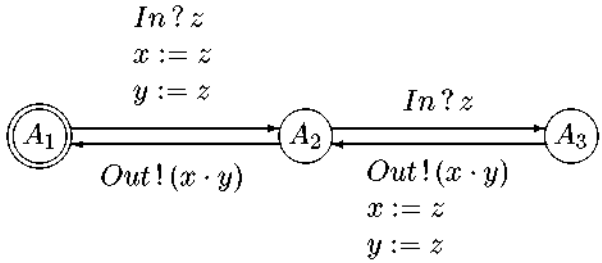
Потоковый граф процесса *Square* имеет вид



Однако процесс *Square* не соответствует своей спецификации *Square_Spec.* Данный факт нетрудно установить при помощи построения графового представления процесса *Square,* который, согласно определению операций параллельной композиции, ограничения и переименования, имеет следующий вид:



После редукции данного процесса мы получим диаграмму



(8.8)

из которой видно, что

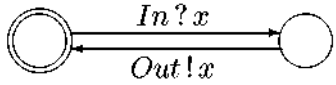
• процесс *Square* может последовательно совершить два действия ввода, не выполняя между ними действия вывода,

• а процесс *Square_Spec* этого сделать не может.

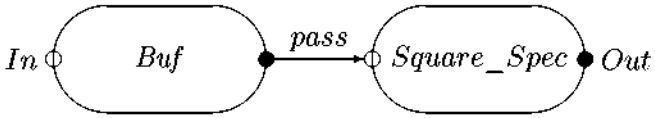
Процесс *Square* соответствует другой спецификации:

$$Square\_Spec' \stackrel{\text{def}}{=} \left(\begin{array}{l} Buf[pass/Out] \mid \\ \mid Square\_Spec[pass/In] \end{array} \right) \setminus \{pass\}$$

где *Buf* - буфер длины 1, поведение которого изображается диаграммой

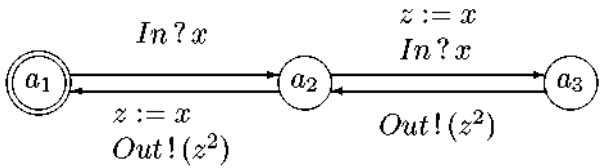


Потоковый граф процесса *Square_Spec'* имеет вид



Графовое представление процесса *Square_Spec'* получается путём
- применения операций параллельной композиции, ограничения и переименования к процессам *Buf* и *Square_Spec,* и
- выполнения редукции получившегося процесса.

Редуцированный процесс *Square_Sped* имеет вид

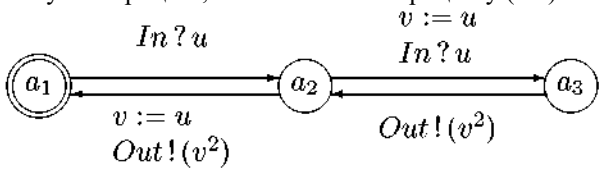


(8.9)

Соответствие процесса *Square* спецификации *Square_Sped* можно понимать как истинность соотношения

$$(8.8) \approx (8.9)$$

Доказать наблюдаемую эквивалентность процессов (8.8) и (8.9) можно, например, при помощи теоремы 34. Для того, чтобы её можно было применить, переименуем переменные в процессе (8.9) (чтобы множества переменных анализируемых процессов не пересекались). Мы получим процесс, эквивалентный процессу (8.9):



(8.10)

Для доказательства соотношения (8.8) ≈ (8.10) при помощи теоремы 34 мы определим функцию

$$\mu : \{A_1, A_2, A_3\} \times \{a_1, a_2, a_3\} \to Fm$$

следующим образом:

- $\mu(A_i, a_j) \stackrel{\text{def}}{=} \bot$, если $i \neq j$
- $\mu(A_1, a_1) \stackrel{\text{def}}{=} \top$
- $\mu(A_2, a_2) \stackrel{\text{def}}{=} (x = y = z = u)$
- $\mu(A_3, a_3) \stackrel{\text{def}}{=} \left\{ \begin{array}{l} x = y = v \\ z = u \end{array} \right\}$

Детальная проверка истинности соответствующих диаграмм остаётся читателю в качестве несложного упражнения.

8.3 Сети Петри

Одной из математических моделей, предназначенных для **описания поведения распределённых систем**, являются **сети Петри.**
Сеть Петри - это граф, **множество вершин которого делится на два класса: позиции *(V)* и переходы (T).** Каждое ребро соединяет позицию с переходом. Каждому переходу $t \in T$ соответствует два множества позиций:

- $in(t) \stackrel{\text{def}}{=} \{v \in V \mid \text{существует ребро из } v \text{ в } t\}$
- $out(t) \stackrel{\text{def}}{=} \{v \in V \mid \text{существует ребро из } t \text{ в } v\}$

Разметка сети Петри представляет собой отображение ξ вида

$$\xi : V \to \{0, 1, 2, \ldots\}$$

Функционирование сети Петри заключается в преобразовании её разметки, которое происходит в результате срабатывания переходов. Разметка ξ_0 в момент времени 0 предполагается заданной. Если в текущий момент времени $i \geq 0$ сеть имела разметку ξ_i, то сработать в этот момент может любой из переходов $t \in T$, который удовлетворяет условию

$$\forall\, v \in in(t) \quad \xi_i(v) > 0$$

Если в момент времени *i* сработал переход *t*, то разметка ξ_{i+1} в момент времени *i* + 1 определяется следующим образом:

$$\forall\, v \in in(t) \quad \xi_{i+1}(v) := \xi(v) - 1$$
$$\forall\, v \in out(t) \quad \xi_{i+1}(v) := \xi(v) + 1$$
$$\forall\, v \in V \setminus (in(t) \cup out(t)) \quad \xi_{i+1}(v) := \xi(v)$$

Каждой сети Петри $\mathcal{N}$ можно сопоставить процесс $P_{\mathcal{N}}$, который моделирует работу этой сети. Компоненты процесса $P_{\mathcal{N}}$ имеют следующий вид.

- $X_{P_{\mathcal{N}}} \stackrel{\text{def}}{=} \{x_v \mid v \in V\}$,
- $I_{P_{\mathcal{N}}} \stackrel{\text{def}}{=} \bigwedge\limits_{v \in V} (x_v = \xi_0(v))$,
- $S_{P_{\mathcal{N}}} \stackrel{\text{def}}{=} \{s^0\}$

• Пусть *t* - произвольный переход сети $\mathcal{N}$, и множества *in(t)* и *out(t)* имеют вид $\{u_1, \ldots, u_n\}$ и $\{v_1, \ldots, v_m\}$ соответственно.
Тогда процесс $P_{\mathcal{N}}$ содержит переход из s^0 в s^0 с меткой

$$\left(\begin{array}{l} (x_{u_1} > 0) \wedge \ldots \wedge (x_{u_n} > 0)\ ? \\ x_{u_1} := x_{u_1} - 1, \ldots, x_{u_n} := x_{u_n} - 1 \\ x_{v_1} := x_{v_1} + 1, \ldots, x_{v_m} := x_{v_m} + 1 \end{array} \right)$$

8.4 Протоколы передачи данных в компьютерных сетях

8.4.1 Понятие протокола

Важным примером процессов с передачей сообщений являются **протоколы передачи данных в компьютерных сетях** (называемые ниже просто **протоколами).**
Каждый протокол можно рассматривать как совокупность нескольких взаимодействующих процессов, в которую входят
• процессы, выполняющие формирование, отправление, приём и обработку сообщений
(такие процессы называются **агентами** протокола, а сообщения, пересылаемые от одного агента другому, называются **кадрами (frame))**

• и процесс, являющийся **моделью поведения среды,** через которую передаются кадры (такую среду обычно называют **каналом связи).** Проходя через среду, кадры могут искажаться или пропадать (например, в результате воздействия радиопомех). Поэтому каждый кадр должен содержать
• не только ту информацию, которую один агент желает передать другому, но и
• средства, позволяющие получателю этого кадра выяснить, был ли этот кадр искажён в процессе передачи.
Ниже мы рассматриваем некоторые методы распознавания искажений в кадрах. Эти методы делятся на два класса:
1. методы, позволяющие
• не только установить факт искажения ,
• но и определить искажённые биты кадра и исправить их
(рассматриваются в параграфе 8.4.2), и
2. методы, позволяющие лишь установить факт искажения кадра (рассматриваются в параграфе 8.4.3).

8.4.2 Методы исправления искажений в кадрах

Методы распознавания искажений в кадрах, позволяющие
• не только установить факт искажения, но и
• определить номера искажённых битов и исправить их
используются в таких ситуациях, когда вероятность того, что каждый передаваемый кадр будет искажён при передаче, является высокой. Например, такая ситуация имеет место в беспроводной связи.
Каждый кадр представляет собой битовую строку. Искажение кадра заключается в инвертировании некоторых битов этого кадра.
Если известно максимальное количество битов кадра, которые могут быть инвертированы, то для распознавания инвертированных битов и их исправления могут быть использованы методы **кодирования с исправлением ошибок,** которые составляют одно из направлений **теории кодирования.**
В этом параграфе мы рассматриваем метод кодирования с исправлением ошибок в простейшем случае, когда в кадре может быть инвертировано не более одного бита. **Данный метод называется кодом Хэмминга для исправления одной ошибки (существуют коды Хэмминга для исправления произвольного количества ошибок).**
Идея данного метода заключается в том, что биты кадра делятся на два класса:

• информационные биты (содержащие ту информацию, которую отправитель хочет передать получателю), и
• контрольные биты (значения которых вычисляются по значениям информационных битов).

Пусть кадр *f* имеет вид (b_1,..., b_n), и
• количество информационных битов в нём равно *k*,
• а количество контрольных битов -*r*,
т.е. $n = k + r$.

Поскольку отправитель может размещать свою информацию в *k* информационных битах, то мы можем считать, что информация, которую отправитель передаёт получателю в кадре *f*, представляет собой битовую строку *M* размера *k*. Кадр, получаемый из строки *M* дополнением её контрольными битами, мы будем обозначать знакосочетанием *φ(M)*.

Для каждого кадра *f* обозначим знакосочетанием <*f*> совокупность всех кадров, получаемых из / инвертированием не более одного бита. Очевидно, что количество элементов в <*f*> равно *n+1*.

Предположение о том, что в кадре *φ(M)* при передаче может произойти искажение не более чем в одном бите, можно переформулировать следующим образом: получатель может получить вместо *φ(M)* любой кадр из множества <*φ(M)*>.

Нетрудно видеть, что для того, чтобы получатель для каждого $M \in \{0,1\}^k$ мог по произвольному кадру из <*φ(M)*> однозначно восстановить *M,* необходимо и достаточно выполнение условия

$$\forall M_1, M_2 \in \{0,1\}^k \quad M_1 \neq M_2 \;\Rightarrow\; \langle\varphi(M_1)\rangle \cap \langle\varphi(M_2)\rangle = \emptyset \quad (8.11)$$

т.е. совокупность подмножеств множества $\{0,1\}^n$ вида

$$\{\langle\varphi(M)\rangle \mid M \in \{0,1\}^k\}$$

состоит из непересекающихся подмножеств. Поскольку
• количество таких подмножеств = 2^k, и
• каждое из этих подмножеств состоит из $n + 1$ элемента
то для выполнения условия (8.11) должно быть верно неравенство

$$(n+1) \cdot 2^k \leq 2^n$$

которое можно переписать в виде

$$(k + r + 1) \leq 2^r \quad (8.12)$$

Нетрудно доказать, что при каждом фиксированном $k > 0$ неравенство (8.12) (где *r* предполагается положительным) эквивалентно неравенству

$$r_0 \leq r$$

где r_0 зависит от *k,* и представляет собой нижнюю оценку количества контрольных битов.

Число r_0 нетрудно вычислить, когда k имеет вид

$$k = 2^m - m - 1, \qquad \text{где } m \geq 1 \quad (8.13)$$

в этом случае (8.12) можно переписать в виде

$$2^m - m \leq 2^r - r \quad (8.14)$$

что эквивалентно $m \leq r$ (т.к. функция $2^x - x$ монотонна при $x \geq 1$).
Таким образом, в данном случае нижняя оценка количества контрольных битов r_0 равна m.
Оказывается, что данная оценка достижима: существует метод кодирования с исправлением одной ошибки, при котором количество r контрольных битов совпадает с минимально возможным значением r_0 .
Ниже мы излагаем метод такого кодирования.
Если k имеет вид (8.13), и $r = r_0 = m$, то $n = 2^m - 1$, т.е. номера битов кадра $(b_1,..., b_n)$ можно представлять кортежами из $\{0,1\}^m$: каждый номер $i \in \{1, \ldots, n\}$ представляется двоичной записью числа i, дополненной слева нулями до кортежа длины m.
Мы будем полагать, что номера контрольных битов имеют кортежную запись

$$(0 \ldots 1 \ldots 0)$$

(одна единица, и остальные нули).
Значения контрольных битов мы будем вычислять следующим образом: для каждого $j = 1,... ,m$ значение контрольного бита с кортежным номером (0 ... 1... 0) (единица в j-й позиции) равно сумме по модулю 2 значений информационных битов, кортежная запись номеров которых содержит 1 в j-й позиции.
При получении кадра $(b_1,..., b_n)$ мы проверяем m равенств

$$\sum_{i_j=1} b_{i_1 \ldots i_m} = 0 \qquad (j = 1, \ldots, m) \quad (8.15)$$

(где сумма рассматривается по модулю 2).
Возможны следующие случаи.

- При передаче этого кадра не было искажений. В этом случае все равенства (8.15) будут верны.
- При передаче этого кадра произошло инвертирование контрольного бита с кортежным номером (0 ... 1... 0) (единица в j-й позиции). Нетрудно видеть, что в этом случае среди равенств (8.15) будет неверно только равенство номер j.
- При передаче этого кадра произошло инвертирование информационного бита с кортежным номером, содержащим единицы в позициях $j_1 ..., j_l$.

Нетрудно видеть, что в этом случае среди равенств (8.15) будут неверны только равенства с номерами $j_1, ..., j_l$.
Таким образом, во всех случаях мы можем
- определить, было ли искажение кадра при передаче, и
- если искажение было - то вычислить номер искажённого бита.

8.4.3 Методы обнаружения искажений в кадрах

Другой класс методов анализа искажений в кадрах связан с установлением самого факта искажения.
Задача определения номеров искажённых битов имеет высокую сложность, и поэтому, если вероятность искажения кадров при передаче невысока (что имеет место при использовании медных или оптоволоконных каналов связи), то более эффективным с точки зрения сложности является повторная посылка искажённых кадров: если получатель кадра обнаруживает искажение в этом кадре, он просит отправителя послать этот кадр ещё раз.
Для сравнения сложности задач исправления искажений и обнаружения искажений рассмотрим следующий пример. Пусть известно, что в кадре может быть искажено не более одного бита. Если размер кадра $n = 1000$, то
- для *исправления* такого искажения нужно 10 контрольных битов,
- а для *обнаружения* такого искажения нужен лишь 1 контрольный бит, значение которого полагается равным чётности количества единиц в информационных битах.

Один из методов кодирования с целью обнаружения искажений заключается в следующем:
- кадр делится на k частей, и
- в каждой части назначается один контрольный бит, значение которого полагается равным чётности количества единиц в остальных битах этой части.

Если при передаче этого кадра биты искажаются равновероятно и независимо, то для каждой такой части кадра вероятность того, что
- эта часть кадра искажена, и
- тем не менее, её чётность верна (т.е. мы считаем её неискажённой) меньше 1/2, поэтому вероятность необнаружения искажения меньше 2^{-k}.

Другим методом кодирования с целью обнаружения искажений является **полиномиальный код** (Cyclic Redundancy Check, CRC). Данный метод основан на рассмотрении битовых строк как многочленов над полем $\mathbf{Z}_2 = \{0, 1\}$:битовая строка вида

$$(b_k, b_{k-1}, \ldots, b_1, b_0)$$

рассматривается как многочлен

$$b_k \cdot x^k + b_{k-1} \cdot x^{k-1} + \ldots + b_1 \cdot x + b_0$$

Пусть необходимо передавать кадры размера *m + 1.* Каждый такой кадр рассматривается как многочлен *M(x)* степени ≤ *m.* Для кодирования этих кадров выбираются

- число *r* < *m,* и
- многочлен *G(x)* степени *r*, имеющий вид

$$x^r + \ldots + 1$$

Многочлен *G(x)* называется **образующим многочленом.**

Для каждого кадра *M(x)* его код *T(x)* вычисляется следующим образом. Многочлен $x^r \cdot M\{x)$ делится с остатком на *G(x):*

$$x^r \cdot M(x) = G(x) \cdot Q(x) + R(x)$$

где *R(x)* - остаток, т.е. многочлен степени < *r.*

Кодом кадра *M(x)* является многочлен

$$T(x) \stackrel{\text{def}}{=} G(x) \cdot Q(x)$$

Нетрудно видеть, что размер *T(x)* больше размера *M(x)* на *r.*

Обнаружение искажений при передаче кадра *T(x)* производится путём деления полученного кадра *T'(x)* на *G(x):* считается, что кадр *T(x)* был передан без искажений (т.е. полученный кадр *T'(x)* совпадает с *T{x)),* если *T'(x)* делится без остатка на *G(x).*

Если кадр *T(x)* был передан без искажений, то исходный кадр *M(x)* можно восстановить путём представления *T(x)* в виде суммы

$$T(x) = x^r \cdot M(x) + R(x)$$

где *R(x)* состоит из всех мономов в *T(x)* степени < *r.*

Если кадр *T(x)* был передан с искажениями, то связь между *T(x)* и *T'(x)* можно представить в виде соотношения

$$T'(x) = T(x) + E(x)$$

где многочлен *E(x)* называется **многочленом искажений,** и соответствует битовой строке, каждая компонента которой равна

- 1, если соответствующий бит кадра *T(x)* был искажён, и
- 0, иначе.

Таким образом,

- если *T(x)* был искажён в одном бите, то $E(x) = x^i$
- если *T(x)* был искажён в двух битах, то $E\{x) = x^i + x^j$,
- и т.д.

Из определений *T'(x)* и *E(x)* следует, что *T'(x)* делится на *G(x)* без остатка тогда и только тогда, когда *E(x)* делится на *G(x)* без остатка. Поэтому искажение, соответствующее многочлену *E(x),* обнаруживается в том и только том случае, когда остаток от деления *E(x)* на *G(x)* не равен нулю.

Рассмотрим подробнее вопрос о том, какие виды искажений могут быть обнаружены при помощи данного метода.

1. Однобитные искажения обнаруживаются всегда, т.к. многочлен $E(x) = x^i$ не делится на *G(x)* без остатка.

2. Двухбитное искажение может не обнаруживаться в том случае, когда соответствующий многочлен

$$E(x) = x^i + x^j = x^j \cdot (x^{i-j} + 1)$$

делится на *G* без остатка (мы предполагаем, что *i > j):*

$$x^j \cdot (x^{i-j} + 1) = G(x) \cdot Q(x) \quad (8.16)$$

Из теоремы о единственности разложения на множители многочленов над полем следует, что соотношение (8.16) влечёт соотношение

$$x^{i-j} + 1 = G(x) \cdot Q_1(x) \quad (8.17)$$

Имеет место следующий факт: если

$$G(x) = x^{15} + x^{14} + 1 \quad (8.18)$$

то для каждого $k = 1,..., 32768$ многочлен $x^k + 1$ не делится на *G(x)* без остатка. Поэтому образующий многочлен (8.18) позволяет обнаружить двухбитные искажения в кадрах длины ≤ 32768.

3. Представим многочлен искажений *E(x)* в виде

$$E(x) = x^j \cdot (x^{k-1} + \ldots + 1) \quad (8.19)$$

Число *k* в этой записи называют **размером пакета ошибок.** Очевидно, что *k* равно размеру подстроки строки искажений (т.е. той строки, которой соответствует многочлен *E(x)),* ограниченной левым и правым единичными битами.

Обозначим знакосочетанием $E_1(x)$ второй множитель в (8.19).

Из теоремы о единственности разложения на множители многочленов над полем следует, что искажение, соответствующее многочлену (8.19), не обнаруживается в том и только том случае, когда *E\{x)* делится без остатка на *G(x).* Это невозможно, если степень $E_1(x)$ меньше степени *G(x),* т.е. если $k — 1 < r$ (или $k \leq r$).

Таким образом, можно обнаружить такие искажения, размер пакета ошибок в которых $\leq r$.

4. Если размер пакета ошибок $k = r + 1$, то многочлен $E_1(x)$ (см. предыдущий пункт) делится без остатка на $G(x)$ в том и только том случае, когда $E_1(x) = G(x)$.
Вероятность такого совпадения равна $2^{-(r-1)}$
Таким образом, если размер пакета ошибок равен $r + 1$, то вероятность необнаружения такого искажения равна $2^{-(r-1)}$.
5. Можно доказать, что если размер пакета ошибок $k > r + 1$, то вероятность необнаружения такого искажения $< 2^{-r}$.
6. Если
• искажено нечётное количество битов, т.е. $E(x)$ состоит из нечётного количества мономов, и
• $G = (x + 1) \cdot G_1$
то такое искажение обнаруживается, т.к. если бы было верно равенство

$$E(x) = G(x) \cdot Q(x)$$

для некоторого многочлена $Q(x)$, то, в частности было бы верно равенство

$$E(1) = G(1) \cdot Q(1) \quad (8.20)$$

чего не может быть, так как левая часть в (8.20) равна 1, а правая - 0.
В заключение отметим, что в стандарте IEEE 802 используется образующий многочлен $G(x)$ вида

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + \\ + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

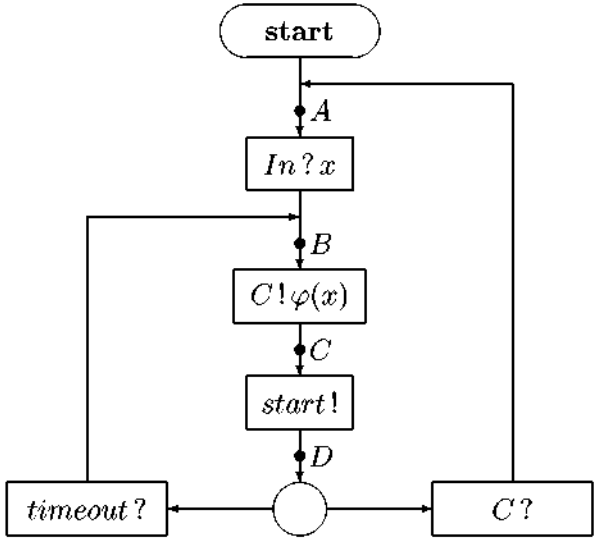
он обнаруживает искажения, в которых размер пакета ошибок ≤32, или искажено нечётное количество битов.

8.4.4 Пример протокола

Рассмотрим простой пример протокола, который состоит из двух агентов: **отправителя** и **получателя.** Задачей протокола является организация доставки кадров от отправителя получателю через ненадёжный канал связи (который может искажать и терять передаваемые кадры).
Работа протокола происходит следующим образом.
1. Отправитель получает сообщения от процесса, не входящего в протокол, который называется **сетевым уровнем (СУ) отправителя.** Эти сообщения называются **пакетами.** Задача отправителя заключается в периодическом выполнении следующих действий:
• получить очередной пакет от своего СУ

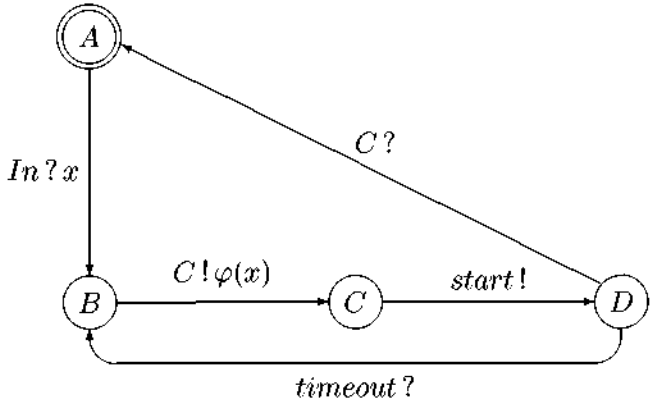
• сформировать из этого пакета кадр
• послать этот кадр в канал связи и включить таймер
• если таймер пришлёт сигнал timeout (который означает, что время ожидания подтверждения посланного кадра закончилось, и, видимо, этот кадр до получателя не дошёл), то послать этот кадр ещё раз
• если придёт подтверждение от получателя, то это означает, что текущий кадр дошёл до него успешно, и можно
— получать следующий пакет от своего СУ,
— формировать из него кадр,
— и т.д.

Блок-схема, представляющая описанное выше поведение, выглядит следующим образом:

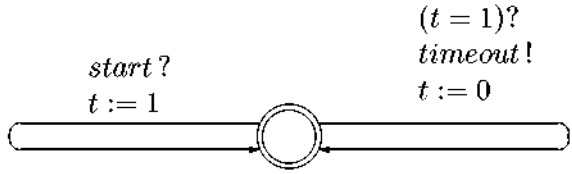


Операторы, входящие в эту блок-схему, имеют следующий смысл.
• *In?x* - получение отправителем очередного пакета от своего СУ, и запись этого пакета в переменную *x*
• *C*! *φ(x)* - отправление в канал связи кадра *φ(x)*
• *start*! - включение таймера
• *timeout* ? - получение от таймера сигнала timeout
• *C* ? - получение из канала связи сигнала подтверждения

Процесс, представляемый этой блок-схемой, обозначается знакосочетанием Sender и имеет следующий вид:

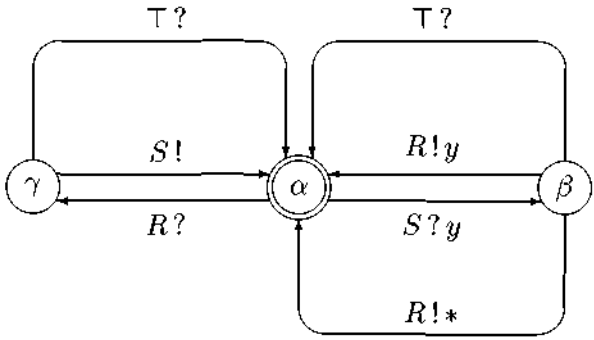


Процесс отправителя является параллельной композицией (с ограничением)
• процесса Sender, и
• процесса Timer, представляющего поведение таймера, и имеющего вид



(8.21)

Начальное условие процесса Timer: $t = 0$.
В этой модели мы не детализируем величину промежутка времени между
— запуском таймера (действие *start?),* и
— посылкой им сигнала *timeout.*
2. **Канал связи** (называемый ниже просто каналом) в каждый момент времени может содержать не более одного кадра или сигнала. Он может выполнять следующие действия:
• получение кадра от отправителя, и
— пересылка этого кадра получателю, или
— пересылка получателю искажённого кадра, или
— потеря кадра
• получение сигнала подтверждения от получателя, и
— пересылка этого сигнала отправителю, или
— потеря сигнала.
Поведение канала представляется следующим процессом:



(8.22)

В этом процессе мы используем следующую абстракцию: символ * означает "искажённый кадр". Мы не уточняем, как именно искажаются кадры в канале. Каждый кадр, поступивший в канал

• либо передаётся из канала в неизменном виде получателю,

• либо пребразуется в абстрактное значение *, и это значение передаётся из канала получателю

• либо пропадает, что выражается переходом процесса (8.22) с меткой $\top?$

3. **Получатель** периодически выполняет следующие действия:

• получение из канала очередного кадра

• проверка наличия искажений в кадре

• если кадр не искажён, то

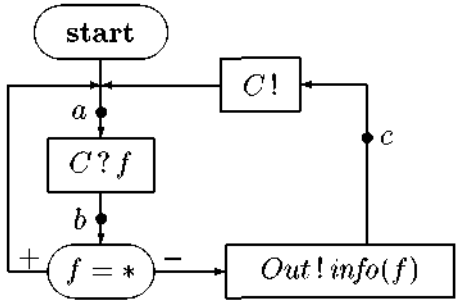
— извлечение из кадра пакета,

— передача этого пакета процессу, называемому **сетевой уровень (СУ) получателя** (этот процесс не входит в протокол)

— посылка отправителю через канал сигнала подтверждения

• если кадр искажён, то получатель его игнорирует (предполагая, что отправитель, не дождавшись подтверждения, пошлёт это кадр ещё раз)

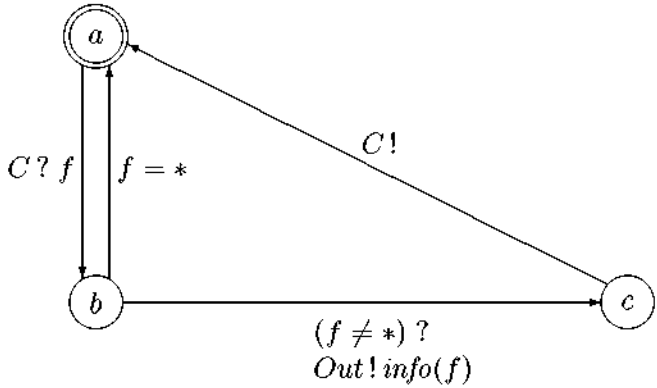
Блок-схема, представляющая описанное выше поведение, выглядит следующим образом:



Операторы, входящие в эту блок-схему, имеют следующий смысл.

- *C* ? *f* - получение из канала очередного кадра, и запись его в переменную *f*
- (*f* = *) - проверка наличия искажения в кадре *f*
- *Out*! *info(f)* - отправление пакета *info(f),* извлечённого из кадра *f,* своему СУ
- *C* ! - посылка сигнала подтверждения

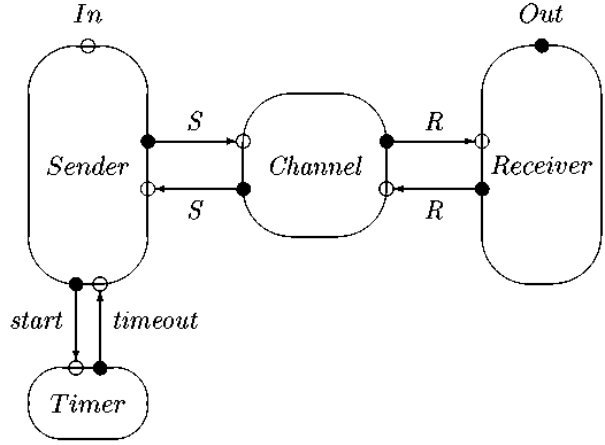
Процесс, представляемый этой блок-схемой, обозначается знакосочетанием Receiver и имеет следующий вид:



Процесс *Protocol,* соответствующий всему протоколу, определяется как параллельная композиция (с ограничением) вышеперечисленных процессов:

$$Protocol \stackrel{\text{def}}{=} \left(\begin{array}{l} Sender\,[S/C]\,| \\ Timer\,| \\ Channel\,| \\ Receiver\,[R/C] \end{array} \right) \setminus \{S, R, start, timeout\} \quad (8.23)$$

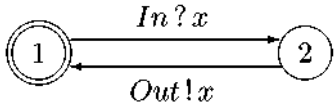
Потоковый граф этого процесса имеет следующий вид:



(8.24)

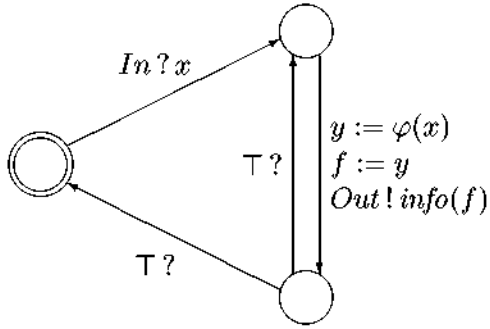
Для того, чтобы можно было анализировать корректность этого протокола, необходимо точно определить спецификацию, которой он должен соответствовать.

Если мы хотим специфицировать только свойства внешних действий, выполняемых этим протоколом (т.е. действий, имеющих вид *In ? v* и *Out !v),* то спецификация может выглядеть следующим образом: поведение данного протокола совпадает с поведением буфера размера 1, т.е. процесс *Protocol* наблюдаемо эквивалентен процессу *Buf,* который имеет вид

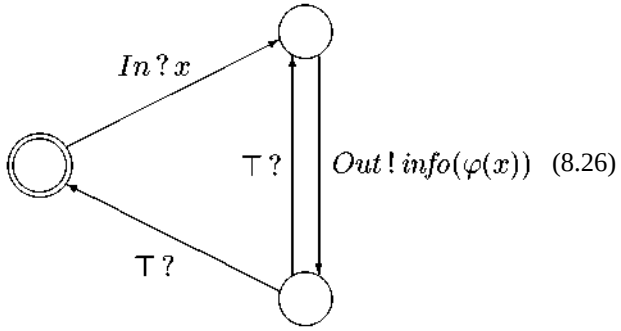


(8.25)

При построении процесса с СО, соответствующего выражению (8.23), и его редукции, получается диаграмма



которая наблюдаемо эквивалентна диаграмме

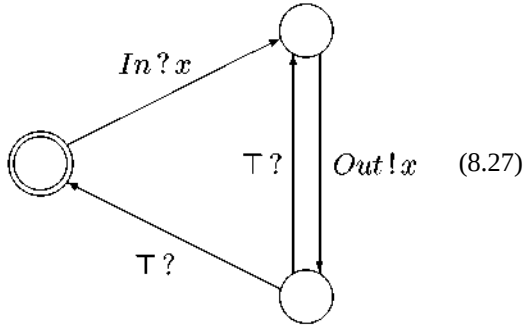


(8.26)

Если мы предположим, что функция *info* извлечения пакетов из кадров является обратной к функции φ формирования кадров, т.е. для каждого пакета x имеет место соотношение

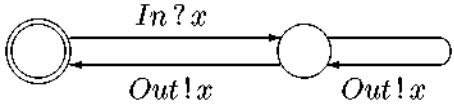
$$info(\varphi(x)) = x$$

то диаграмму (8.26) можно перерисовать следующим образом:



(8.27)

Процесс (8.27) можно редуцировать, в результате чего получается процесс



(8.28)

При сопоставлении процессов (8.28) и (8.25) можно заключить, что данные процессы не могут быть эквивалентными ни в каком приемлемом смысле. Например,

• процесс (8.25) после получения пакета *x* может только
— передать этот пакет СУ получателя, и
— перейти в состояние ожидания следующего пакета
• в то время как процесс (8.28) может после получения пакета *x* передать этот пакет СУ получателя несколько раз.

Такая повторная передача может происходить, например, при следующем варианте работы протокола.

• Первый кадр, посланный отправителем, доходит до получателя успешно.
• Получатель
— пересылает пакет, извлечённый из этого кадра, своему СУ, и
— посылает отправителю через канал подтверждение.
• Это подтверждение пропадает в канале.
• Отправитель, не дождавшись подтверждения, посылает этот кадр ещё раз, и этот кадр снова доходит успешно.
• Получатель воспринимает этот кадр как новый. Он
— пересылает пакет, извлечённый из этого кадра, своему СУ, и
— посылает отправителю через канал подтверждение, которое опять пропадает канале.
• и т.д.

Эта ситуация может возникнуть потому, что в данном протоколе отсутствует механизм, с помощью которого получатель мог бы отличить новый кадр от переданного повторно.

В следующем параграфе мы приводим пример протокола, в котором этот механизм присутствует. Для такого протокола уже можно формально доказать его соответствие спецификации (8.25).

8.4.5 Протокол с чередующимися битами

Протокол с чередующимися битами (называемый в англоязычной литературе словосочетанием **Alternating Bit Protocol,** или, сокращённо, **ABP)** предназначен для решения той же задачи, что и предыдущий протокол: доставка кадров от отправителя получателю

через ненадёжный канал связи (который может искажать и терять передаваемые кадры).

Механизм, с помощью которого получатель может отличить новый кадр от переданного повторно, реализован в данном протоколе следующим образом: среди переменных отправителя и получателя присутствуют булевы переменные *s* и *r* соответственно, значения которых имеют следующий смысл:

- значение переменной *s* равно чётности номера очередного кадра, которого пытается послать отправитель, и
- значение переменной *r* равно чётности номера очередного кадра, которого ожидает получатель.

В начальный момент значения *s* и *r* равны 0 (первый кадр имеет номер 0).

Как и в предыдущем протоколе, в этом протоколе используется абстрактное значение *, обозначающее искажённый кадр.

Работа протокола происходит следующим образом.

1. Отправитель, получив очередной пакет от своего СУ,

- записывает его в переменную *x,*
- формирует кадр, который представляет собой значение некоторой кодирующей функции φ на паре (*x, s),*
- посылает этот кадр в канал,
- запускает таймер
- после чего он ожидает подтверждение посланного кадра.

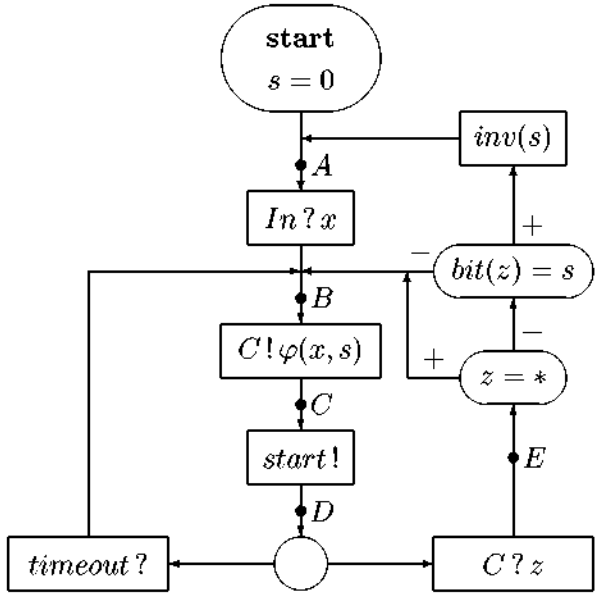
Если у отправителя истекает время ожидания, и он не получает подтверждения от получателя, то он повторно посылает уже посланный кадр.

Если же он получает подтверждение, которое представляет собой неискажённый кадр, содержащий бит, то он анализирует значение этого бита: если оно совпадает с текущим значением *s,* то отправитель

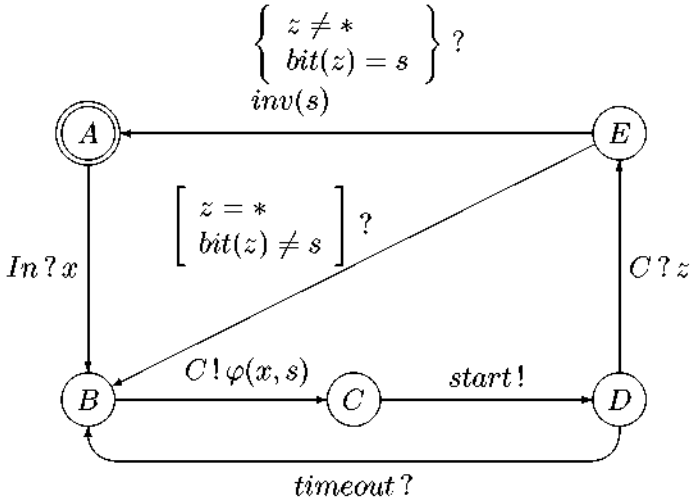
- инвертирует значение переменной *s* (используя для этого функцию $inv(x) = 1 - x),$ и
- начинает новый цикл своей работы.

В противном случае он опять посылает уже посланный кадр.

Блок-схема, представляющая такое поведение, выглядит следующим образом:



Процесс, представляемый этой блок-схемой, обозначается знакосочетанием Sender и имеет следующий вид:
Init = (*s* = 0)

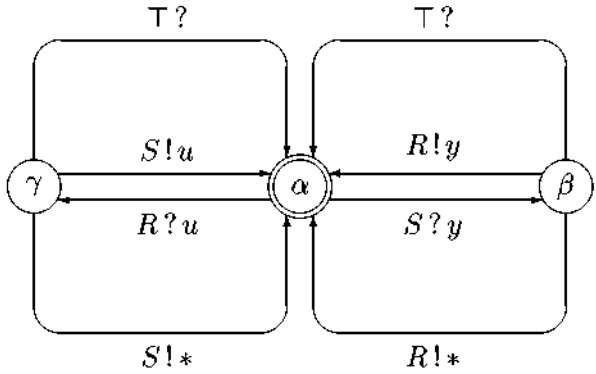


Процесс отправителя является параллельной композицией (с ограничением) процесса Sender, и процесса Timer.

2. **Канал** в каждый момент времени может содержать не более одного кадра. Он может выполнять следующие действия:

• получение кадра от отправителя, и
— пересылка этого кадра получателю, или
— пересылка получателю искажённого кадра, или
— потеря кадра
• получение кадра с подтверждением от получателя, и
— пересылка этого кадра отправителю, или
— пересылка отправителю искажённого кадра, или
— потеря кадра.

Поведение канала представляется следующим процессом:



(8.29)

3. Получатель при получении очередного кадра из канала

• проверяет, не искажён ли этот кадр,
• и если не искажён, то извлекает из него пакет и связанный с ним бит при помощи функций *info* и *bit,* обладающих следующими свойствами:

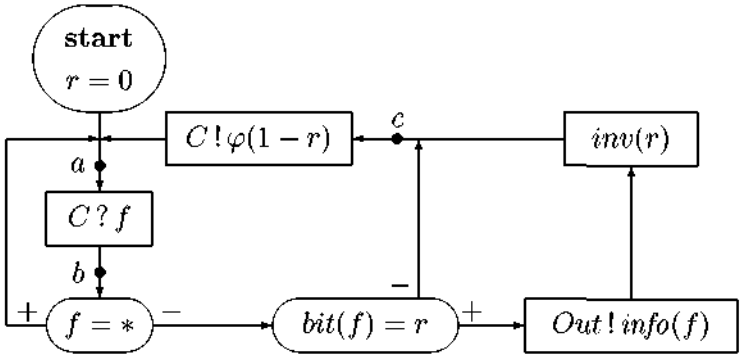
$$info(\varphi(x,b)) = x, \quad bit(\varphi(x,b)) = b$$

Получатель проверяет, совпадает ли значение бита, извлечённого из кадра, с ожидаемым значением, которое содержится в переменной *r*. Если проверка дала положительный результат, то получатель

• передаёт пакет, извлечённый из этого кадра, своему СУ
• инвертирует значение переменной *r*, и
• посылает отправителю через канал подтверждение.

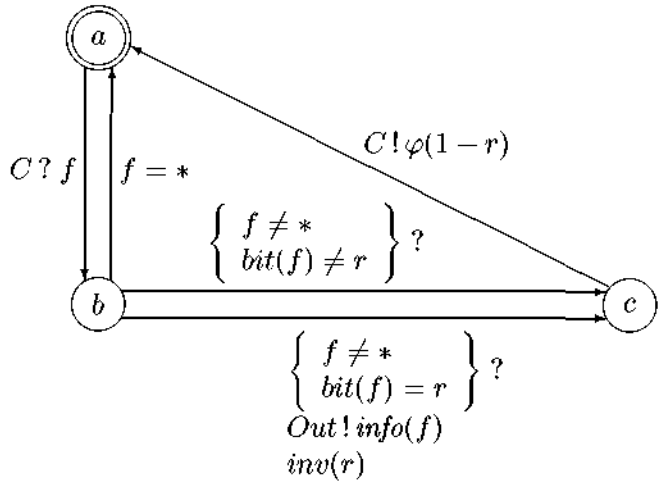
Если проверка дала отрицательный результат, то получатель посылает кадр подтверждения с неверным битом (что заставит отправителя послать свой кадр ещё раз).

Если же кадр искажён, то получатель его игнорирует (предполагая, что отправитель, не дождавшись подтверждения, пошлёт это кадр ещё раз) Блок-схема, представляющая описанное выше поведение, выглядит следующим образом:



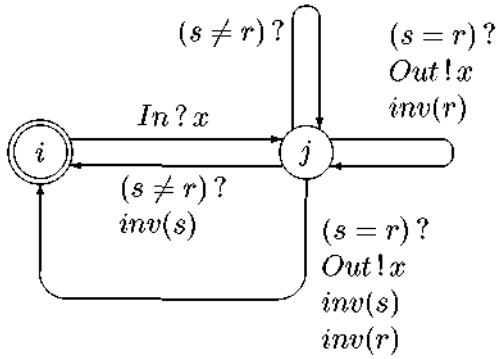
Процесс, представляемый этой блок-схемой, обозначается знакосочетанием Receiver и имеет следующий вид:

$$Init = (r = 0)$$



Процесс *Protocol,* соответствующий всему протоколу, определяется так же, как и в предыдущем пункте, выражением (8.23). Потоковый граф данного процесса имеет вид (8.24).

Спецификация данного протокола тоже имеет такой же вид, т.е. представляет собой процесс (8.25).
При построении процесса с СО, соответствующего данному протоколу, и его последующей редукции, получается следующая диаграмма:



(8.30)

Доказать наблюдаемую эквивалентность процессов (8.25) и (8.30) можно, например, при помощи теоремы 34, определив функцию μ вида

$$\mu : \{1,2\} \times \{i,j\} \to Fm$$

следующим образом:

$$\begin{cases} \mu(1,i) \stackrel{\text{def}}{=} (s = r) \\ \mu(2,i) \stackrel{\text{def}}{=} \bot \\ \mu(1,j) \stackrel{\text{def}}{=} (s \neq r) \\ \mu(2,j) \stackrel{\text{def}}{=} (s = r) \end{cases}$$

8.4.6 Двунаправленная передача

Рассмотренные выше протоколы относятся к классу симплексных протоколов: они реализуют однонаправленную передачу информационных кадров (т.е. кадров, содержащих пакеты) от одного агента к другому.
В большинстве ситуаций пересылки данных между двумя агентами требуется **двунаправленная передача,** т.е. передача информационных кадров в обоих направлениях. В данном случае каждый из агентов выступает как в роли отправителя, так и в роли получателя.
Протоколы, реализующие двунаправленную передачу, называются **дуплексными** протоколами.

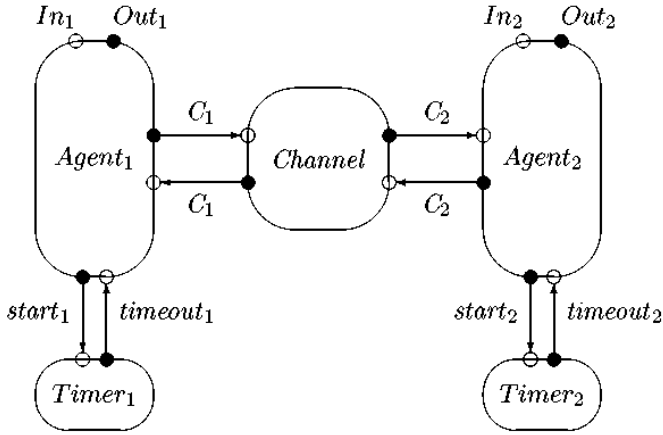
В дуплексных протоколах посылка подтверждений может быть совмещена с посылкой информационных кадров: если агент *B* успешно принял информационный кадр / от агента *A,* он может послать своё подтверждение получения кадра / не отдельным кадром, а в составе своего информационного кадра.
Ниже мы излагаем примеры некоторых дуплексных протоколов.

8.4.7 Дуплексный протокол с чередующимися битами

Простейшим дуплексным протоколом является излагаемый в этом параграфе **дуплексный протокол с чередующимися битами.** Данный протокол является обобщением протокола АВР.
В этом протоколе тоже участвуют два агента, но, в отличие от протокола АВР, поведение каждого из агентов описывается одним и тем же процессом, который совмещает в себе процессы отправителя и получателя из протокола АВР.
Каждый кадр *f*, пересылаемый каким-либо из этих агентов, содержит пакет *x* и два бита: *s* и *r*, где
• *s* имеет тот же смысл, что и в протоколе АВР: это бит, сопоставленный пакету *x,*
• *r* является битом подтверждения последнего полученного неискажённого кадра.
Для формирования кадров используется функция *φ.* Для извлечения пакетов и битов из кадров используются функции *info, seq* и *аск,* обладающие следующими свойствами:

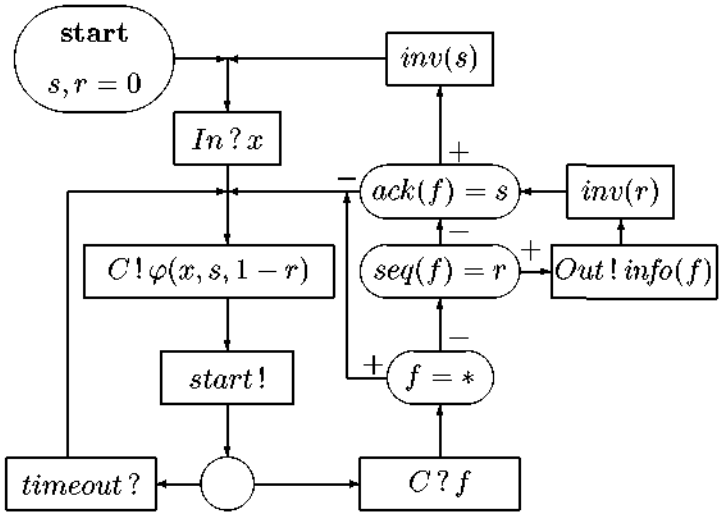
$$\begin{aligned} info(\varphi(x,s,r)) &= x \\ seq(\varphi(x,s,r)) &= s \\ ack(\varphi(x,s,r)) &= r \end{aligned}$$

Также агенты используют функцию *inv* для инвертирования значений битовых переменных.
С каждым из агентов связан свой таймер, поведение которого описывается процессом *Timer,* представленным диаграммой (8.21).
Потоковый граф этого протокола имеет следующий вид:



(8.31)

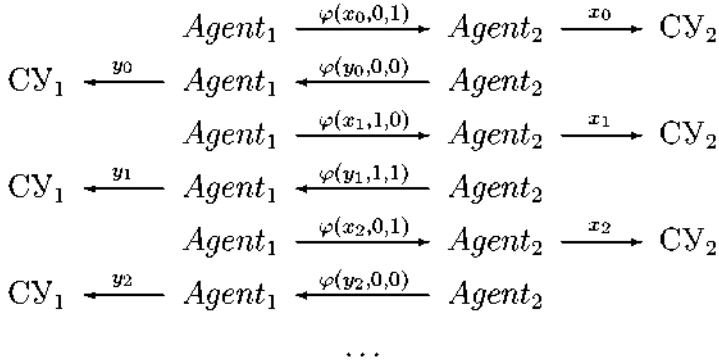
Процесс, описывающий поведение каждого из агентов, представляется в виде следующей блок-схемы:



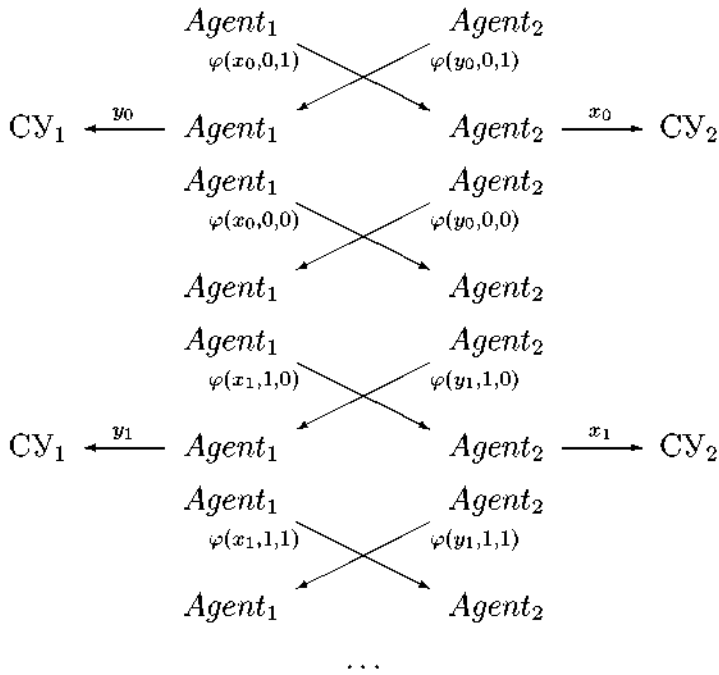
Блок-схема, описывающая поведение конкретного агента, получается из этой блок-схемы приписыванием соответствующего индекса к переменным и именам, входящим в эту блок-схему.
Поведение канала представляется процессом (8.29), к которому применено переименование

$$[C_1/S,\, C_2/R]$$

Таким образом, каждый агент в этом протоколе посылает свой следующий пакет только после получения подтверждения своего предыдущего пакета.
Нормальную работу данного протокола можно изобразить следующей диаграммой:



Однако, если оба агента одновременно вышлют друг другу начальные кадры, то работа протокола будет не вполне нормальной (хотя, тем не менее, передача пакетов в данном случае является корректной), и может быть изображена следующей диаграммой:



Читателю предлагается самостоятельно

- — определить процесс *Spec,* являющийся спецификацией этого протокола, и

— доказать соответствие этого протокола спецификации *Spec,*

- — модифицировать определённый в этом параграфе протокол таким образом, чтобы при любом варианте работы модифицированного протокола не возникало аномальных эффектов, аналогичных приведённому выше, и

— доказать корректность (т.е. соответствие спецификации *Spec)* модифицированного протокола.

8.4.8 Двунаправленная конвейерная передача

Дуплексный протокол с чередующимися битами является практически приемлемым только в том случае, когда длительность пересылки кадров по каналу пренебрежимо мала.

Если же время прохождения кадра по каналу большое, то лучше использовать **конвейерную** передачу, при которой отправитель может

послать несколько кадров подряд, не дожидаясь подтверждений. Ниже мы рассматриваем два протокола двунаправленной конвейерной передачи, называемые **протоколами скользящего окна (ПСО)** (sliding window).
Данные протоколы являются развитием дуплексного протокола с чередующимися битами, изложенного в параграфе 8.4.7.
В этих протоколах
• тоже участвуют два агента, поведение каждого из которых описывается одним и тем же процессом, совмещающим в себе функции отправителя и получателя,
• аналогом бита, связанного с передаваемым кадром, является элемент множества вычетов $\mathbf{Z}_n = \{0, \ldots, n-1\}$, где n - некоторое фиксированное число вида 2^k.
Элемент множества Z_n, связанный с кадром, называется **номером** этого кадра. Отметим, что номер кадра и порядковый номер кадра - это разные понятия: порядковые номера уникальны, а номера циклически повторяются.

8.4.9 Протокол скользящего окна с возвратом

Первый из рассматриваемых нами ПСО называется **ПСО с возвратом** (или **ПСО с повторной передачей).**
Процесс, описывающий поведение агента этого ПСО, содержит среди своих переменных массив *x[n],* в компонентах которого могут содержаться отправленные, но ещё не подтверждённые пакеты. Совокупность компонентов массива *x,* в которых содержатся такие пакеты в текущий момент времени, называется **окном.** С окном связаны три переменные этого процесса:
• *b* - нижняя граница окна,
• *s* - верхняя граница окна, и
• *w* - количество пакетов в окне.
Значения переменных *b, s* и *w* принадлежат множеству Z_n.
В начальный момент времени окно является пустым, и значения переменных *b, s* и *w* равны 0.
Добавление нового пакета к окну происходит путём выполнения следующих операций:
• данный пакет записывается в компоненту *x[s],* и число *s* считается номером этого пакета
• верхняя граница окна *s* увеличивается на 1 по модулю *n,* т.е. новое значение *s* полагается равным

– $s+1$, если $s < n-1$, и

– 0, если $s = n-1$,

• количество пакетов в окне *w* увеличивается на 1. Удаление пакета из окна происходит следующим образом:

• нижняя граница окна *b* увеличивается на 1 по модулю *n,* и

• количество пакетов в окне *w* уменьшается на 1

т.е. удаляется тот пакет, номер которого равен нижней границе окна.

Для упрощения понимания операций работы с окном можно использовать следующую образную аналогию:

• совокупность компонентов массива *x* можно рассматривать как "свёрнутую в кольцо" (после компоненты *x[n* — 1] идёт компонента *x[0])*

• в каждый момент времени окно представляет собой связное подмножество этого кольца,

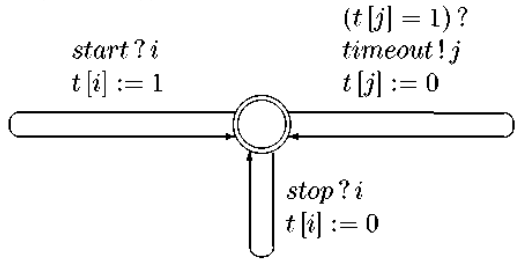
• во время работы процесса окно перемещается по этому кольцу в одном и том же направлении.

Если окно достигает максимального размера *(n* — 1), то агент не принимает новые пакеты от своего СУ до тех пор, пока размер окна не уменьшится. Возможность приёма новых пакетов определяется булевой переменной *enable:* если её значение равно 1, то агент может принимать новые пакеты от своего СУ, а если 0, то не может.

Когда агент получает подтверждение пакета, номер которого равен нижней границе окна, этот пакет удаляется из окна.

С каждой компонентой массива *x* связан таймер, при помощи которого определяется время ожидания подтверждения получения пакета, содержащегося в этой компоненте. Совокупность всех этих таймеров рассматривается как один процесс *Timers,* который имеет массив *t [n]* булевых переменных. Данный процесс определяется следующим образом:

$Init = (t = (0, \ldots, 0))$



(8.32)

Правую стрелку в этой диаграмме следует понимать как сокращённое обозначение для совокупности из *n* переходов, метка каждого из

которых получается заменой в метке этой стрелки символа *j* на любое из значений из множества Z_n.

Отметим, что в данном процессе наряду с операторами запуска таймеров и посылки ими сигнала тайм-аута присутствует также оператор *stop?i,* выполнение которого досрочно завершает работу соответствующего таймера.

ПСО с возвратом имеет следующие особенности.

• Когда заканчивается лимит времени ожидания подтверждения какого-либо пакета, агент начинает передавать повторно все пакеты из своего окна.

• Когда поступает подтверждение получения какого-либо пакета, все предыдущие пакеты в окне тоже считаются подтвержденными (даже если их подтверждения не дошли).

Каждый кадр *f*, пересылаемый каким-либо из агентов этого протокола, содержит пакет *x* и два целых числа: *s* и *r*, где

• *s* - номер, сопоставленный пакету *x* (который по определению равен номеру кадра *f*),

• *r* - номер последнего полученного неискажённого кадра.

Для формирования кадров используется функция φ. Для извлечения пакетов и номеров из кадров используются функции *info, seq* и *ack,* обладающие следующими свойствами:

$$\begin{aligned} info(\varphi(x,s,r)) &= x \\ seq(\varphi(x,s,r)) &= s \\ ack(\varphi(x,s,r)) &= r \end{aligned}$$

Описание процесса, представляющего поведение агента данного ПСО, мы даём в блок-схемном виде, по которому несложно построить блок-схему этого процесса.

В данном описании мы используем следующие обозначения.

• Символы

$$\underset{n}{+} \quad \textbf{и} \quad \underset{n}{-}$$

обозначают сложение и вычитание по модулю *n.*

• Символ *r* обозначает переменную с множеством значений Z_n. Значение переменной *r* равно номеру ожидаемого кадра.

Агент посылает своему СУ пакет, извлечённый только из такого кадра *f*, у которого номер *seq(f)* совпадает со значением этой переменной *r*. Пакеты из кадров с другими значениями *seq(f)* игнорируются, у таких кадров учитывается лишь компонента *ack(f).*

• Знакосочетание *send* является сокращённым обозначением следующей группы операторов:

$$send = \left\{ \begin{array}{l} C\,!\,\varphi(x[s], s, r-1) \\ start\,!\,s \\ s := s +_n 1 \end{array} \right\}$$

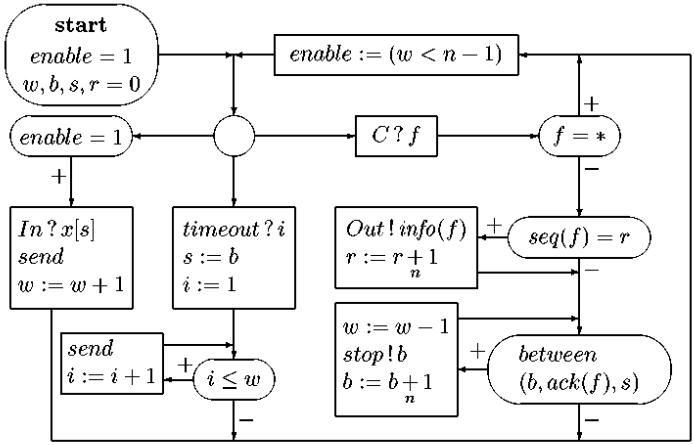
• Знакосочетание *between(a, b,* c) является сокращённым обозначением формулы

$$\Big(a \le b < c \Big) \vee \Big(c < a \le b \Big) \vee \Big(b < c < a \Big) \quad (8.33)$$

• Знакосочетание *enable := (w < n—1)* является сокращённой записью оператора

$$\textbf{if } (w < n-1) \textbf{ then } enable := 1 \textbf{ else } enable := 0$$

Процесс, представляющий поведение агента данного ПСО, имеет следующий вид:



Читателю предлагается самостоятельно

• определить процесс "канал" для этого протокола

(канал может содержать несколько кадров, которые могут не только искажаться и пропадать, но ещё и переупорядочиваться)

• определить спецификацию *Spec* этого протокола

• доказать соответствие этого протокола спецификации *Spec*

• исследовать этот протокол на наличие аномальных эффектов, аналогичных тому эффекту, который был изложен в параграфе 8.4.7 (если аномальные эффекты присутствуют, то модифицировать этот протокол так, чтобы таких эффектов не было, и доказать корректность модифицированного протокола)

В заключение отметим, что данный ПСО неэффективен при большом количестве ошибок при передаче кадров.

8.4.10 Протокол скользящего окна с выборочным повтором

Второй ПСО отличается от предыдущего тем, что у агента этого ПСО имеется не одно, а два окна.
1. Первое окно имеет те же функции, которые имеет окно первого ПСО (данное окно называется **посылающим окном).**
Максимальный размер посылающего окна равен $m \stackrel{\text{def}}{=} n/2,$ где число *n* имеет тот же статус, который описан в параграфе 8.4.8 (в частности, номера кадров являются элементами Z_n).
2. Второе окно (называемое **принимающим окном)** предназначено для размещения пакетов, поступивших от другого агента, которые пока не могут быть переданы СУ, потому что ещё не пришли некоторые пакеты с меньшими номерами.
(агент-получатель должен передавать принятые пакеты своему СУ в том же самом порядке, в котором они поступили к агенту-отправителю от его СУ)
Принимающее окно имеет фиксированный размер *m = n/2.*
Каждый кадр *f*, пересылаемый каким-либо из агентов этого протокола, содержит 4 компоненты:
1. *k* - вид кадра,
данная компонента может принимать одно из трёх значений:
• *data* (информационный кадр)
• *аск* (кадр, содержащий лишь подтверждение)
• *пак* (кадр, содержащий запрос на повторную передачу)
2. *x* - пакет
3. *s* - номер этого кадра
4. *r* - номер последнего полученного неискажённого кадра.
Если значение первой компоненты кадра равно *аск* или *пак,* то вторая и третья компоненты этого кадра имеют фиктивный характер.
Для формирования кадров используется функция *φ*.
Для извлечения компонентов из кадров используются функции *kind, info, seq* и *аск,* обладающие следующими свойствами:

$$\begin{aligned} kind(\varphi(k,x,s,r)) &= k \\ info(\varphi(k,x,s,r)) &= x \\ seq(\varphi(k,x,s,r)) &= s \\ ack(\varphi(k,x,s,r)) &= r \end{aligned}$$

Процесс, описывающий поведение агента данного ПСО, имеет следующие переменные.

1. Массивы $x[m]$ и $y[m]$, предназначенные для размещения посылающего окна и принимающего окна соответственно.
2. Переменные *enable, b, s,w,* имеющие
- те же множества значений, и
- тот же смысл

которые они имеют в предыдущем протоколе.

3. Переменные, связанные с принимающим окном:
- *r* - нижняя граница принимающего окна
- *u* - верхняя граница принимающего окна

значения переменных *r* и *u* принадлежат множеству Z_n.

Если в принимающем окне присутствует пакет, номер которого равным нижней границе принимающего окна (*r*), то агент
- передаёт этот пакет своему СУ, и
- увеличивает на 1 (по модулю *n)* значения *r* и *u.*

4. Булевский массив *arrived[m],* компоненты которого имеют следующий смысл: *arrived[i]* = 1 тогда и только тогда, когда в *i*-й компоненте принимающего окна содержится пакет, пока ещё не переданный СУ.

5. Булева переменная *по пак,* используемая в следующих целях.

Если агент получает
- искажённый кадр, или
- кадр, номер которого отличен от нижней границы принимающего окна (*r*)

то он посылает своему коллеге запрос на повторную передачу кадра, номер которого равен *r*.

Данный запрос называется NAK (Negative Acknowledgement).

Булева переменная *по_пак* используется для того, чтобы избежать нескольких запросов на повторную передачу одного и того же кадра: эта переменная имеет значение 1, если NAK для кадра с номером *r* еще не был послан.

Когда агент получает неискажённый кадр *f* вида *data,* он выполняет следующие действия.

• Если номер *seq(f)* попадает в прнинимающее окно, т.е. истинна формула

$$between(r, seq(f), u)$$

где предикатный символ *between* имеет тот же смысл, что и в предыдущем протоколе (см. (8.33)), то агент — извлекает из этого кадра пакет, и

— помещает этот пакет в своё принимающее окно.

• Если условие из предыдущего пункта не выполнено (т.е. номер *seq(f)* кадра *f* не попал в принимающее окно), то

— пакет в таком кадре игнорируется, и

— у такого кадра учитывается лишь компонента *ack(f).*

В данном протоколе участвуют следующие таймеры.

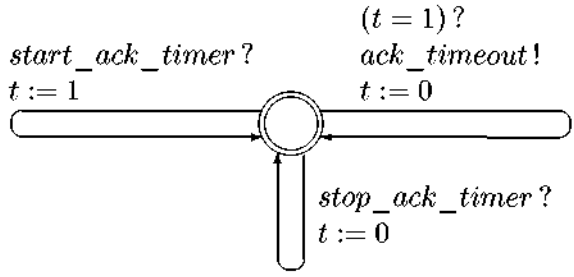
1. Массив из *m* таймеров, поведение которых представляется процессом *Timers* (см. (8.32), с заменой п на га), каждый таймер из этого массива предназначен для оповещения агента о том, что

• время ожидания подтверждения пакета с соответствующим номером из посылающего окна закончилось, и

• необходимо послать кадр с этим пакетом ещё раз

2. Дополнительный таймер, поведение которого представляется следующим процессом:

$$Init = (t = 0)$$



Этот таймер используется со следующей целью.

Посылка агентом подтверждений кадров, поступивших от другого агента, может производиться двумя способами:

(a) подтверждение посылается в составе информационного кадра (т.е. кадра вида *data),* или

(b) подтверждение посылается отдельным кадром вида *аск.*

Когда агенту необходимо послать такое подтверждение а, он

• включает вспомогательный таймер (т.е. выполняет действие *start_ack_timer \),*

• если до сигнала тайм-аута от вспомогательного таймера агент получил новый пакет от своего СУ, он
— формирует кадр вида *data* с этим пакетом,
— включает в этот кадр подтверждение *a* как компоненту *аск* этого кадра, и
— посылает этот кадр своему коллеге
• если после истечения работы вспомогательного таймера (т.е. после получения от него сигнала *ack_timeout)* агент так и не получил новый пакет от своего СУ, он посылает подтверждение *a* отдельным кадром вида
аск.

Описание процесса, представляющего поведение агента данного ПСО, мы даём в блок-схемном виде, по которому несложно построить блок-схему этого процесса.

В данном описании мы используем следующие обозначения и соглашения.

1. Если *i* - целое число, то знакосочетание *i%m* обозначает остаток от деления *i на, m.*
2. Если
• *mass* - имя массива из *m* компонентов (т.е. *x, y, arrived,* и т.д.), и
• *r* - целое число,
то знакосочетание *mass[i]* следует понимать как *mass[i%m].*
3. Знакосочетание вида *send(kind,i)* является сокращённым обозначением следующей группы операторов:

$$send(kind,i) = \left\{ \begin{array}{l} C\,!\,\varphi(kind, x[i], i, r-1) \\ \mathbf{if}\ \ (kind = nak)\ \ \mathbf{then}\ \ no\_nak := 0 \\ \mathbf{if}\ \ (kind = data)\ \ \mathbf{then}\ \ start\,!\,(i\%m) \\ stop\_ack\_timer\,! \end{array} \right\}$$

4. Знакосочетания

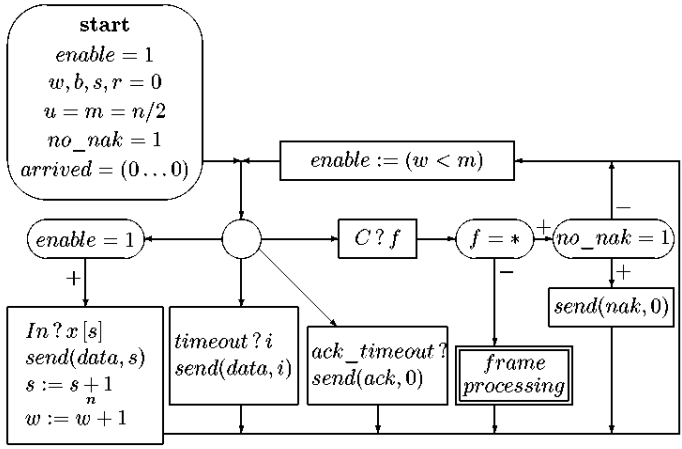
$$between(a,b,c) \quad \text{и} \quad enable := (w < m)$$

имеют тот же смысл, что и в описании предыдущего протокола.
5. Если в каком-либо овале присутствуют не одна, а несколько формул, то мы предполагаем, что эти формулы связаны символом конъюнкции.
6. В целях экономии места некоторые выражения вида

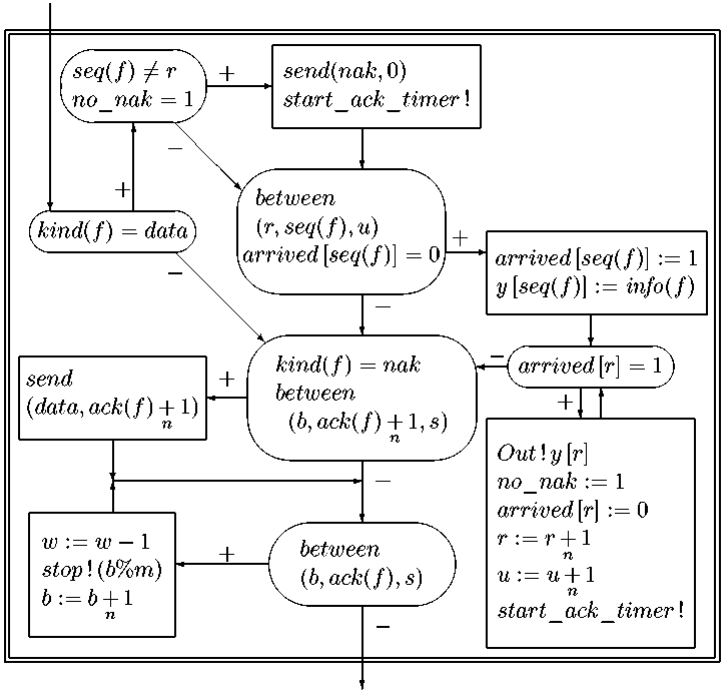
$$f(e_1, \ldots, e_n)$$

написаны в две строки (в первой строке - *f*, а во второй - список (e_1,..., e_n)).

Процесс, представляющий поведение агента данного ПСО, имеет следующий вид:



Фрагмент *frame processing* в этой диаграмме выглядит следующим образом.



Читателю предлагается самостоятельно исследовать для данного протокола все вопросы, которые были перечислены в конце предыдущего параграфа.

8.5 Криптографические протоколы

Важным примером процессов с передачей сообщений являются **криптографические протоколы.**

8.5.1 Понятие криптографического протокола

Криптографический протокол (КП) - это распределённый алгоритм (т.е. совокупность нескольких взаимодействующих процессов), предназначенный для безопасной передачи, обработки и хранения информации или материальных объектов в небезопасной среде.
В работе КП принимают участие несколько **агентов,** которыми могут быть люди, вычислительные процессы, компьютеры, сети связи, базы данных, банковские карточки, банкоматы, и т.д.

Каждый из агентов КП функционирует в соответствии с некоторым последовательным алгоритмом.
Действия, исполняемые агентами, могут иметь следующий вид.
• **Посылка сообщения** другому агенту или группе агентов.
В качестве сообщения в КП может выступать как **информационный, так и материальный объект. Например это может быть набор банкнот или товар.**
• **Приём сообщения** от другого агента.
• **Внутреннее действие,** которое заключается **в преобразовании агентом имеющихся у него информационных и материальных объектов.**
Главное отличие КП от остальных распределённых алгоритмов заключается в том, что
• при функционировании КП допускается высокая вероятность некоторых угроз безопасности сообщений, передаваемых агентами этого КП, и
• КП должен содержать механизмы противодействия этим угрозам.
Под **угрозой безопасности** сообщения понимается возможность выполнения с этим сообщением какой-либо из следующих операций.
1. Нарушение **конфиденциальности** сообщения, т.е., например, перехват этого сообщения (т.е. получение его копии), и
• полное или частичное ознакомление с содержанием этого сообщения субъектами, не имеющими для этого соответствующих полномочий, или
• извлечение из сообщения новой информации, например,
— извлечение из шифртекстов фрагмента криптографического ключа, используемого при создании этих шифртекстов, или
— извлечение из сообщений о покупках какого-либо лица предположений о его финансовом положении или о его предпочтениях.
2. Нарушение **целостности** сообщения, т.е., например, искажение передаваемого текста в процессе передачи.
3. Кража сообщения (например, кража денег или товара в процессе их доставки от одного агента другому агенту).
4. Подмена сообщения, т.е.
• изъятие сообщения, которое агент *A* послал агенту *B,* и
• посылка агенту *B* другого сообщения вместо изъятого.
Ниже под угрозами безопасности мы будем понимать угрозы безопасности сообщений, передаваемых между агентами какого-либо КП.
Угрозы безопасности могут быть внешними и внутренними.
1. **Внешние угрозы** связаны с вторжением в работу агентов КП других агентов, не входящих в этот КП (их называют **противниками).**

Противники делятся на следующие два класса.
(a) **Пассивные противники,** которые могут лишь перехватывать сообщения, пересылаемые другими агентами, и анализировать их.
(b) **Активные противники,** которые могут
• перехватывать сообщения, пересылаемые другими агентами, и анализировать их
• модифицировать или удалять пересылаемые сообщения
• создавать новые сообщения и посылать их другим агентам
• и т.д.
2. **Внутренние угрозы** связаны с наличием среди агентов КП нечестных агентов (их называют **мошенниками)**, которые могут
• неправильно выполнять действия, которые им полагается выполнять в ходе работы КП,
• выдавать себя за других агентов,
• объединяться в коалиции и совместно использовать сообщения, получаемые ими в ходе работы КП, в злонамеренных целях
• и т.д.
Свойства безопасности КП представляют собой описание видов угроз безопасности, которым может быть оказано противодействие во время функционирования этого КП.
Наиболее надёжными считаются такие КП, для которых имеется математическое доказательство утверждения о том, что реализованные в этих КП механизмы противодействия угрозам безопасности действительно выполняют поставленные перед ними задачи.

8.5.2 Шифрование сообщений

Для противодействия угрозам нарушения конфиденциальности некоторые из пересылаемых сообщений могут передаваться от одних агентов КП другим агентам в зашифрованном виде.
Шифрование сообщения ***m*** представляет собой применение некоторого алгоритма к паре *(K,m),* где *K* - объект, называемый **ключом шифрования.** Результат применения алгоритма шифрования к паре *(K, m)* обозначается знакосочетанием *K(m).*
Операция извлечения исходного сообщения из зашифрованного сообщения *m'* называется **дешифрованием.** Данная операция тоже представляет собой применение некоторого алгоритма к паре (*K',m'),* где *K'* - объект, называемый **ключом дешифрования.** Результат применения алгоритма дешифрования к паре *(K', m')* обозначается знакосочетанием *K'(m').* Ключи шифрования *K* и дешифрования *K'* должны быть связаны соотношением

для каждого сообщения m $\quad K'(K(m)) = m$ (8.34)

Если символ *K* обозначает некоторый ключ шифрования, то знакосочетание K^{-1} обозначает ключ дешифрования *K',* удовлетворяющий условию (8.34).

Системой шифрования (СШ) называется пара, состоящая из алгоритма шифрования и алгоритма дешифрования. Используемые на практике СШ подразделяются на два класса:

- **симметричные СШ,** в которых каждый ключ шифрования *K* совпадает с соответствующим ему ключом дешифрования K^{-1}, и
- **СШ с открытым ключом,** в которых ключи шифрования и дешифрования сопоставлены конкретным агентам: если символ *A* обозначает некоторого агента, использующего СШ с открытым ключом, то знакосочетания K^{+}_{A} и K^{-}_{A} обозначают сопоставленные ему ключи шифрования и дешифрования в этой СШ, причём

— ключ шифрования K^{+}_{A} (называемый **открытым ключом (ОК)** агента *A)* известен всем агентам

— ключ дешифрования K^{-}_{A} (называемый **закрытым ключом (ЗК)** агента *A)* известен только агенту *A*

(т.е. если какой-либо агент докажет, что он умеет дешифровать сообщения вида K^{+}_{A} (*m),* то он совпадает с *A)*

Главные различия между СШ с открытым ключом и симметричными СШ заключаются в следующем:

- по сравнению с симметричными СШ, СШ с открытым ключом являются существенно более стойкими к атакам, связанным с нарушением конфиденциальности зашифрованных сообщений путём криптоанализа,
- однако сложность выполнения операций шифрования и дешифрования в СШ с открытым ключом примерно на три порядка выше сложности этих операций в симметричных СШ.

Поэтому во многих КП использование этих СШ носит комбинированный характер:

- Симметричные СШ используются для основной связи, но ключи, используемые в этих СШ, регулярно обновляются.

Это делается для того, чтобы у противника, перехватывающего сообщения, не было большого количества перехваченных сообщений, зашифрованных на одном и том же ключе, по которым он мог бы

— вычислить используемый ключ,

— или иным образом нарушить конфиденциальность

путём криптоанализа перехваченных сообщений.

- Новые ключи для симметричных СШ пересылаются использующим их агентам в зашифрованном виде, причём шифрование этих ключей происходит с использованием СШ с открытым ключом.

8.5.3 Формальное описание КП

Одним из простейших видов формального описания КП является задание списка $\sum$ действий, каждое из которых имеет вид

$$A \to B : m \quad (8.35)$$

где *A* и *B* - имена агентов, и *m* - выражение, значением которого является сообщение или неопределённый объект. Выполнение действия (8.35) происходит следующим образом. Если значение выражения *m* определено, то

- *A* посылает *B* сообщение, равное значению выражения *m,* и
- *B* принимает это сообщение.

Если значение выражения ***m*** не определено, то это действие пропускается.

Действия, входящие в список $\sum$, выполняются в соответствии с тем порядком, в котором они входят в этот список. Если текущее действие не относится к какому-либо из агентов, то этот агент не функционирует в момент исполнения этого действия.

В описание КП могут также входить условия, выражающие, например, меру доверия одних агентов другим агентам.

8.5.4 Примеры КП

КП продажи компьютера

У *A* есть компьютер, а, у *B* есть деньги. *B* хочет купить у *A* компьютер, для чего *B* должен передать *A* деньги, а *A* должен передать *B* компьютер.

A и *B* не верят друг другу, поэтому

- *A* хочет сначала получить деньги, и после этого передать *B* компьютер
- *B* хочет сначала получить компьютер, и после этого отдать *A* деньги.

Таким образом, решить проблему продажи компьютера силами лишь *A и B* невозможно.

Данная проблема может быть решена при помощи КП, в котором кроме *A и B* принимает участие **доверенный посредник** *T,* относительно которого *у A и B* имеется уверенность в том, что *T* будет точно выполнять все действия, предписанные ему этим КП.

Искомый КП может иметь следующий вид:

- $A \to T$: компьютер
- $B \to A$: деньги

• $A \to T$: подтверждение или опровержение того, что полученная от *B* сумма соответствует стоимости компьютера *(A* посылает опровержение вместе с деньгами *B)*
• $T \to B$:(от *A* поступило подтверждение) ? компьютер
• $T \to A$:(от *A* поступило опровержение) ? компьютер
В этом КП значение выражения вида *b?m,* где *b* - условие, и *m* - сообщение
• равно *m,* если *b* истинно, и
• не определено, если *b* ложно.
Читателю предлагается самостоятельно
• разработать язык спецификаций, в терминах которого можно было бы
— выразить условие того, что *A и B* доверяют посреднику Т, и
— сформулировать спецификацию *Spec,* выражающую свойство корректности этого КП
• проанализировать соответствие этого КП спецификации *Spec*
— с учётом условия, что *A и B* доверяют *T,* и
— без учёта этого условия.

Обедающие криптографы

За круглым столом сидят три криптографа и обедают. После того, как они пообедали, официант сообщает им, что их обед полностью оплачен, но не уточняет, кто именно платил. Возможен один из двух вариантов:
• обед оплатил один из криптографов,
• за обед заплатила ФСБ.
Криптографы хотят выяснить, какой именно из вариантов имеет место, причём, если имеет место первый вариант, то те из них, которые не платили, не должны узнать, кто же конкретно заплатил.
Для решения этой задачи предлагается следующий КП.
Поскольку криптографы сидят за круглым столом, то каждая пара соседей может подбрасывать монету между собой, так, чтобы результат был известен только им двоим.
После того, как все три пары подбросили монету, каждый из них знает результаты двух подбрасываний (решка или орёл), которые могут быть
• либо одинаковыми (оба раза была решка, или оба раза был орёл),
• либо разными (один раз была решка, а другой - орёл).

Каждый криптограф говорит другим "одинаково" или "по разному", причём тот, кто заплатил, говорит противоположное (т.е. если надо сказать "одинаково" то он говорит "по-разному", и наоборот) .
Если число ответов "по разному" четно, то это значит, что обед оплатила ФСБ, иначе - один из них.

Подтверждение приёма

Имеется группа агентов, которые используют одну и ту же СШ с открытым ключом, обладающую следующим свойством: алгоритм дешифрования этой СШ может использоваться также и для шифрования, причём имеет место условие:

$$\text{для каждого агента } A \text{ и каждого сообщения } m \quad K_A^+(K_A^-(m)) = m \tag{8.36}$$

(большинство СШ с открытым ключом обладает этим свойством).
Члены этой группы хотят обмениваться друг с другом зашифрованными сообщениями.
Если, например, агент *A* хочет послать агенту *B* сообщение *m,* то для этого *A и B* предполагают использовать следующий КП из двух действий:

$$\begin{cases} A \to B : (A, K_B^+(K_A^-(m))) \\ B \to A : (B, K_A^+(K_B^-(m))) \end{cases}$$

т.е. *A* посылает *B* пару, состоящую из

- своего имени *(A),* чтобы *B* знал, от кого ему пришло сообщение, и
- шифрованного сообщения $K_B^+(K_A^-(m))$.

Получив сообщение $K_B^+(K_A^-(m))$,агент *B* при помощи своего ЗК K^-_B извлекает из него сообщение $K^-_A(m)$, из которого он, используя ОК K^+_A, может извлечь *m* (это возможно на основании (8.36)).
Второе действие в этом КП представляет собой подтверждение агентом *B* получения сообщения от *A:* оно посылается отправителю для того, чтобы он был уверен, что его сообщение *m* дошло до *B* в неискажённом виде.
К сожалению, данный КП не содержит механизма противодействия угрозам, связанным с мошенничеством. Если некоторый агент *M* из этой группы перехватил сообщение

$$(A, K_B^+(K_A^-(m)))$$

то он может извлечь из него сообщение *m,* например, следующим способом.

- *M* посылает *B* сообщение $(M, K_B^+(K_A^-(m)))$

• *B* обрабатывает его в соответствии с КП, т.е. извлекает из него вторую компоненту и вычисляет сообщение

$$K_M^+(K_B^-(K_B^+(K_A^-(m)))) \quad = K_M^+(K_A^-(m))$$

• Получившийся результат не содержит никакого смысла, но, согласно используемому КП, *B* обязан послать *M* подтверждение

$$(B, K_M^+(K_B^-(K_M^+(K_A^-(m)))))$$

• Из полученного сообщения агент *M,* используя свой ЗК K^-_M и известные ему ОК K^+_B и K^+_A, извлекает *m.*

КП двусторонней аутентификации

Имеется группа агентов, которые используют
• одну и ту же СШ с открытым ключом, и
• одну и ту же симметричную СШ.
Члены этой группы хотят обмениваться друг с другом зашифрованными сообщениями.
Каждая пара агентов *A, B* из этой группы использует для основной связи симметричную СШ, и в процессе своего взаимодействия *A* и *B* предполагают регулярно обновлять ключи шифрования для этой СШ, т.е.
• некоторый промежуток времени (который называется **сеансом взаимодействия)** агенты *A* и *B* используют для шифрования своих сообщений один ключ (называемый **сеансовым ключом),**
• затем при помощи СШ с открытым ключом этот ключ обновляется, и для шифрования сообщений в следующем сеансе взаимодействия *A и B* используется новый сеансовый ключ
• и т.д.
В целях противодействия возможному мошенничеству, члены группы хотят производить обновление сеансовых ключей при помощи КП, который должен решать следующие задачи:
• агенты *A* и *B,* желающие создать новый сеансовый ключ, должны принимать равноправное участие в создании этого ключа
• перед тем, как начать очередной сеанс взаимодействия, они хотят удостовериться в подлинности друг друга, т.е.
— агент *A* намерен удостовериться, что тот агент, совместно с которым он генерирует новый сеансовый ключ, это действительно *B,* а не мошенник, выступающий от имени *B,* и — агент *B* имеет аналогичное намерение.
Процедура взаимодействия агентов, целью которой является проверка подлинности кого-либо из них, называется **аутентификацией этого агента**. Если в результате взаимодействия *A* и *B* они доказывают свою

подлинность друг другу, то такая аутентификация называется двусторонней.

Агенты *A* и *B* собираются решить задачи

- двусторонней аутентификации, и
- генерации нового сеансового ключа K_{AB}

при помощи следующего КП из трёх действий (данный КП называется **КП Нидхема - Шредера** (Needham - Schroeder, 1979), мы будем сокращённо обозначать его знакосочетанием NS):

$$\begin{cases} A \to B : K_B^+(A, N_A) \\ B \to A : K_A^+(N_A, N_B) \\ A \to B : K_B^+(N_B) \end{cases}$$

где

- символ *A* в первом сообщении обозначает имя агента *A,*
- символы N_A и N_B обозначают битовые строки, длина которых равна длине создаваемого сеансового ключа K_{AB}. Данные строки называются нонсами. Как правило, ноне представляет собой псевдослучайную последовательность битов.

Нонсы предназначены для решения следующих двух задач.

1. Каждый из нонсов является вкладом сгенерировавшего его агента в формирование ключа K_{AB}*:* данный ключ по определению полагается равным побитовой сумме по модулю 2 нонсов N_A и N_B.

Нетрудно видеть, что условие равноправного участия агентов *A* и *B* в формировании ключа K_{AB} выполнено.

2. Нонсы решают задачу аутентификации следующим образом.

- Когда *B* посылает *A* сообщение $K_A^+(N_A,N_B)$, он тем самым демонстрирует свою способность извлечь из полученного им зашифрованного сообщения $K^+_B (A, N_A)$ нонс N_A (т.к., по предположению, *B* не мог получить ноне N_A никаким другим способом).

Это убеждает *A* в том, что тот агент, с которым он взаимодействует, знает ЗК K^-_B (т.к., по предположению, не зная K^-_B , извлечь сообщение *m* из сообщения вида $K^+_B (m)$ невозможно), т.е. этот агент действительно есть *B*.

- Когда *A* посылает *B* сообщение $K^+_B (N_B)$, он тем самым демонстрирует свою способность извлечь из полученного им зашифрованного сообщения $K^+_A(N_A, N_B)$ нонс N_B (т.к., по предположению, *A* не мог получить ноне N_B никаким другим способом).

Это убеждает *B* в том, что тот агент, с которым он взаимодействует, знает ЗК K^-_A (т.к., по предположению, не зная K^-_A, извлечь сообщение *m*

из шифртекста вида $K^{+}_{A}(m)$ невозможно), т.е. этот агент действительно есть *A*.

Одна из уязвимостей данного КП связана с тем, что один из агентов (например, *B)* может мошенничать, выдавая себя за другого агента. Работа КП NS в этом случае может иметь следующий вид.

1. После того, как агент *A* выразит желание взаимодействовать с *B* по КП NS, и пришлёт ему первое сообщение (т.е. $K^{+}_{B}(A, N_A)$), агент *B* может использовать это сообщение для того, чтобы

- взаимодействовать по КП NS с ещё одним агентом *C,*
- и в этом взаимодействии *B* будет выдавать себя за агента *A.*

Для этого *B*

- договаривается с C о взаимодействии с ним по КП NS,
- и посылает ему сообщение $K^{+}_{C}(A, N_A)$.

Получив это сообщение и расшифровав его, *C* делает вывод, что это сообщение было послано агентом *A* (о чём говорит ему первая компонента расшифрованного сообщения).

На основании этого, *C*

- генерирует нонс *Nc,* и
- посылает агенту *B* сообщение

$$K^{+}_{A}(N_A, N_C) \quad (8.37)$$

(думая, что он посылает его агенту *A)*

2. *B* пересылает полученное от *C* сообщение (8.37) агенту *A,* эта пересылка представляет собой второе действие в КП NS взаимодействия *A и B.*

3. *A* рассматривает извлечённую из полученного сообщения (8.37) компоненту N_C как нонс, сгенерированный агентом *B.*

В соответствии с третьим действием КП NS взаимодействия *A и B,* агент *A* посылает агенту *B* сообщение

$$K^{+}_{B}(N_C)$$

B извлекает N_C, и посылает агенту *C* сообщение $K^{+}_{C}(N_C)$.

C рассматривает эту пересылку как третье действие в КП NS взаимодействия *A и C.*

После этого *B* знает нонсы N_A и N_C, которые позволит ему вычислить сеансовый ключ для шифрованного взаимодействия с C, в котором он будет выступать от имени *A.*

Читателю предлагается самостоятельно

- разработать язык спецификаций, в терминах которого можно было бы выразить свойство КП противодействовать мошенничеству описанного выше вида, и

• разработать алгоритм проверки соответствия КП спецификациям, выраженным на этом языке.

9. Представление структур данных в виде процессов

9.1 Понятие структуры данных

Структура данных (СД) - это дискретная динамическая система, каждое состояние которой можно интерпретировать как совокупность **данных** (т.е. некоторых значений), хранящихся в текущий момент в этой системе.
Как правило, каждое состояние СД представляет собой совокупность из нескольких компонентов, каждая из которых является содержимым некоторого ресурса (например, содержимым ячейки памяти).
Одним из способов формального описания СД является представление их в виде процессов.
Если процесс *P* является представлением некоторой СД, то имена в *P* можно интерпретировать как точки доступа к данным, содержащимся в этой СД. Объекты, соответствующие этим именам, могут иметь виртуальный характер. Например, в качестве таких объектов могут выступать криптографические ключи.
Представление СД в виде процессов позволяет формализовать понятие **активных данных (= знаний), которые развиваются в результате наличия у них неполноты или противоречивости**. Активность структур данных заключается в том, что они могут сами инициировать запросы к окружающей среде с целью получения дополнительной информации для достижения согласованности в знаниях, которые содержатся в этих СД.
В этой главе мы приводим несколько примеров описания СД в виде процессов. Каждый из этих процессов задаётся в виде рекурсивного определения.
Ниже мы будем отождествлять процесс, представляющий некоторую СД, с самой этой СД.

9.2 СД "память с 2^k ячейками"

СД "память с 2^k ячейками" (где $k \geq 0$) - это процесс

$$[\![MEM_k(\vec{x})]\!] \quad (9.1)$$

где MEM_k - процессное имя, и $\vec{x}$ - список из 2^k переменных вида

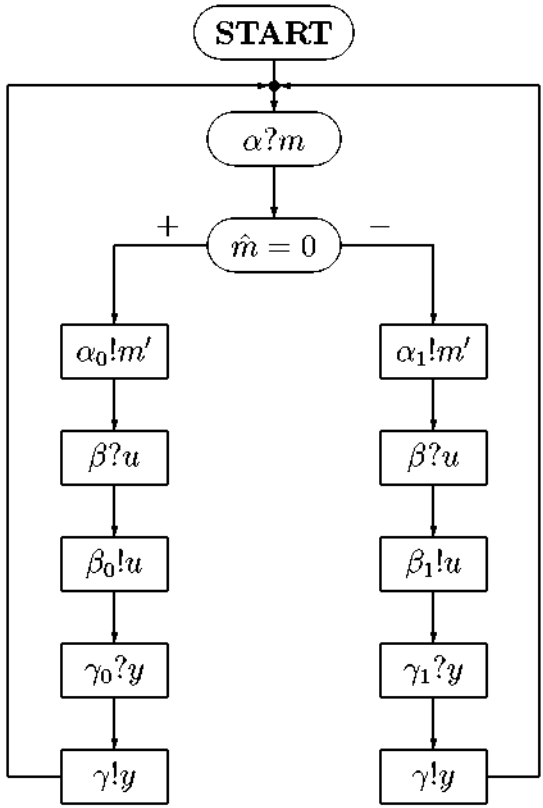
$$\vec{x} = \{x_m \mid m \in \{0,1\}^k\}$$

каждая из которых соответствует некоторой **ячейке памяти:**
для каждого $m \in \{0,1\}^k$ значение переменной x_m представляет собой содержимое ячейки памяти с адресом *m.*
Мы будем предполагать, что процесс $[\![MEM_k(\vec{x})]\!]$ работает следующим образом:
1. сначала, если $k > 0$, то через точку доступа α в процесс $[\![MEM_k(\vec{x})]\!]$ поступает адрес *m,*
2. затем через точку доступа β в процесс $[\![MEM_k(\vec{x})]\!]$ поступает значение *v,* которое надо записать в ячейке по адресу *m*
3. после чего через точку доступа γ процесс $[\![MEM_k(\vec{x})]\!]$ выдаёт значение, которое содержалось в ячейке по адресу *m* до выполнения предыдущего шага.

Процессы, соответствующие процессным выражениям вида $MEM_k(\vec{x})$, определяются следующим РО:
• $MEM_0(x) = \beta?y\ .\ \gamma!x\ .\ MEM_0(y)$
(процесс $[\![MEM_0(x)]\!]$ - это - ячейка памяти, хранящая значение *x)*
• для каждого $k \geq 0$

$$MEM_{k+1}(\vec{x} \cdot \vec{y}) = \left(\begin{array}{l} SWITCH \mid \\ MEM_k(\vec{x})\,[\,\alpha_0/\alpha,\ \beta_0/\beta,\ \gamma_0/\gamma\,] \mid \\ MEM_k(\vec{y})\,[\,\alpha_1/\alpha,\ \beta_1/\beta,\ \gamma_1/\gamma\,] \end{array} \right) \setminus \left\{ \begin{array}{l} \alpha_0, \beta_0, \gamma_0, \\ \alpha_1, \beta_1, \gamma_1 \end{array} \right\}$$

где
— $\vec{x} \cdot \vec{y}$ - список из 2^{k+1} различных переменных, представленный в виде конкатенации двух списков из 2^k переменных
— *SWITCH* - это процесс, называемый **переключателем,** и представляемый следующей блок-схемой:

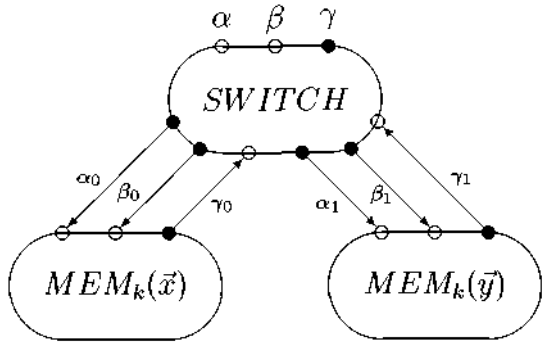


где

— $\hat{m}$ =первый бит в битовой строке *m,* и

— *m′* = строка из *k*— 1 битов, получаемая из *m* удалением первого бита.

Процесс $[\![MEM_{k+1}(\vec{x}\cdot\vec{y})]\!]$ изображается следующим потоковым графом:



Читателю предлагается самостоятельно доказать, что процесс $[\![MEM_k(\vec{x})]\!]$ удовлетворяет следующей спецификации: если $k > 0$, то для каждого процесса P

$$\left(\begin{array}{l} MEM_k(\vec{x}) \mid \\ \alpha!m \,.\, \beta!u \,.\, \gamma?y \,.\, P \end{array} \right) \setminus \{\alpha,\beta,\gamma\} \stackrel{+}{\approx}$$
$$\stackrel{+}{\approx} \left(\begin{array}{l} (x_m := u) \,.\, MEM_k(\vec{x}) \mid \\ (y := x_m) \,.\, P \end{array} \right) \setminus \{\alpha,\beta,\gamma\}$$

Операция

- чтения значения, содержащегося в ячейке процесса $[\![MEM_k(\vec{x})]\!]$

по адресу m (без изменения содержимого этой ячейки), и

- занесения этого значения в переменную y процесса P представляется процессным выражением

$$\left(\begin{array}{l} MEM_k(\vec{x}) \mid \\ \alpha!m \,.\, \beta!0 \,.\, \gamma?y \,.\, \alpha!m \,.\, \beta!y \,.\, \gamma?z \,.\, P \end{array} \right) \setminus \{\alpha,\beta,\gamma\} \qquad (\text{где } z \notin P) \tag{9.2}$$

Читателю предлагается самостоятельно доказать, что

$$(9.2) \stackrel{+}{\approx} \left(\begin{array}{l} MEM_k(\vec{x}) \mid \\ (y := x_m) \,.\, P \end{array} \right) \setminus \{\alpha,\beta,\gamma\}$$

Можно изменить определение процесса $[\![MEM_k(\vec{x})]\!]$ так, чтобы новый процесс удовлетворял спецификации

$$MEM_k(\vec{x}) \stackrel{+}{\approx}$$
$$\stackrel{+}{\approx} \alpha?m. \left(\beta?y \,.\, (x_m := y) \,.\, MEM_k(\vec{x}) \;+\; \gamma!x_m \,.\, MEM_k(\vec{x}) \right)$$

т.е.

- сначала в процесс $[\![MEM_k(\vec{x})]\!]$ вводится адрес *m,*
- а потом, в зависимости от выбора окружающей среды,

— либо в $[\![MEM_k(\vec{x})]\!]$ записывается новое значение по адресу ***m***

— либо читается значение, хранящееся в $[\![MEM_k(\vec{x})]\!]$ по адресу ***m***

Для этого надо по-другому определить процесс $[\![MEM_0(x)]\!]$:

$$MEM_0(x) = \beta?y\ .\ MEM_0(y)\ +\ \gamma!x\ .\ MEM_0(x)$$

Работа с новым процессом $[\![MEM_k(\vec{x})]\!]$ происходит следующим образом.

Операция записи значения выражения e в ячейку по адресу ***m*** представляется процессным выражением

$$\left(\ MEM_k(\vec{x})\ \mid\ \alpha!m\ .\ \beta!e\ .\ \mathbf{0}\ \right)$$

Операция чтения процессом *P* значения, записанного в ячейке процесса $[\![MEM_k(\vec{x})]\!]$ по адресу *m,* и занесения этого значения в переменную *y* процесса *P,* представляется процессным выражением

$$\left(\ MEM_k(\vec{x})\ \mid\ \alpha!m\ .\ \gamma?y\ .\ P\ \right)$$

9.3 СД "стек"

Стек - это СД с точками доступа α и γ, с которой можно выполнять следующие операции:

1. положить значение *v* в стек (действие $\alpha\,!\,v$)
2. если стек непуст, то взять головной элемент стека (действие γ?*x*).

СД "стек" представляется совокупностью процессных выражений

$$\{PUSH_n(x_1,\ldots,x_n)\mid n\geq 0\}\quad (9.3)$$

где для каждого $n \geq 0$ $PUSH_n$- процессное имя, и ПВ

$$PUSH_n(x_1,\ldots,x_n)$$

представляет процесс "стек, хранящий *n* значений".

Процессы, соответствующие ПВ из совокупности (9.3) определяются следующим РО:

1. $PUSH_0 = \alpha?x\ .\ PUSH_1(x)\ +\ \gamma!\$\ .\ \mathbf{0}$

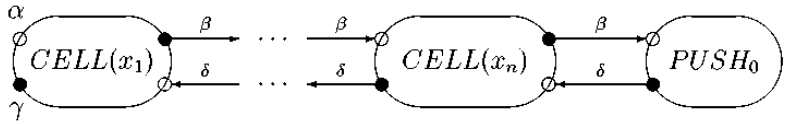
2. для каждого *n ≥0*

$$PUSH_{n+1}(x, x_1, \ldots, x_n) = \\ = \left(\begin{array}{l} CELL(x) \mid \\ PUSH_n(x_1, \ldots, x_n)[\beta/\alpha, \delta/\gamma] \end{array} \right) \setminus \{\beta, \delta\}$$

где ПВ *CELL(x)* представляет процесс с точками доступа *α,β, γ, δ,* называемый **ячейкой памяти,** и определяемый следующим образом:

$$CELL(x) = \\ = \left(\begin{array}{l} \alpha?y \,.\, \beta!x \,.\, CELL(y) + \\ \gamma!x \,.\, \delta?y \,.\, \left(\; (y = \$) \; ? \; PUSH_0 \; + \; (y \neq \$) \; ? \; CELL(y) \; \right) \end{array} \right)$$

Потоковый граф процесса $[\![PUSH_n(x_1, \ldots, x_n)]\!]$ имеет следующий вид:



Читателю предлагается самостоятельно доказать, что совокупность процессов, соответствующих ПВ из (9.3), удовлетворяет следующей спецификации: для каждого *n ≥0*

$$PUSH_{n+1}(x, x_1, \ldots, x_n) \overset{+}{\approx} \left(\begin{array}{l} \alpha?y \,.\, PUSH_{n+2}(y, x, x_1, \ldots, x_n) + \\ \gamma!x \,.\, PUSH_n(x_1, \ldots, x_n) \end{array} \right)$$

9.4 СД "очередь"

Очередь - это СД с точками доступа α и γ, с которой можно выполнять следующие операции:

1. добавить значение *v* в конец очереди (действие $\alpha\,!\,v$)
2. если очередь непуста, то взять первый элемент очереди (действие $\gamma\,?\,x$).

СД "очередь" представляется совокупностью процессных выражений

$$\{QUEUE_n(x_1, \ldots, x_n) \mid n \geq 0\} \quad (9.4)$$

где для каждого $n \geq 0$ $QUEUE_n$- процессное имя, и ПВ

$$QUEUE_n(x_1, \ldots, x_n)$$

представляет процесс "очередь, хранящая *n* значений".

Процессы, соответствующие ПВ из совокупности (9.4) определяются следующим РО:

$$1.\; QUEUE_0 = \alpha?x \,.\, QUEUE_1(x) \; + \; \gamma!\$ \,.\, \mathbf{0}$$

2. для каждого *n* > 0

$$QUEUE_{n+1}(x, x_1, \ldots, x_n) =$$
$$= \left(\begin{array}{l} CELL(x) \mid \\ QUEUE_n(x_1, \ldots, x_n)[\beta/\alpha, \delta/\gamma] \end{array} \right) \setminus \{\beta, \delta\}$$

где ПВ *CELL(x)* представляет процесс с точками доступа *α,β, γ, δ,* называемый **ячейкой памяти,** и определяемый следующим образом:

$$CELL(x) =$$
$$= \left(\begin{array}{l} \alpha?y \,.\, \left(\beta!x \,.\, CELL(y) + \gamma!x \,.\, CELL(y) \right) + \\ \gamma!x \,.\, \delta?y \,.\, \left(\; (y = \$) \,?\, QUEUE_0 + (y \neq \$) \,?\, CELL(y) \; \right) \end{array} \right)$$

Потоковый граф процесса $[\![QUEUE_n(x_1, \ldots, x_n)]\!]$ имеет такой же вид, как и потоковый граф в пункте 9.3.

Читателю предлагается самостоятельно доказать, что совокупность процессов, соответствующих ПВ из (9.4), удовлетворяет следующей спецификации: для каждого *n≥0*

$$QUEUE_{n+1}(x, x_1, \ldots, x_n) \overset{+}{\approx}$$
$$\overset{+}{\approx} \left(\begin{array}{l} \alpha?y \,.\, QUEUE_{n+2}(x, x_1, \ldots, x_n, y) + \\ \gamma!x \,.\, QUEUE_n(x_1, \ldots, x_n) \end{array} \right)$$

10. Семантика языка параллельного программирования

Семантика языка программирования представляет собой правило, сопоставляющее каждой конструкции этого языка некоторый математический объект. В качестве такого объекта может выступать, например, логическая формула или процессное выражение.

Главной целью определения семантики языка программирования является сведение задач анализа свойств программ на этом языке к задачам анализа математических утверждений, соответствующих этим свойствам.

В этой главе мы рассмотрим в качестве примера определение семантики простейшего языка параллельного программирования в терминах теории процессов.

10.1 Описание языка параллельного программирования

В этом параграфе мы описываем простейший язык параллельного программирования, который мы будем обозначать символом $\mathcal{L}$

10.1.1 Конструкции языка $\mathcal{L}$

Конструкции языка $\mathcal{L}$ делятся на следующие три класса.

1. Выражения.

Понятие выражения языка $\mathcal{L}$ совпадает с аналогичным понятием, определённым в параграфе 7.2.3. Выражения строятся из переменных, констант и ФС. Каждому выражению e сопоставлен некоторый тип *type(e).*

2. Декларации.

Каждая декларация *D* имеет один из следующих трёх видов:

(a) **объявление локальной переменной**:

$$\texttt{VAR}\ x \quad (10.1)$$

где *x* - имя переменной

(b) **объявление ресурса**:

$$\texttt{RESOURCE}\ R \quad (10.2)$$

где *R* - имя ресурса, который может представлять собой, например, устройство ввода или вывода

(c) **описание процедуры:**

$$\texttt{PROCEDURE}\ G(u,v)\ \texttt{IS}\ C \quad (10.3)$$

где

- *G* - имя процедуры,
- *u* - переменная, изображающая аргумент процедуры
- *v* - переменная, изображающая результат процедуры
- *C* - оператор (тело процедуры), в который переменные *u* и *v* входят как формальные параметры этой процедуры.

3. Операторы.

Каждый **оператор представляет собой описание некоторого алгоритма**.

Ниже

- операторы обозначаются символом *C*
- декларации обозначаются символом *D*
- переменные обозначаются символами *x, y, z, u, v, ...*

• выражения обозначаются символом *e*
• формулы (т.е. выражения типа *boot)* обозначаются символом *b*
причём при всех этих символах могут быть индексы.
Операторы имеют следующий вид.
(a) **присваивание:**
$x := e$
где$type(x) = type(e)$
(b) условный переход:
$\texttt{if}\ b\ \texttt{then}\ C_1\ \texttt{else}\ C_2$
(c) цикл:
$\texttt{while}\ b\ \texttt{do}\ C$
(d) ввод:
$\texttt{input}\ x$
(e) вывод:
$\texttt{output}\ e$
(f) пустой оператор:
$\texttt{skip}$
(g) последовательная композиция:
$C_1; C_2$
(h) параллельная композиция:
$C_1\ \texttt{par}\ C_2$
(i) блок:
$\texttt{begin}\ \{D_1, \ldots, D_k; C\}\ \texttt{end}$

Блок "связывает" все вхождения объектов, объявленных в декларациях D_1,..., D_k: эти объекты являются видимыми только внутри этого блока.
(j) **вызов процедуры:**
$\texttt{call}\ G(e, z)$

где
• *G* - имя вызываемой процедуры,
• *e* - выражение, значение которого является аргументом процедуры *G*, и
• *z* - переменная, в которую будет занесён результат выполнения процедуры *G*.
(k) **связывание оператора с ресурсом:**
$\texttt{with}\ R\ \texttt{do}\ C$

где *R* - имя некоторого ресурса.

10.1.2 Программы на языке $\mathcal{L}$

Программа на языке $\mathcal{L}$ представляет собой оператор

$$\texttt{begin}\ \{D_1, \ldots, D_k; C\}\ \texttt{end}$$

где все переменные, ресурсы, и процедуры, имена которых входят в С, объявлены

- в декларациях $D_1, \ldots, D_k$,
- или в тех декларациях, которые содержатся в C.

Программа может взаимодействовать с окружающей средой только путём выполнения операторов ввода и вывода.

10.2 Семантика языка $\mathcal{L}$

В этом параграфе мы определяем семантику языка $\mathcal{L}$, которая представляет собой правило, сопоставляющее каждой конструкции Q языка $\mathcal{L}$ некоторое ПВ $\langle\!|Q|\!\rangle$,называемое **семантикой конструкции** Q.

При определении этой семантики мы будем использовать следующие обозначения и соглашения.

1. В каждом ПВ B, соответствующем какой-либо конструкции языка $\mathcal{L}$, знакосочетания

$$\delta!, \quad \rho!, \quad i? \quad \textbf{и} \quad o!$$

обозначают действия, имеющие заранее предопределённый смысл:

(a) δ! обозначает сигнал о завершении исполнения процесса, соответствующего процессному выражению B

(b) $\rho!$ обозначает вывод значения, являющегося результатом работы процесса, соответствующего процессному выражению B

(c) $i?$ обозначает ввод значения в программу

(d) o! обозначает вывод значения из программы

2. Символ **done** обозначает ПВ $\delta!\ .\ \mathbf{0}$

3. Для каждой пары ПВ B_1, B_2

(a) знакосочетание $B_1\ \texttt{before}\ B_2$ обозначает ПВ

$$\Big(\ B_1[\beta/\delta]\ |\ \beta?.B_2\ \Big) \setminus \{\beta\}$$

где β не входит в B_1 и B_2

(b) знакосочетание $B_1\ \texttt{result}\ B_2$ обозначает ПВ

$$(B_1 \mid B_2) \setminus \{\rho\}$$

(c) знакосочетание $B_1\ \texttt{par}\ B_2$обозначает ПВ

$$\left(\begin{array}{l} B_1[\delta_1/\delta] \mid \\ B_2[\delta_2/\delta] \mid \\ \delta_1!.\delta_2!.\mathtt{done} + \delta_2!.\delta_1!.\mathtt{done} \end{array} \right) \setminus \{\delta_1, \delta_2\}$$

где δ_1и δ_2 не входят в B_1 и B_2

4. Для каждого объекта с именем *n,* объявленного в некоторой декларации *D,* символ L_n обозначает подмножество множества *Names,* состоящее из **точек доступа** к этому объекту:

$$L_n \stackrel{\text{def}}{=} \begin{cases} \{\alpha_x, \gamma_x, \pi_x, \varphi_x\}, & \text{если } D = (10.1) \\ \{\pi_R,\ \varphi_R\}, & \text{если } D = (10.2) \\ \{\alpha_G,\ \gamma_G\}, & \text{если } D = (10.3) \end{cases}$$

Если имена n_1 и n_2 таких объектов различны, то

$$L_{n_1} \cap L_{n_2} = \emptyset$$

10.2.1 Семантика выражений

Каждому выражению *e* языка $\mathcal{L}$ соответствует ПВ $\langle\!|e|\!\rangle$, определяемое индукцией по построению выражения *e*:

- для каждой переменной $x \in Var$

$$\langle\!|x|\!\rangle \stackrel{\text{def}}{=} \gamma_x?y.\ \rho!y.\ \mathbf{0}$$

- для каждой константы $c \in Con$

$$\langle\!|c|\!\rangle \stackrel{\text{def}}{=} \rho!c.\ \mathbf{0}$$

- для каждого ФС *f*

$$\langle\!|f(e_1, \ldots, e_n)|\!\rangle \stackrel{\text{def}}{=}$$
$$\stackrel{\text{def}}{=} \left(\begin{array}{l} \langle\!|e_1|\!\rangle\ [\rho_1/\rho] \mid \ \ldots\ \mid \langle\!|e_n|\!\rangle\ [\rho_n/\rho] \mid \\ \rho_1?x_1.\ \ldots\ \rho_n?x_n\,.\,\rho!\ f(x_1, \ldots, x_n)\ .\ \mathbf{0} \end{array} \right) \setminus \{\rho_1, \ldots, \rho_n\}$$

10.2.2 Семантика деклараций

При описании семантики деклараций мы будем использовать вспомогательные ПВ, соответствующие процессам, определяемым в виде РО:

- $REG_x(y) = \alpha_x?z\ .\ REG_x(z)\ + \gamma_x!y\ .\ REG_x(y)$
  ($[\![REG_x(y)]\!]$ = регистр для хранения значения переменной x)
- $LOC_x \stackrel{\text{def}}{=} \alpha_x?y\ .\ REG_x(y)$
  (регистр без начального содержания)
- $SEM_x = \pi_x!\ .\ \varphi_x!\ .\ SEM_x$
- $SEM_R = \pi_R!\ .\ \varphi_R!\ .\ SEM_R$

Семантика деклараций имеет следующий вид.

1. Если $D = \texttt{VAR}\ x$, то

$$\langle\!|D|\!\rangle \stackrel{\text{def}}{=} LOC_x\,|\,SEM_x$$

2. Если $D = \texttt{RESOURCE}\ R$, то

$$\langle\!|D|\!\rangle \stackrel{\text{def}}{=} SEM_R$$

3. Если $D = \texttt{PROCEDURE}\ \ G(u,v)\ \ \texttt{IS}\ C$, то процессному выражению $\langle\!|D|\!\rangle$ соответствует РО

$$\langle\!|D|\!\rangle = \left(\begin{array}{l} LOC_u\,| \\ LOC_v\,| \\ \alpha_G?x\ .\ \alpha_u!x\ .\ ((\langle\!|D|\!\rangle\,|\,\langle\!|C|\!\rangle) \setminus L_G) \\ \quad \texttt{before}\ \gamma_v?y\ .\ \gamma_G!y\ .\ \langle\!|D|\!\rangle \end{array}\right) \setminus (L_u \cup L_v)$$

Если в РО для $\langle\!|\texttt{PROCEDURE}\ldots|\!\rangle$ вместо

$$(\langle\!|D|\!\rangle\,|\,\langle\!|C|\!\rangle) \setminus L_G$$

было бы написано просто $\langle\!|C|\!\rangle$, то неправильно транслировались бы такие процедуры, которые рекурсивно вызывают сами себя, т.е. такие *G*, в теле которых есть оператор $\texttt{call}\ G$.

Вышеприведённая семантика деклараций вида (10.3) является неидеальной. Она неправильно сопоставляет ПВ таким операторам, в которых есть параллельные вызовы одной и той же процедуры, например, оператору вида

$$\texttt{call}\ G(6,z)\ \texttt{par}\ \texttt{call}\ G(7,w)$$

Если известно максимальное число *n* доступных процессоров (т.е. одновременно *n* процедур могут быть активными), то семантику декларации *D* вида (10.3) лучше определить в виде РО

$$\left\{\begin{array}{l} \langle\!| D |\!\rangle = \langle\!| D |\!\rangle_1 \mid \ldots \mid \langle\!| D |\!\rangle_n \\ \forall i = 1, \ldots, n \\ \quad \langle\!| D |\!\rangle_i = \alpha_{G,i}?x \,.\, \left(\begin{array}{l} LOC_u \mid \\ LOC_v \mid \\ \alpha_u!x \,.\, ((\langle\!| D |\!\rangle \mid \langle\!| C |\!\rangle) \setminus L_G) \\ \quad \texttt{before}\ \gamma_v?y \,.\, \gamma_{G,k}!y \,.\, \langle\!| D |\!\rangle_i \end{array} \right) \setminus (L_u \cup L_v) \end{array}\right. \tag{10.4}$$

где $L_G \stackrel{\text{def}}{=} \{\alpha_{G,1}, \ldots, \alpha_{G,n},\ \gamma_{G,1}, \ldots, \gamma_{G,n}\}$.

10.2.3 Семантика операторов

1. $\langle\!| x := e |\!\rangle \stackrel{\text{def}}{=} \pi_x? \,.\, \langle\!| e |\!\rangle\ \texttt{result}\ (\rho?y \,.\, \alpha_x!y \,.\, \varphi_x? \,.\, \texttt{done})$

2. $\langle\!| C_1; C_2 |\!\rangle \stackrel{\text{def}}{=} \langle\!| C_1 |\!\rangle\ \texttt{before}\ \langle\!| C_2 |\!\rangle$

3. $\langle\!|\, \texttt{if}\ e\ \texttt{then}\ C_1\ \texttt{else}\ C_2 |\!\rangle \stackrel{\text{def}}{=}$
$\stackrel{\text{def}}{=} \langle\!| e |\!\rangle\ \texttt{result}\ \rho?x \,.\, (x\ ?\ \langle\!| C_1 |\!\rangle\ +\ \neg x\ ?\ \langle\!| C_2 |\!\rangle)$

4. если $C = \texttt{while}\ e\ \texttt{do}\ C'$, то ПВ $\langle\!| C |\!\rangle$ связано со следующим РО:

$$\langle\!| C |\!\rangle = \langle\!| e |\!\rangle\ \texttt{result}\ (\rho?x. \left(\begin{array}{l} x\ ?\ \langle\!| C' |\!\rangle\ \texttt{before}\ \langle\!| C |\!\rangle\ + \\ \neg x\ ?\ \texttt{done} \end{array} \right))$$

5. $\langle\!|\, \texttt{begin}\ \{D_1, \ldots, D_k; C\}\ \texttt{end} |\!\rangle \stackrel{\text{def}}{=}$
$\stackrel{\text{def}}{=} (\langle\!| D_1 |\!\rangle \mid \ldots \mid \langle\!| D_k |\!\rangle \mid \langle\!| C |\!\rangle) \setminus (L_{D_1} \cup \ldots \cup L_{D_k})$

6. $\langle\!| C_1\ \texttt{par}\ C_2 |\!\rangle \stackrel{\text{def}}{=} \langle\!| C_1 |\!\rangle\ \texttt{par}\ \langle\!| C_2 |\!\rangle$

7. $\langle\!|\, \texttt{input}\ x |\!\rangle \stackrel{\text{def}}{=} i?y \,.\, \alpha_x!y \,.\, \texttt{done}$

8. $\langle\!|\, \texttt{output}\ e |\!\rangle \stackrel{\text{def}}{=} \langle\!| e |\!\rangle\ \texttt{result}\ (\rho?x \,.\, o!x \,.\, \texttt{done})$

9. $\langle\!|\, \texttt{skip} |\!\rangle \stackrel{\text{def}}{=} \texttt{done}$

10. $\langle\!|\, \texttt{call}\ G(e, z) |\!\rangle \stackrel{\text{def}}{=}$
$\stackrel{\text{def}}{=} \langle\!| e |\!\rangle\ \texttt{result}\ (\rho?x \,.\, \alpha_G!x \,.\, \gamma_G?z \,.\, \alpha_z!z \,.\, \texttt{done})$

11. $\langle\!|\, \texttt{with}\ R\ \texttt{do}\ C |\!\rangle \stackrel{\text{def}}{=} \pi_R? \,.\, \langle\!| C |\!\rangle\ \texttt{before}\ (\varphi_R? \,.\, \texttt{done})$

Если известно максимальное число *n* доступных процессоров (т.е. одновременно *n* процедур могут быть активными), и семантика деклараций вида (10.3) определяется в соответствии с (10.4), то семантику оператора $\texttt{call}\ G(e,z)$ следует определить следующим образом:

$$\langle\!|\texttt{call}\ G(e,z)|\!\rangle \stackrel{\text{def}}{=}$$
$$\stackrel{\text{def}}{=} \langle\!|e|\!\rangle\ \texttt{result}\ (\rho?x\ .\ \sum_{i=1}^{n}\ \alpha_{G,i}!x\ .\ \gamma_{G,i}?z\ .\ \alpha_z!z\ .\ \texttt{done})$$

Читателю предлагается самостоятельно
- определить семантику каких-либо современных языков параллельного или распределённого программирования (MPI и т.п.) или языков проектирования бизнес-процессов (BEPL и т.п.), и
- разработать на основе этой семантики автоматизированную систему доказательства свойств программ или описаний систем на этих языках.

11. Облачные вычисления

11.1. Общие сведения об облачных вычислениях

Без использования современных информационных технологий не может эффективно работать ни одно образовательное учреждение. "Облачные вычисления" (Cloud computing) являются хорошей альтернативой классической модели обучения.

11.1.1. Описание проблемы

Как мы сказали, без использования современных информационных технологий не может эффективно работать ни одно образовательное учреждение. При этом содержание и развитие собственной IT-инфраструктуры при каждом образовательном центре обходится очень дорого. С каждым годом уровень данных затрат все больше и больше возрастает. Учреждения расходуют большие суммы на **компьютерную технику, телекоммуникационное оборудование и**

программное обеспечение. Помимо вышеуказанных затрат значительные финансовые вложения требуются и для поддержания высокого уровня профессионализма этих сотрудников.

"Облачные вычисления" (Cloud computing) являются хорошей альтернативой классической модели обучения. Главным ее плюсом можно считать существенную экономию средств образовательного учреждения, в котором они используются. Ведь в этом случае компьютерная инфраструктура и/или информационные сервисы предоставляются как **услуги "облачного" провайдера**. **Документы, электронные письма, программы и прочие данные участников образовательного процесса хранятся на удаленных серверах провайдера.** При этом для учреждения нет необходимости содержать собственную дорогостоящую IT-инфраструктуру и переплачивать за вычислительные ресурсы, которые в большинстве случаев не задействованы на полную мощность. **Единственное, чем необходимо обеспечить преподавателей и обучающихся с использованием облачных технологий, – это доступ к сети Интернет.**

В настоящее время существует множество поставщиков облачных решений. **Такие крупные компании как Amazon, Google, Microsoft и т.д. предлагают значительные скидки образовательным учреждениям, за счёт чего они получают доступ к облачным сервисам практически бесплатно**.

Надежность, доступность и легкая масштабируемость являются ключевыми достоинствами облачных технологий. Может ли все это означать, что в скором времени большая часть образовательных услуг будет предоставляться на базе облачных вычислений? Приведет ли это к полному отказу образовательных учреждений от собственных громоздких IT-инфраструктур?

Настоящий курс ставит целью рассмотреть и оценить все преимущества и недостатки использования облачных вычислений в сфере образования, а также даёт практические рекомендации по применению облачных вычислений в процессе обучения как в школе, так и в вузах.

11.1.2. Характеристики облачных вычислений

В облачных вычислениях выделяют следующие ключевые характеристики:

- **Самообслуживание по требованию.** Потребитель самостоятельно выбирает, каким набором вычислительных возможностей и ресурсов он будет пользоваться (например, сетевые хранилища, базы данных, процессорное время, объем оперативной памяти). Также потребитель может при необходимости изменять этот набор без согласования с провайдером в автоматическом режиме.
- **Высокая эластичность (гибкость) сервисов.** Вычислительную мощность можно легко уменьшить или увеличить, исходя из потребностей пользователя. В случае высокой нагрузки на сервис количество ресурсов оперативно повышается, в случае уменьшения нагрузки – ресурсы освобождаются. Если образовательному учреждению потребуется срочно увеличить объем вычислительных ресурсов, то руководству учреждения не придется тратить средства и время на закупку и настройку дополнительного оборудования и программного обеспечения, которое впоследствии может использоваться достаточно редко.
- **Возможность объединение ресурсов.** Вычислительные ресурсы "облачного" провайдера группируются в пулы с возможностью динамического перераспределения **физических и виртуальных ресурсов** между конечными потребителями. С применением современных технологий виртуализации это позволяет "облачному" провайдеру легко наращивать мощности и заменять вышедшее из строя оборудование без снижения уровня производительности и надежности.
- **Учет потребления ресурсов и оплата по факту использования.** Потребители платят только за фактически потребленные услуги (например, за объем переданной информации, пропускную способность и т.д.).
- **Технологичность.** Можно смело утверждать, что в **дата-центрах** поставщиков облачных услуг используются более современные инновационные технологии, чем в большинстве учебных заведений. Эти технологии позволяют автоматически оптимизировать использование вычислительных ресурсов и

сократить издержки на обслуживание оборудования по сравнению аналогичными издержками в учебных заведениях.

- **Отказоустойчивость и высокий уровень доступности.** Дата-центры для облачных вычислений представляют собой надежную распределенную сеть, узлы которой могут располагаться в различных уголках мира. Отказоустойчивость у такой сети как правило заведомо выше любой пользовательской локальной сети, т.к. обеспечивается многократным резервированием и квалифицированным обслуживанием технического персонала. В итоге, такая распределенная сеть позволяет получить услуги с высоким уровнем доступности. Позволить себе организовать подобную сеть дата-центров может далеко не каждое образовательное учреждение. Кроме того, дата-центры как правило строят вблизи дешевых источников электроэнергии, что является экономически более целесообразным, чем поддержание работоспособности ИТ-инфраструктуры при работе по обычным для небольших потребителей тарифам на электроэнергию.

11.1.3. Классификация облачных вычислений

В облачных вычислениях традиционно выделяют три типа (уровня):

- Инфраструктура как услуга.
- Платформа как услуга.
- Программное обеспечение как услуга.

Рассмотрим подробно каждый из этих типов, т.к. каждый из них имеет свою целевую аудиторию и цели, о которых нужно иметь чёткое представление при переходе с традиционной парадигмы организации вычислений на "облачную".

На рис. 1.1 в виде обобщенной схемы представлен каждый из перечисленных выше видов облачных услуг, названия которых приведены в центре в виде перевернутой пирамиды. Больший размер блока пирамиды означает, что он включает в себя всю инфраструктуру более маленького блока. Например, для предоставления сервиса "Платформа как услуга" с точки зрения поставщика услуг необходимо

также иметь возможность обеспечить сервис "Инфрастуктура как услуга".

В левой части рисунка указано, каким видом ИТ-ресурсов необходимо обладать, чтобы предоставить соответствующие услуги. В правой части рисунка перечислены виды целевой аудитории предоставляемых облачных услуг.

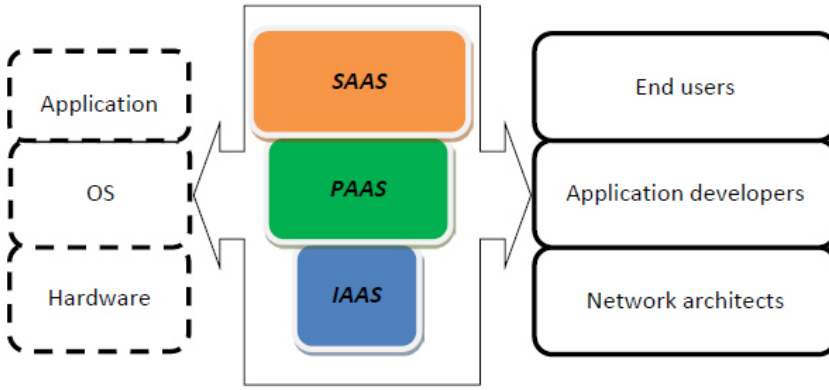


Рис. 1.1. Виды "облачных" услуг

Инфраструктура как услуга (IaaS, Infrastructure as a Service). На этом уровне потребитель может самостоятельно конструировать свою IT-инфраструктуру в облаке и управлять ей. Например, **создавать виртуальные сети, добавлять виртуальное оборудование (серверы, хранилища, базы данных), устанавливать необходимое для работы прикладное программное обеспечение и операционные системы, т.е. использовать облако так, как если бы это была реальная IT-инфраструктура образовательного учреждения.** Самые известные IaaS-решения: Amazon CloudFormation, Google Compute Engine, Windows Azure.

Платформа как услуга (PaaS, Platform as a Service). **На этом уровне провайдер облачных услуг предоставляет пользователю доступ к операционным системам, системам управления базами данными, средствам разработки и тестирования.** Таким образом, потребитель облачных услуг получает возможность и средства для самостоятельного создания, тестирования и эксплуатации

программного обеспечения. При этом вся информационная инфраструктура (вычислительные сети, серверы и системы хранения) управляется провайдером. Вот перечень наиболее известных PaaS-сервисов:

- Google App Engine (для разработки программного обеспечения на языках Java, Python);
- Windows Azure (для ASP.NET, PHP);
- Cloud Foundry (языки программирования Java, Ruby, Scala).

Программное обеспечение как услуга (SaaS, software as a service). **На этом уровне поставщик предоставляет пользователям облака готовое программное обеспечение.** Все данные хранятся в облаке, и для доступа к ним пользователю требуется только наличие веб-браузера. Это наиболее интересный для образовательных учреждений тип облачных вычислений, поскольку он не требует дополнительных затрат на установку и настройку программного обеспечения, как это требуется при использовании IaaS и PaaS. Следует также иметь в виду, что в большинстве случае плата за использование программного обеспечения в рамках SaaS рассчитывается с учётом количества пользователей и не предполагает так называемых Enterprise-лицензий, позволяющих использовать некоторый сервис для любого количества пользователей без ограничений. **Примеры бесплатных SaaS-решений для образовательных учреждений – это Google Apps for Education и Microsoft Office 365 for education.** Они содержат в себе функции офисного пакета (работа с документами, таблицами и презентациями), средств коммуникации (электронная почта, календари, мгновенные сообщения) и средств эффектной подачи информации (в виде статических презентаций, видеороликов или интерактивных приложений).

11.1.4. Отличие облачных вычислений от Web2.0

Некоторые сотрудники сферы образования часто путают облачные вычисления с технологиями Веб 2.0, ошибочно полагая, что облачные вычисления – это любые сервисы, предоставляемые с помощью Интернет.

Перечислим типичные приложения Веб 2.0:

- онлайн-энциклопедии (например, http://www.wikipedia.org);
- блоги (например, http://www.livejournal.com);
- каналы RSS для рассылки дайджеста новостей с возможностью отслеживать обратную связь с читателями по количеству переходов на полную версию той или иной новости;
- сервисы mash-up, использующие в качестве источников информации другие сервисы (например, сервис поиска магазинов, который использует сторонний сервис для отображения найденных магазинов на карте);
- метки tags, позволяющие выявить наиболее популярные материалы среди пользователей на данный момент;
- медиа-библиотеки, формируемые участниками в режиме онлайн;
- социальные сети (например, http://www.vk.com);

Ключевой особенностью всех этих технологий является возможность онлайн-редактирования содержимого веб-страниц их посетителями. **При этом все приложения Веб 2.0 могут быть размещены как в облаке, так и в локальной IT-инфраструктуре использующего их учреждения.** Таким образом, главное отличие облачных вычислений от Веб 2.0 заключается в том, что приложения Веб 2.0 **– это только определенный вид программного обеспечения**, тогда как **облачные вычисления – это метод хранения данных и предоставления их конечному пользователю.**

11.1.5. Применение облачных вычислений в образовании

Одной из первых облачных услуг, которую стали использовать европейские образовательные учреждения, стала электронная почта. Обеспечение работоспособности (аутсорсинг) сервиса электронной почты – несложная задача, которая определенно не играет ключевой роли в работе образовательного учреждения. **Корпорации Google и Microsoft предоставляют сотрудникам и учащимся образовательных учреждений доступ к электронной почте бесплатно.**

Помимо услуг электронной почты эти корпорации обеспечивают возможность использовать в облаке функции стандартного

офисного пакета для совместной работы с электронными документами, таблицами и для создания презентаций. Облачные сервисы для образовательных организаций Google Apps for Education и Microsoft Office 365 for education позволяют использовать встроенные системы для обмена мгновенными сообщениями, календари для совместного планирования и общие адресные книги. **Каждый пользователь облачных систем получает значительное дисковое пространство для хранения любой информации, которая была получена в результате работы с облаком**.

Может показаться странным, что эти услуги предоставляются образовательным учреждениям бесплатно, в то время как для коммерческих организаций цены на программное обеспечения как были, так и остаются традиционно высокими. Такая ценовая политика объясняется следующим образом. На современном рынке облачных технологий сохраняется высокая конкуренция между поставщиками программного обеспечения, поэтому они стараются предоставлять свои сервисы образовательным учреждениям бесплатно. Расчет идет на будущих выпускников, которые после получения образования устроятся на работу и смогут убедить будущих работодателей приобрести программный продукт, о преимуществах которого они уже знают. Также это обеспечит привязанность и лояльность пользователей к продуктам определенной марки и её узнаваемость и популярность.

Если для образовательного учреждения безопасность доступа к данным не является приоритетным направлением, тогда может оказаться выгодным использование низкоуровневых IaaS-сервисов в качестве систем хранения данных, например для видео- и аудиоматериалов.

Для некоторых образовательных учреждений может оказаться выгодным перемещение в "облако" внутренних **систем управления обучением** (LMS, Learning Management Systems). Это хорошая возможность для таких учреждений, которые не могут позволить себе покупку и поддержку дорогостоящего оборудования и программного обеспечения, что позволяет оптимизировать расходы на IT-инфраструктуру в современных посткризисных условиях.

11.2. Характеристика основных типов "облачных" услуг

11.2.1. Программирование в "облаке"

В этом разделе показано, как можно использовать облачные вычисления при обучении основам программирования. **Приводится подробный пример работы с Web-сервисом, позволяющим создавать и отлаживать учебные программы на любом языке программирования с помощью облачного сервиса сайта http://ideone.com.**

Практика программирования предполагает активное использование **специализированных интегрированных средств разработки (IDE – Integrated Development Environment).** Их использование связано со следующими двумя сложностями:

- Настройка и установка IDE требует высокой квалификации системного администратора.
- Современные IDE достаточно требовательны к ресурсам вычислительной машины, на которой они используются.

Поясним каждый пункт подробно. Для обеспечения полнофункциональной работы IDE требуется, чтобы квалификация системного администратор, осуществляющего установку, настройку и поддержку IDE была достаточно высока. Это приводит к необходимости нанимать в учебные заведения на должность **системного администратора** высококвалифицированных сотрудников, заработная плата которых может оказаться существенной статьёй расходов в бюджете образовательного учреждения.

Более того, затраты образовательного учреждения могут возрасти вследствие того, что современные IDE требуют наличия высокопроизводительных вычислительных машин. Например, одна из самых распространенных IDE Microsoft Visual Studio 2012 требует для нормальной работы процессор мощностью 1,6 ГГц или выше, 1 ГБ ОЗУ (или 1,5 ГБ для виртуальной машины), 10 ГБ свободного дискового пространства. Для большинства задач образовательных учреждений не требуется компьютеров с такой высокой

производительностью, поэтому их покупка может оказаться недопустимой роскошью.

Обе указанные проблемы позволяет решить применение облачных технологий при обучении программированию. В настоящее время существуют большое количество так называемых **онлайн-IDE**, которые не требуют установки на компьютер пользователя и которые требуют для запуска лишь наличие **Интернет-браузера**. Системные требования браузеров к оборудованию вычислительной машины традиционно являются скромными. Например, популярный Веб-браузер **Mozilla Firefox 17** требует для установки процессор от 1300 МГц, 512 МБ ОЗЦ и 200 МБ свободного дискового пространства, что существенно меньше приведённых ранее цифр для IDE Microsoft Visual Studio 2012.

Рассмотрим ниже, как можно использовать онлайн-IDE в учебных заведениях для обучения основам программирования на примере http://ideone.com. Этот сервис позволяет в режиме онлайн создавать тексты программ на разных языках программирования и запускать эти программы на исполнение с возможностью анализа полученных результатов. Основные рабочие элементы Ideone показаны на рис. 2.1.

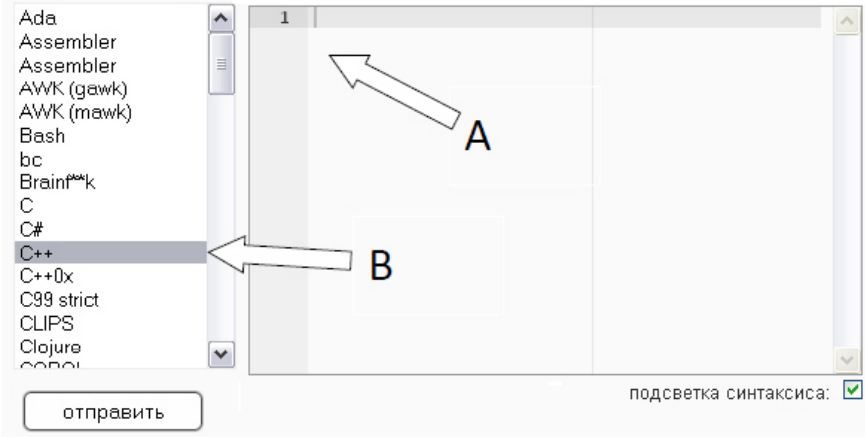


Рис. 2.1. Основные рабочие элементы Ideone

В поле "**A**" необходимо ввести текст программы, а в поле "**B**" нужно выбрать используемый язык программирования, затем нужно нажать кнопку "**Отправить**". Ideone поддерживает работу со следующими 55 популярными языками программирования: Ada, Assembler, AWK, Bash, bc, Brainf**k, C, C#, C++, C++0x, C99 strict, CLIPS, Clojure, COBOL, Common Lisp (clisp), D (dmd), Erlang, F#, Factor, Falcon, Forth, Fortran, Go, Groovy, Haskell, Icon, Intercal, Java, JavaScript, Lua, Nemerle, Nice, Nimrod, Node.js, Objective-C, Ocaml, Oz, PARI/GP, Pascal, Perl, PHP, Pike, Prolog, Python, R, Ruby, Scala, Scheme (guile), Smalltalk, SQL, Tcl, Text, Unlambda, VB.NET, Whitespace. Очевидно, что **этого перечня достаточно при обучении основам программирования практически в любом учебном заведении мира**. Более того, при использовании сервиса ideone.com у преподавателя появляется возможность использовать при обучении сразу несколько языков программирования без необходимости поддерживать работу нескольких IDE.

Покажем на примере, как может быть организована работа в группе при обучении основам программирования. Рис 2.2 иллюстрирует способ запуска простой программы на языке Си. Как можно видеть, в тексте программы используется подсветка синтаксиса, аналогичная той, что пользователи привыкли использовать в обычных офлайн-IDE. Однако при желании подсветка может быть отключена с помощью элемента управления "**A**". С помощью элемента управления "**B**" можно указать перечень входных данных для программы, что позволяет реализовать более сложную логику работы программы, чем в приведённом примере.

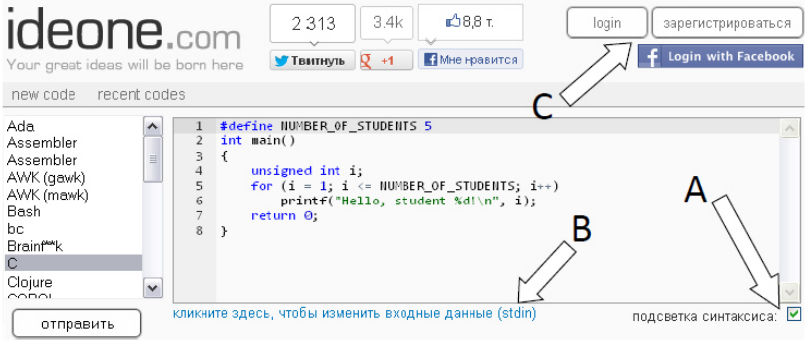


Рис. 2.2. Запуск программы в Ideone

Очень важным является элемент управления "**C**", который позволяет персонифицировать работу с программой. Данная возможность крайне ценна при организации учебного процесса. Если преподаватель попросит всех студентов зарегистрироваться в Ideone (или использовать для входа свою учётную запись Facebook), то **появляется возможность сделать процесс работы с программой коллективным, а процесс совместной работы с программой будет проходить с использованием современных технологий Web 2.0.** Подробнее об этом расскажем, используя рис. 2.3.

Рис 2.3 представляет собой результаты запуска программы, приведённой на рис. 2.2, самые важные из которых помечены знаком "**A**". Это консольный вывод программы и возвращаемое значение. Знаком "**B**" отмечена Интернет-ссылка, которую преподаватель может переслать студентам для ознакомления с результатам работы демонстрационной программы, либо сами студенты могут выслать подобную ссылку преподавателю в качестве отчёта о проделанной работе.

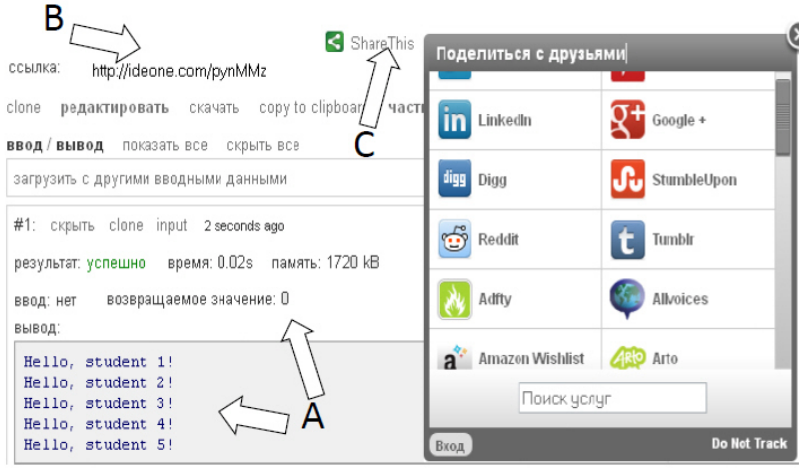


Рис. 2.3. Результаты работы программы в Ideone

Следующим этапом совместной работы может стать онлайн-обсуждение результатов работы программы с помощью средств Web

2.0 одного из популярных сервисов социальных сетей. Это становится возможным благодаря использованию элемента управления "**C**" на рис. 2.3.

Этот элемент управления позволяет выбрать из списка в правой части экрана один виджетов популярных сайтов социальных сетей.

К сожалению, онлайн-IDE Ideone позволяет реализовать не все из функций традиционных офлайн-IDE. Например, отсутствует возможность использовать функции работы с сетью, обращения к файлам и некоторые другие. Также невозможно запустить программу, время выполнения которой займёт более 15 секунд или потребности в оперативной памяти превысят 256 МБ, или объём программы превысит 64 КБ. Все эти ограничения являются достаточно серьёзными, если планируется использовать Ideone для разработки профессионального программного обеспечения. Однако для образовательных целей эти ограничения более чем приемлемы. Кроме того, для более требовательных преподавателей существуют платные и бесплатные сервисы, аналогичные Ideone, которые при этом в большей степени реализуют функционал традиционных офлайн-IDE. Таким примером являются сервисы Cloud9 IDE (www.c9.io), CodeRun. На рис. 2.4 показан вид интерфейса сервиса CodeRun: пользователи, работавшие с традиционными офлайн-IDE, сразу увидят много знакомых элементов управления. Присутствует панель со списком используемых классов, со списком задействованных в проекте файлов, а также окна с отладочной информацией о состоянии стека вызовов подпрограмм и с ошибками времени комплиляции или времени выполнения.

Созданный проект можно сохранить в офлайн, но предпочтительным является режим работы только в облаке. Все операции, включая отладочную сборку, анализ результатов выполнения в консольком режиме, компиляцию под разные платформы и операционные системы можно выполнить в режиме онлайн. В идеале окончанием работы программиста будет скачивание готовых бинарных файлов с работающей программой. Такой подход позволяет сэкономить используемой офлайн дисковое пространство, а также позволяет компилировать проект существенно более быстро, чем на рабочем месте пользователя, если это рабочее место оборудовано устаревшим аппаратным обеспечением. Итогом этого является возможная финансовая экономия для образовательного учреждения.

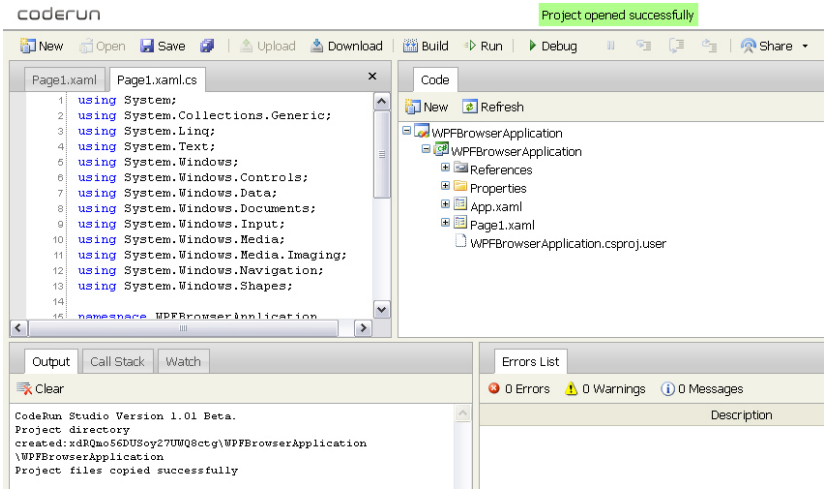


Рис. 2.4. Интерфейс системы Coderun

Однако помимо экономического эффекта, можно получить и существенные преимущества при организации образовательного процесса. Студенты получают возможность совместно редактировать программные проекты, находясь у себя дома. Это позволяет реализовывать сложные курсовые проекты и лабораторные работы с существенной экономией на осуществление организационных мероприятий со стороны преподавателя.

11.2.2. "Облачные" сервисы хранения данных

В этом разделе показано, как можно использовать облачные технологии для хранения различных типов данных в "облаке". В качестве примера рассматривается популярный сервис "Dropbox", официальный сайт https://www.dropbox.com.

На конец 2012 в сети Интернет существовало более 30-ти **бесплатных(!)** сервисов облачного хранения данных. **Каждый из них предлагает возможности по хранению данных любых типов, начиная от офисных документов и заканчивая мультимедийной информацией.**

Почти все (около 90%) из поставщиков этих сервисов предлагают следующие услуги бесплатно:

- Объём бесплатного хранилища – 2 и более ГБ (до 18 ГБ предлагает сервис Dropbox, до 20 ГБ – Яндекс.Диск, 50 ГБ – Adrive).
- Автоматическая синхронизация хранимых данных между всеми устройствами, которые подключены к облачному сервису. **Не нужно использовать внешнее запоминающее устройство (Flash-диск, CD/DVD-накопители) для того, чтобы перенести данные на другое устройство (ПК, ноутбук, планшет, смартфон и т.д.).** При подключении устройства к сети Интернет актуальная версия данных будет автоматически загружена на устройство. Эта функция экономит массу времени – можно быстро продолжить работу над текущей учебной задачей, вернувшись домой из учебного заведения.
- Безопасность хранения данных в "облаке". Весь трафик между клиентом и "облаком" шифруется (используется, как минимум, протокол SSL, а в некоторых RSA+AES), что очень сильно затрудняет просмотр передаваемой информации посторонними лицами. Поэтому уровень безопасности работы с данными выше, чем, например, при отправке обычным письмом по электронной почте. Некоторые сервисы облачного хранения (SpiderOak, Wuala) предлагают шифрование данных не только при передаче, но и при хранении в "облаке".
- Возможность публичного доступа через Интернет к файлам, хранящимся в облаке, для любого человека. Достаточно отправить коллеге ссылку на нужный файл, чтобы он смог ознакомиться, например, с результатами вчерашнего теста или последней версией учебного материала.
- Надежность хранения данных. Поставщики облачных решений хранят данные на своих сервисах с использованием избыточности, что само по себе гарантирует надежность. Дополнительно к этому, на любом из устройств, подключенных к "облаку", хранится, ещё как минимум, еще одна актуальная копия данных.

Для образовательного учреждения использование онлайн-сервисов хранения данных является экономически выгодным. Для создания сетевого файлового хранилища своими силами необходимо:

- приобрести сетевое и серверное оборудование;
- разработать политики хранения и общего доступа к информации;
- произвести первоначальную установку и настройку программного обеспечения;
- обеспечить регулярное резервное копирование данных и возможность быстрого восстановления;
- выделить рабочий персонал для администрирования файлового хранилища.

Финансовые затраты на проведение вышеперечисленных мероприятий зависят от числа пользователей и могут быть очень большими. Например, для организации файлового хранилища для 1000 учащихся с выделением 5 ГБ дискового пространства на каждого потребуется закупить минимум 4 жестких диска емкостью 2,5 ТБ (10 ТБ для обеспечения дублирования хранимых данных по технологии RAID 1), высокопроизводительный RAID-контроллер и серверное оборудование. Помимо дополнительных расходов на заработную плату обслуживающему персоналу, необходимо помнить о регулярных расходах на оплату счетов за потребляемую оборудованием электроэнергию.

Регистрация. Начать использовать любой облачный сервис по хранению данных очень легко (см. пример для сервиса Dropbox на рис. 2.5). Как правило для регистрации требуется **указать действующий адрес электронной почты и придумать новый пароль для доступа к "облаку"**.

Сразу после регистрации пользователю доступно 2 ГБ в **файловом хранилище**. Однако выполнив ряд несложных действий, можно увеличить доступный размер бесплатного хранилища в несколько раз. Для этого нужно порекомендовать сервис Dropbox друзьям или выполнить иные действия рекламного характера.

Существует два режима работы с облачным сервисом: через вэб-интерфейс и через программу-клиент. Рассмотрим подробно каждый из них ниже.

Рис. 2.5. Страница регистрации в сервисе "Dropbox"

Работа через веб-интерфейс. Веб-интерфейс работы с облачным хранилищем очень прост. Для работы с ним нужен только доступ к сети Интернет и вэб-браузер. На рис. 2.6 показаны возможности работа в популярном сервисе облачного хранения данных "Dropbox". Меткой "**А**" указаны общие элементы управления данными:

- загрузка новых файлов (пользователь указывает, какой файл на своем компьютере он желает поместить в "облако");
- добавление новых каталогов (каталоги позволяют удобно структурировать данные пользователя для ускорения доступа к ним);
- предоставление общего доступа (пользователь может указать, кому именно следует дать права на чтение или запись его файлов);
- удаление (если файл стал не нужен, то его можно удалить из "облака").

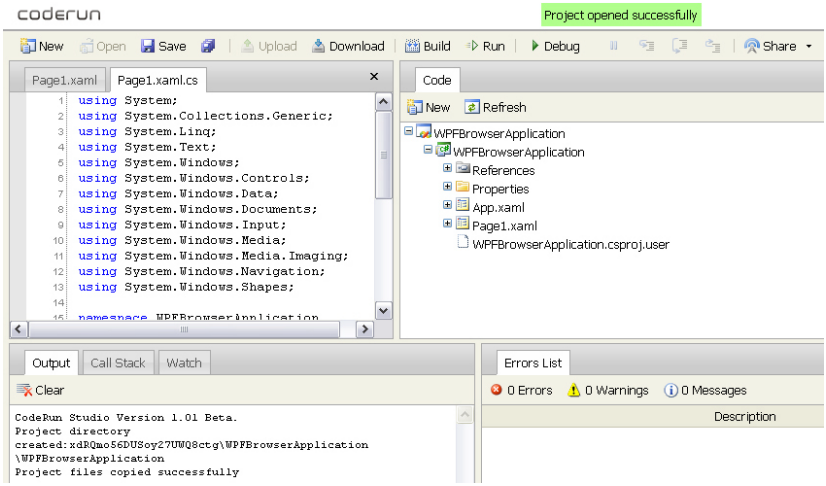


Рис. 2.6. Веб-интерфейс сервиса "Dropbox"

Рядом с элементом "**A**" располагается строка поиска "**B**". Общий доступ на чтение к файлам и каталогам можно предоставить, нажав на значок "**C**", после чего на указанные электронные адреса участников будет отправлена ссылка на выбранные объекты общего доступа.

Работа через программу-клиент. Режим работы через программу-клиент предоставляет больше возможностей пользователю. Для работы с ней необходимо скачать установочный файл и установить программу на локальный компьютер. У самых популярных "облачных" сервисов имеется **кросс-платформенные клиенты**, которые можно запускать из разных операционных систем (Windows/Linux/Mac/Android и т.д.). Как видно ниже на рис. 2.7, после установки программы-клиента необходимо ввести регистрационные данные и указать имя устройства (Computer name), данные которого будут синхронизироваться с "облаком".

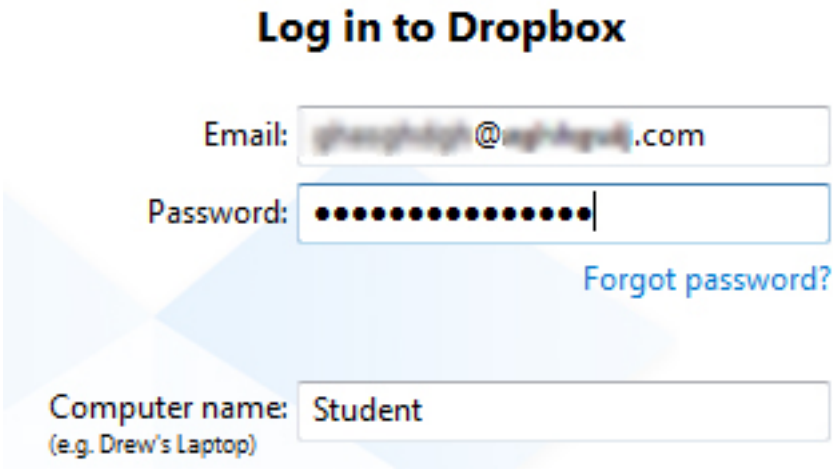


Рис. 2.7. Ввод регистрационных данных в Dropbox

После входа в клиентскую программу нужно настроить один (в сервисе Dropbox) или несколько каталогов (в сервисах SpiderOak, Box, Wuala, BitCasa и другие), содержимое которых будет синхронизироваться с "облаком". На рис. 2.8 показано, как может выглядеть содержимое каталога Dropbox при запуске программы-клиента на операционной системе Ubuntu 11.10.

Важно понимать относительность понятия "Computer name", которое используется в большей степени не для идентификации того или иного компьютера, а для идентификации конкретного набора файлов в облаке. Поэтому если установить клиент Dropbox на другие устройства, но указать в программе-клиенте те же учетные данные, то при изменении файлов в каталоге "Dropbox" на одном устройстве, файлы в каталоге "Dropbox" на всех остальных устройствах с одинаковым "Computer name" будут также обновлены. Это утверждение справедливо не только для сервиса Dropbox, но и для прочих клиентских программ облачных сервисов хранения данных.

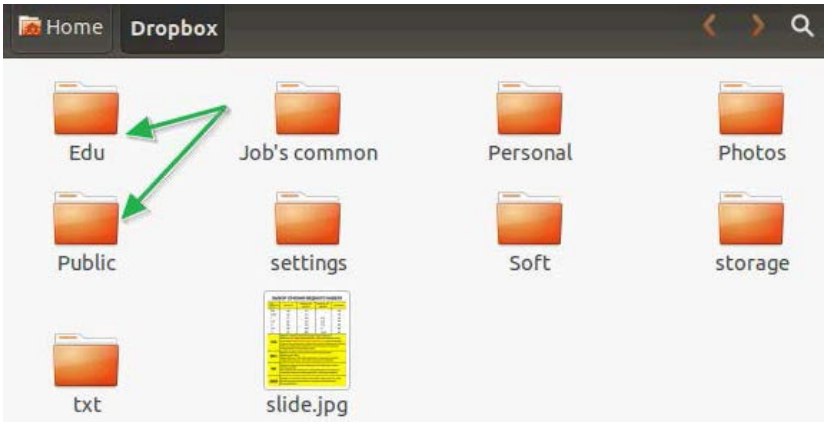
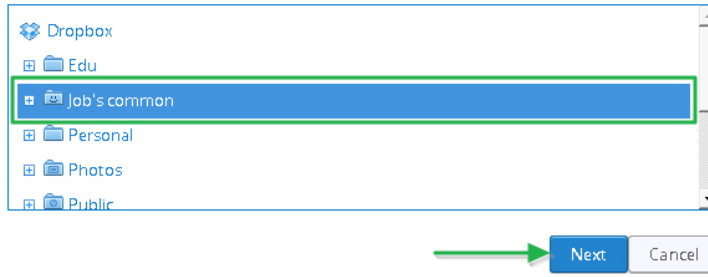


Рис. 2.8. Содержимое синхронизированного каталога "Dropbox" в Ubuntu 11.10

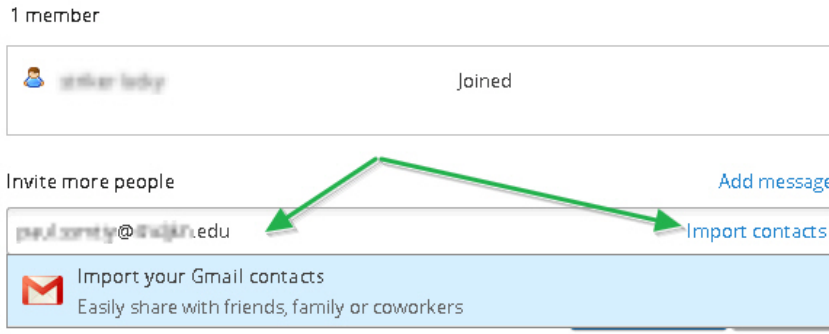
Предоставление общего доступа. Общий доступ к файлу или каталогу в "облаке" осуществляется в три шага (на примере "Dropbox"):

- В веб-клиенте в разделе "**Sharing**" (предоставление общего доступа) необхо-димо нажать кнопку "**New shared folder**":

- Далее следует выбрать существующий каталог либо создать новый:



- Затем добавить адреса электронной почты участников проекта, которым необходимо предоставить общий доступ к указанному каталогу:



После этих действий каждый из участников получит по электронной почте ссылку, по которой он сможет открыть общий ресурс и начать участие в проекте.

Общие рекомендации по использованию онлайн-сервисов хранения данных в образовательных учреждениях можно сформулировать следующим образом. **В учебных заведениях можно предложить следующий вариант использования хранения данных в "облаке".**

Прежде всего, необходимо зарегистрировать участников образовательного процесса на сервисе "облачного" хранения. Затем создать общие каталоги и настроить права доступа пользователей к ним. Например, учебные материалы по курсу можно поместить в каталог, к которому сделать доступ только на чтение. Для каждого слушателя можно сделать индивидуальный каталог с полным доступом для сдачи лабораторных работ или отчетов. Это один из вариантов использования облачных сервисов хранения данных. Можно придумать любую другую удобную схему обмена данными для решения задач образования в выбранном учебном заведении.

Сравнение облачных сервисов для хранения данных. Как было отмечено в начале раздела, существует достаточное количество онлайн-сервисов хранения данных. Для удобства выбора характеристики самых популярных их этих сервисов сведены в табл. 2.1. Критерием выбора тех сервисов, которые попали в сравнительную таблицу, была в том числе минимизация стоимости лицензии, т.к. это является важным фактором при выборе облачных услуг в образовательных учреждениям. Поэтому в табл.1 включены данные только о бесплатных сервисов тех или иных поставщиков услуг.

Таблица 2.1. Сравнение бесплатных облачных услуг для хранения данных

Название	**Бесплатный объем, ГБ**	**Шифрование данных**	**Поддерживаемые операционные системы**	**Общий доступ**	**Коллективная работа**	**Кол-во клиентских компьютеров**
Drop box	2	SSL, AES 256	Windows, Mac OS, Linux, Android, iOS	Да	Нет	
Spider Oak	2	RSA 2048, AES 256	Windows, Mac OS, Linux, Android, iOS	Да	Нет	
MS Sky Drive	7	SSL, AES 128	Android, iOS, Windows, Mac OS	Да	Да	

Box.com	5	SSL, AES 256	Android, Windows Mobile,Ipad, Iphone	Да	Да	
Wuala	5	AES 256, RSA 2048, SHA-256	Windows, Mac OS, Linux, Android, iOS	Да	Нет	
Adrive	50	SSL	Android, iOS	Да	Да	1
Яндекс. Диск	10	Нет	Windows, Mac OS, Linux, Android, iOS	Да	Нет	

Подводя итоги краткого сравнения "облачных" сервисов хранения данных, можно сделать следующие выводы:

- любые современные онлайн-сервисы предлагают достаточное количество дискового пространства для хранения документов и учебных материалов;
- почти все сервисы поддерживают современные алгоритмы шифрования при передаче данных;
- в качестве хранилища резервных копий учебных материалов можно использовать сервис "Adrive" благодаря **бесплатному объему в 50 ГБ**;
- если необходима конфиденциальность хранения данных, то самый высокий уровень защиты при передаче и хранении информации в облаке обеспечивает SpiderOak (шифрование данных происходит на клиентском устройстве);
- для совместной работы над документами и электронными таблицами прекрасно подойдет сервис Box.com.

Онлайн-сервисы хранения данных обладают большими преимуществами по сравнению с локальными сетевыми хранилищами. Использование в процессе обучения одного или нескольких облачных сервисов хранения данных значительно повысит его эффективность, а также позволит образовательному учреждению идти в ногу со временем.

11.2.3. Защита информации при использовании сервисов облачного хранения

Рассмотрим вопросы обеспечения защиты информации при использовании сервисов облачного хранения на примере Dropbox. Поскольку условия обслуживания Dropbox могут измениться без предварительного уведомления пользователя (т.е. учебного заведения), а данные могут храниться на сервисе Dropbox достаточно долгое время (для использования в учебном процессе в течение последующих лет), необходимо уделить особое внимание защите информации при использовании сервисов облачного хранения. Для этого необходимо решить следующие задачи:

- Защита персональных данных.
- Защита метаданных в открытых файлах.
- Защита доступа к закрытым данным.

Защита персональных данных. Для защиты персональных данных учащихся, студентов, преподавателей и других сотрудников учебного заведения необходимо применять следующие меры: 1) не предоставлять избыточную информацию и 2) отключить элемент **cookies** (куки) в используемом браузере. Рассмотрим подробно каждую из двух предложенных мер.

Для обеспечения защиты персональных данных не стоит предоставлять избыточную информацию при регистрации на сервисе в личном профиле, т.к. потенциальный злоумышленник может использовать избыточную информацию в преступных целях. Например, не имеет смысла указывать следующие персональные:

- номер телефона;
- сведения о кредитной карте;
- почтовый адрес;
- сведения о регистрации в социальных сетях.

Для корректной работы облачных сервисов, как правило, требуются лишь минимальный набор персональных данных: имя пользователя, действующий адрес электронной почты и пароль.

Второй рекомендацией является совет отключить **cookies** (куки) в браузере для того, чтобы при посещении сайта не сохранялась информация о персональных предпочтениях и настройках пользователя, не записывались данные о логине и пароле пользователя, не велась статистика различных персональных особенностей работы пользователя. Например, для того чтобы отключить куки в браузере Mozilla Firefox, необходимо всего лишь отключить галочку "**Принимать куки с сайтов**" в настройках браузера на странице "**Приватность**" (рис. 2.9).

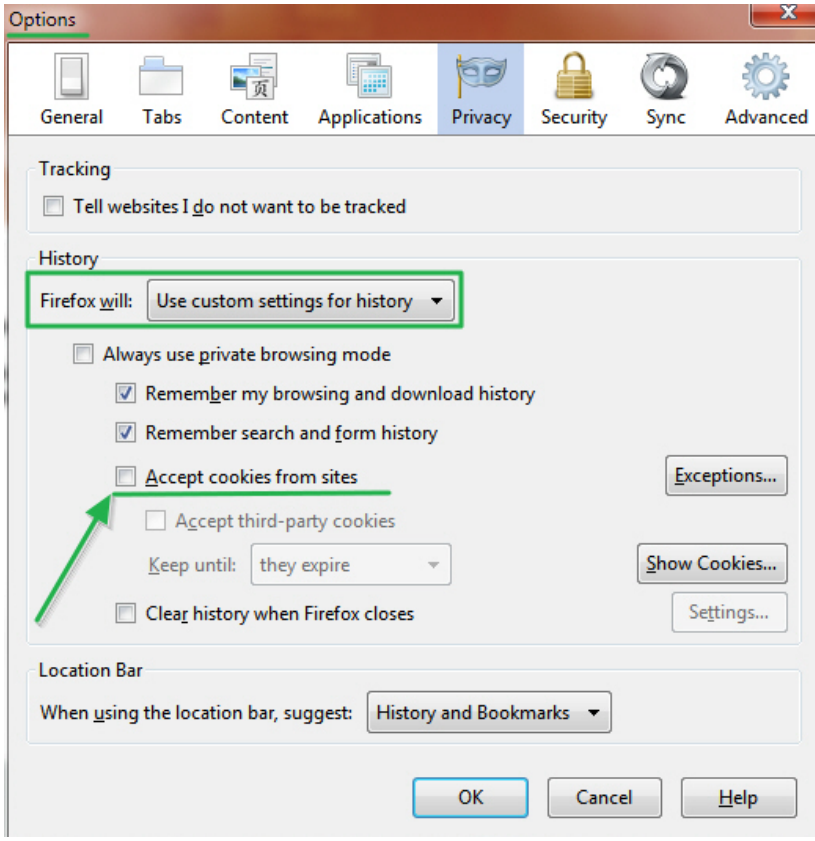


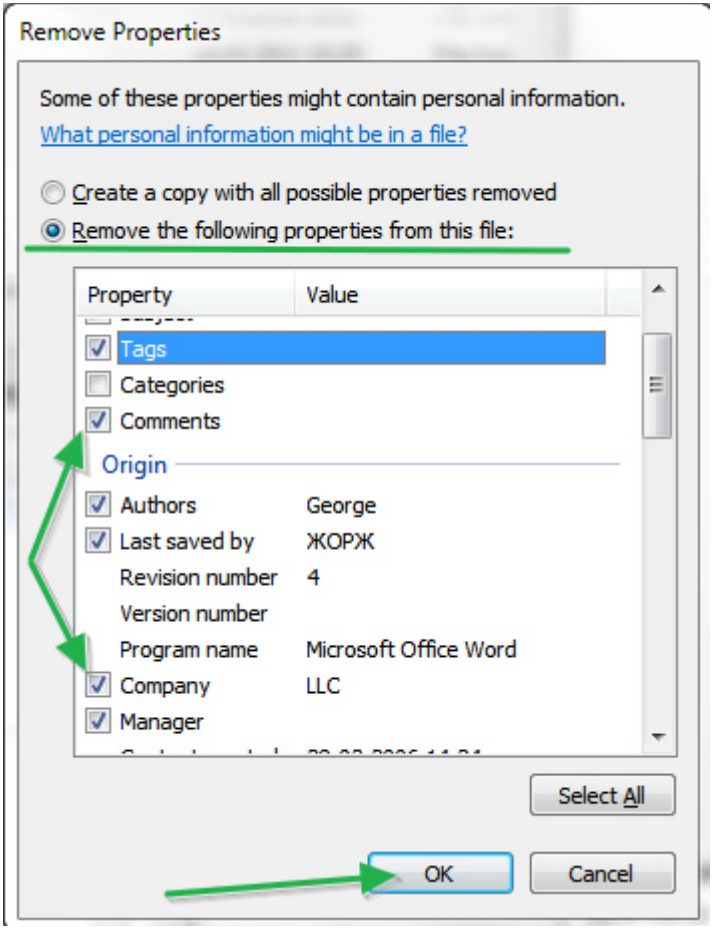
Рис. 2.9. Отключение куки в популярном браузере Mozilla Firefox (версия 18.0)

Стоить помнить, что после отключении куки, браузер не будет запоминать логин и пароль для входа на сервис облачного хранения, поэтому придется вводить эти данные вручную при каждом посещении сервиса через web-интерфейс.

Защита метаданных. Под метаданными понимаются некоторые виды автоматически создаваемых в этих файлах служебных записей, которые характеризуют каким-либо образом эти файлы. К метаданным относятся, например, EXIF-данные, автоматически добавляемые современной фото- и видеоаппаратурой в создаваемые файлы с медиаданными (например, JPEG).

Защитить метаданные при предоставлении общего доступа можно только путем их удаления из файлов перед загрузкой на сервис облачного хранения. Это можно сделать как средствами операционной системы, так и при помощи специального программного обеспечения.

Удаление метаданных средствами операционной системы. Для просмотра метаданных файла в Windows 7 необходимо зайти в контекстное меню файла (нажать на нём правой кнопкой мыши), перейти на вкладку "**Подробно**" и выбрать "**Удаление свойств и личной информации**" (указано стрелкой слева на рис. 2.10).



Для того, чтобы удалить персональную информацию нужно в появившемся окне (справа на рис. 2.10) выбрать режим "**Удалить следующие свойства для этого файла**", отметить галочками пункты свойств, которые необходимо удалить и нажать кнопку "**ОК**".

Удаление данных EXIF-данных. Существует специальные программы для удаления мета-данных из изображений и других медиафайлов. Одной из таких программ является бесплатная программа MetaStripper (http://www.photothumb.com/metastripper/),

которая удаляет EXIF-информацию из изображений, сохраненных в формате JPEG.

Программа очень проста в использовании: после запуска необходимо указать исходный каталог с изображениями и каталог, куда будут сохранены изображения без метаданных EXIF (рис. 2.11):

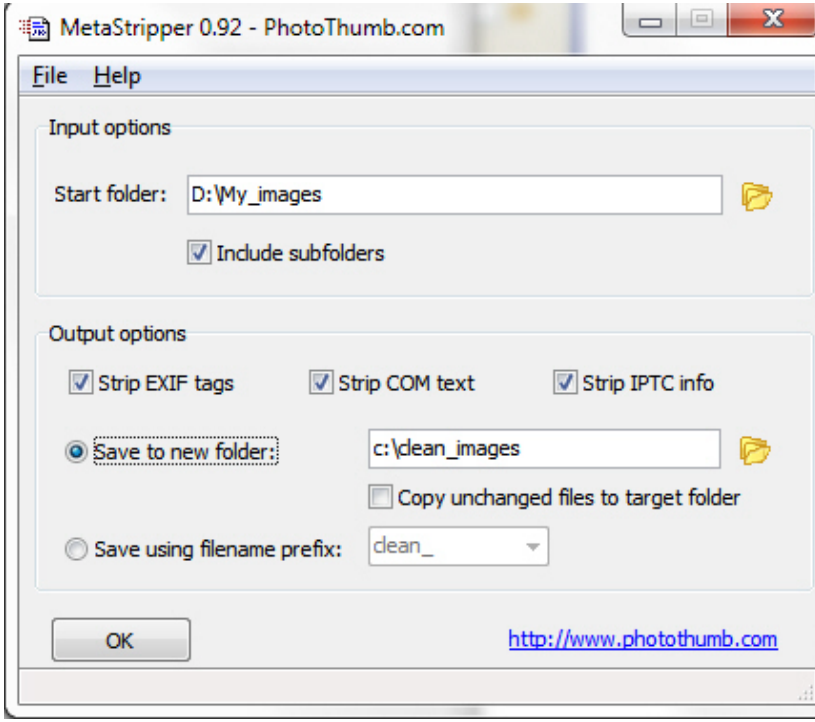


Рис. 2.11. Окно настроек программы MetaStripper

Единственный недостаток программы в том, что она работает только под управлением операционной системы Windows. В операционной системе Ubuntu следует пользоваться программным средством jhead (http://manpages.ubuntu.com/manpages/intrepid/man1/jhead.1.html). Для установки этого средства необходимо открыть окно терминала и в командной строке написать (если используется ОС Ubuntu):

```
sudo apt-get install jhead
```

Чтобы полностью удалить информацию **EXIF** из определенной картинки необходимо выполнить в командной строке команду:

```
jhead -purejpg /path_to_image.jpg
```

В этой строке "path_to_image.jpg" обозначает путь к файлу, из которого необходимо удалить метаданные. В качестве пути к файлам можно указать шаблон, например "/usr/home/Pictures /*.jpg", тогда exif-информация будет удалена из всех файлов с расширением .jpg в папке "**/usr/home/Pictures/*.jpg**".

Защита доступа к закрытым данным. Надежным способом защиты информации при размещении в облачном хранилище является применение средств шифрования. При шифровании данных исходная информация, которую необходимо защитить от посторонних, преобразуется специальным образом с использованием одного или нескольких криптографических алгоритмов.

Будучи зашифрованной, эта информация может быть прочитана или изменена, только если пользователь, которому предназначается информация, знает правильный пароль, при помощи которого он сможет расшифровать эту информацию. Для целей шифрования информации рекомендуется использовать бесплатное открытое программное обеспечение TrueCrypt (http://www.truecrypt.org/).

TrueCrypt позволяет создать виртуальный зашифрованный логический диск, хранящийся в виде обычного файла с любым расширением (файл-контейнер). Этот файл можно опубликовать на сервисе облачного хранения и предоставить к нему доступ тем пользователям, с которыми необходимо поделиться конфиденциальной информацией.

Достоинства TrueCrypt. TrueCrypt является кросс-платформенной программой и поддерживается такими операционными системами как Windows 7/Vista/XP/2000, Linux, Mac OS X. TrueCrypt является свободным программным обеспечением с открытым исходным кодом, поэтому его можно рекомендовать к применению в образовательных учреждениях. Также TrueCrypt поддерживает несколько криптографических алгоритмов, позволяет осуществлять каскадное

шифрование файлов, а также предлагает возможность правдоподобного отрицания. Рассмотрим эти достоинства подробно.

TrueCrypt поддерживает следующие криптографические алгоритмы блочного шифрования: AES, Twofish, Serpent (рис. 2.12). Эти алгоритмы являются победителями открытого конкурса AES, организованного NIST (https://en.wikipedia.org/wiki/Advanced_Encryption_Standard_process), и гарантируют хороший уровень стойкости к взлому при использовании надежных паролей. Алгоритм AES (Rijndael), который является победителем конкурса AES, принят в качестве стандарта Агентства национальной безопасности США для защиты сведений, составляющих государственную тайну (https://en.wikipedia.org/wiki/Rijndael). Алгоритмы Twofish и Serpent также являются надежными алгоритмами шифрования – в настоящее время не было обнаружено каких-либо реально реализуемых атак на них http://www.ixbt.com/soft/alg-encryption-aes-2.shtml)

Рис. 2.12. Выбор алгоритмов шифрования в TrueCrypt

Для шифрования сведений уровня "**Совершенно секретно**" (Top Secret) рекомендуется использовать длину ключа шифрования в 256 бит. Для уровня "**Секретно**" (Secret) используются ключи длиной 128 и

192 бита. Стандарт AES поддерживает эти три длины ключа шифрования (128/192/256) и поддерживает размер блока в 128 бит. Это означает, что перед шифрованием информация разбивается на блоки размером 128 бит, каждый из которых шифруется с использованием ключа шифрования.

Каскадное шифрование. Помимо поддержки криптостойких алгоритмов шифрования, TrueCrypt поддерживает каскадное шифрование (http://www.truecrypt.org/docs/cascades) различными шифрами. Например, данные можно зашифровать каскадом из сразу трёх шифров **AES+Twofish+Serpent**. В этом случае каждый из 128-битных блоков исходной информации будет зашифрован сначала при помощи алгоритма AES (с длиной ключа 256 бит), затем при помощи Twofish (с длиной ключа 256 бит) и в конце – при помощи алгоритма Serpent (с такой же длиной ключа в 256 бит). Каждый из каскадных шифров использует свой собственный ключ. Все ключи шифрования абсолютно независимы друг от друга, даже не смотря на то, что формируются они на основе одного пароля.

Другие виды каскадов, которые поддерживает TrueCrypt: AES-Twofish, Serpent-AES, Serpent-Twofish-AES, Twofish-Serpent. Очевидно, что использование каскадного шифрования значительно увеличивает уровень защиты зашифрованной информации.

Правдоподобное отрицание. TrueCrypt позволяет использовать правдоподобное отрицание (Plausible Deniability), если кто-то принуждает пользователя сообщить пароль от файла-контейнера. Это становится возможным благодаря двум особенностям (http://www.truecrypt.org/docs/plausible-deniability):

- Возможность создания скрытого тома внутри файла-контейнера, для доступа к которому необходимо указать отдельный пароль. Все данные, которые необходимо уберечь от посторонних глаз, хранятся на скрытом томе, а данные, которые пользователь сможет выдать в случае шантажа – на обычном томе. Таким образом, в случае шантажа пользователь может сообщить злоумышленнику пароль от обычного тома и ценная информация при этом не будет раскрыта.
- Невозможность идентификации томов TrueCrypt. Файл-контейнер TrueCrypt невозможно отличить от набора случайных данных (не содержит никаких сигнатур), то есть

файл нельзя ассоциировать с TrueCrypt как с программой, его создавшей.

Использование TrueCrypt с сервисами облачного хранения данных. Покажем на примере, как можно использовать программу TrueCrypt для защиты данных при работе с системами облачного хранения данных. Для создания зашифрованного файла-контейнера нужно в программе TrueCrypt открыть меню "**Тома**", выбрать пункт "**Создать новый том**", в появившемся окне отметить пункт "**Создать зашифрованный файл-контейнер**" (рис. 2.13).

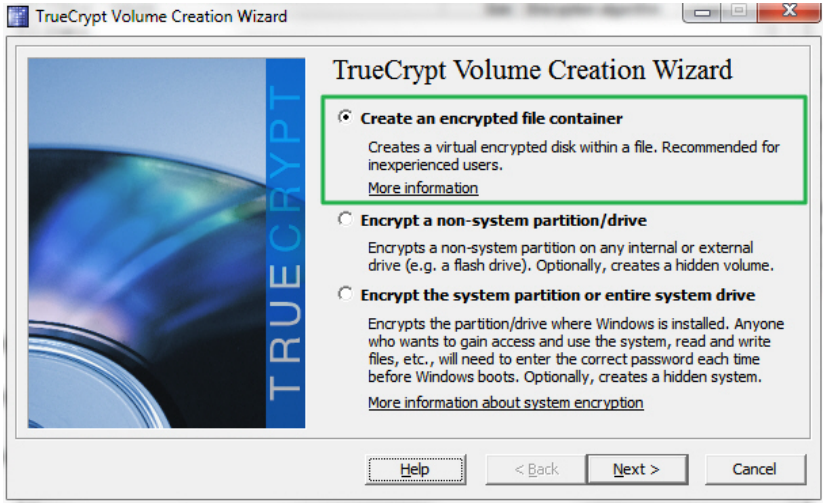


Рис. 2.13. Мастер создания томов TrueCrypt

На следующем шаге необходимо выбрать вид тома, который необходимо создать: стандартный или скрытый (рис. 2.14). Далее необходимо указать путь к новому файлу, в котором будет храниться контейнер с зашифрованной информацией. Для использования в облачных сервисах хранения необходимо указать путь к файлу в стандартном каталоге облачного сервиса, например для сервиса Dropbox это стандартная папка "**Dropbox**".

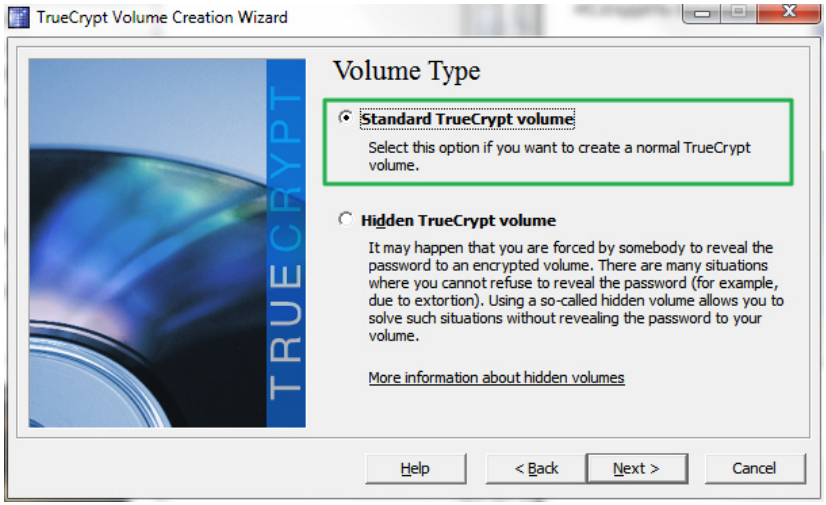


Рис. 2.14. Выбор типа тома TrueCrypt

Имя файла и его расширение могут быть произвольными. В целях повышения конфиденциальности лучше придумать нейтральное имя файла (например, "students_manual.pdf", "course.iso" и т.д.), чтобы оно не вызывало большого интереса у потенциальных злоумышленников. Для дополнительного увеличения уровня безопасности необходимо поставить галочку "**Никогда не сохранять историю**", см. рис. 2.15.

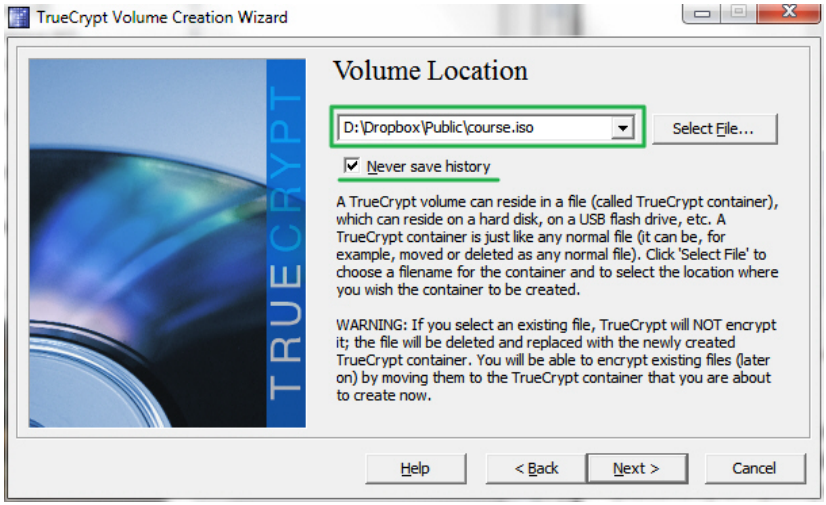


Рис. 2.15. Выбор файла-контейнера

На следующем этапе следует выбрать алгоритм шифрования и алгоритм хеш-функции, которые будут использоваться при создании тома. Стоит отметить, что от выбора алгоритмов шифрования зависит скорость работы с файлом-контейнером. Если используется каскад алгоритмов, скорость работы будет ниже, чем при использовании одного алгоритма. Оценить производительность системы можно при нажатии кнопки "**Оценить производительность**" ("**Benchmark**"), см. рис. 2.16.

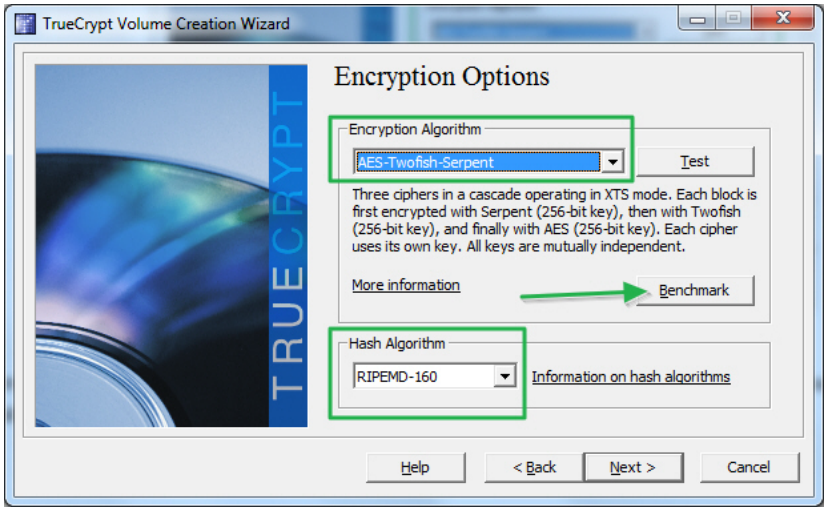
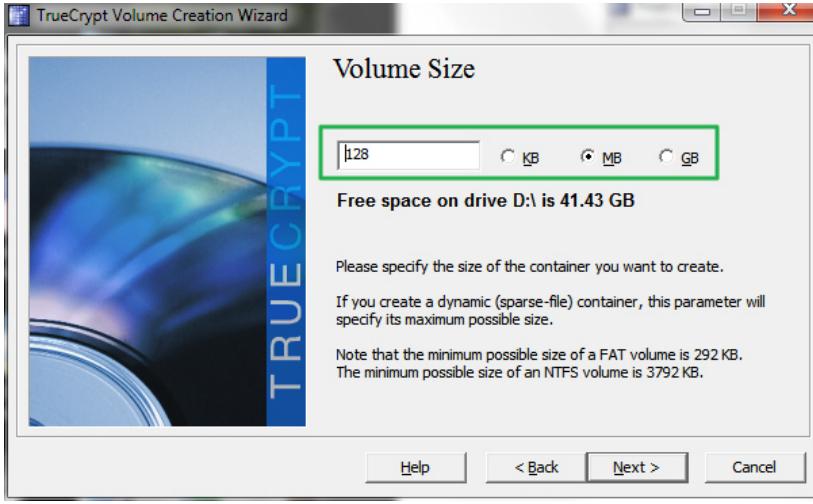
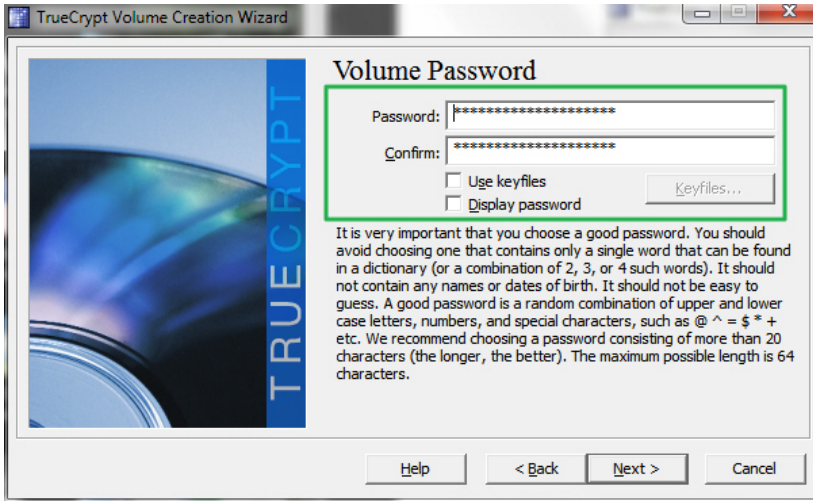


Рис. 2.16. Выбор опций шифрования в TrueCrypt

Затем необходимо указать размер тома и придумать надежный пароль для доступа к содержимому файла-контейнера. Рекомендации по созданию надежного пароля следующие: он не должен содержать слов из словаря, общеизвестных данных пользователя и представлять собой случайную комбинацию из букв, цифр и специальных символов в верхнем и нижнем регистрах. Длина пароля не должна быть меньше 20-ти символов, см. рис. 2.17.





Сразу после вода пароля мастер предложит выбрать тип файловой системы и отформатирует том. Заметим, что хотя на приведённом рис. 2.18 TrueCrypt позволяет выбрать лишь файловые системы FAT и NTFS, однако это не означает, что нельзя использовать в качестве типа файловой системы один из популярных типов ОС Linux. Указанное ограничение связано с тем, что программа TrueCript случае запускалась автором на ОС Windows.

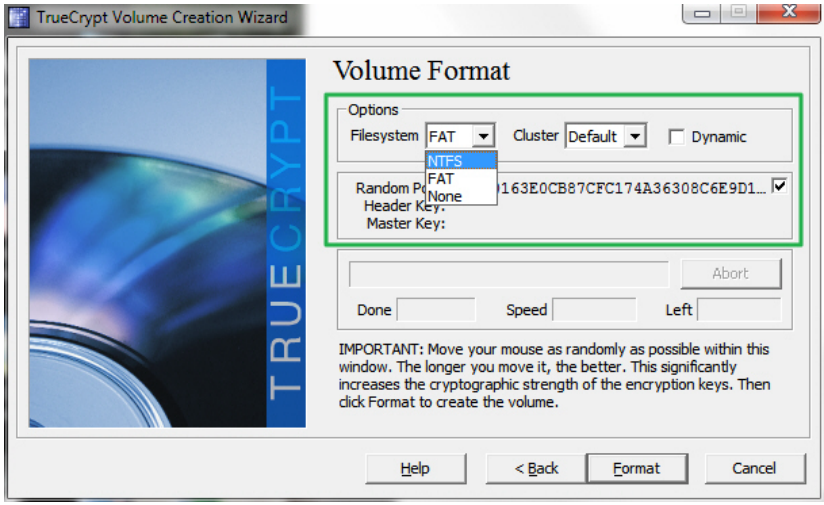


Рис. 2.18. Параметры форматирования тома TrueCrypt

После окончания процесса форматирования зашифрованный контейнер готов к использованию.

Работа с зашифрованным файлом-контейнером. Схема работы с зашифрованным файлом-контейнером очень проста: файл-контейнер монтируется как виртуальный диск через TrueCrypt, после чего с ним можно работать так же, как с обычным диском (или USB флеш-накопителем). Для монтирования нужно выполнить действия, указанные на рис. 2.19).

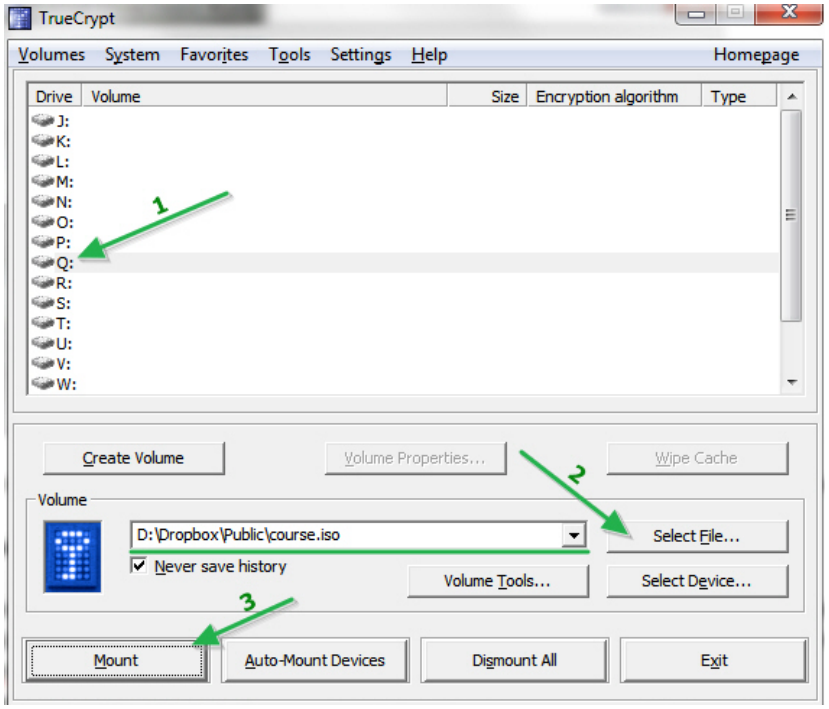


Рис. 2.19. Монтирование зашифрованного файла-контейнера

Перечислим эти действия с подробным описанием:

- выбрать букву диска, к которому необходимо подключить виртуальный зашифрованный диск;
- указать путь к зашифрованному файлу-контейнеру;
- нажать кнопку "**Смонтировать**" для монтирования контейнера и ввести правильный пароль для доступа к зашифрованной информации.

После ввода пароля файл будет примонтирован в качестве виртуального диска и будет доступен для дальнейшей работы (рис. 2.20).

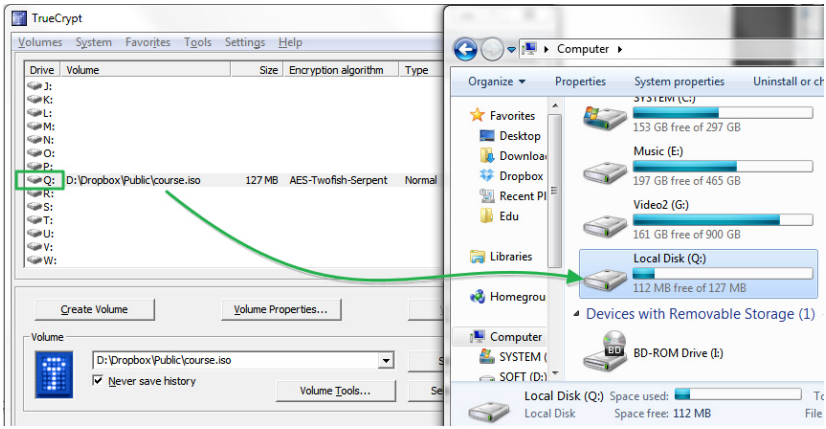


Рис. 2.20. Зашифрованный файл-контейнер доступен для работы

Шифрование данных при работе с виртуальным диском осуществляется "на лету". Таким образом, не нужно совершать никаких дополнительных действий для шифрования/дешифрования информации. Единственное, что необходимо сделать по окончании работы с защищаемыми данными, находящимися на виртуальном диске – демонтировать виртуальный диск. Для этого нужно (рис. 2.21):

- указать в главном окне программы букву диска, который необходимо демонтировать;
- нажать на кнопку "**Демонтировать**"

Стоит особо отметить, что перед процедурой демонтирования виртуального диска необходимо закрыть все программы, которые использовали данные на этом диске. В противном случае возможно повреждение данных.

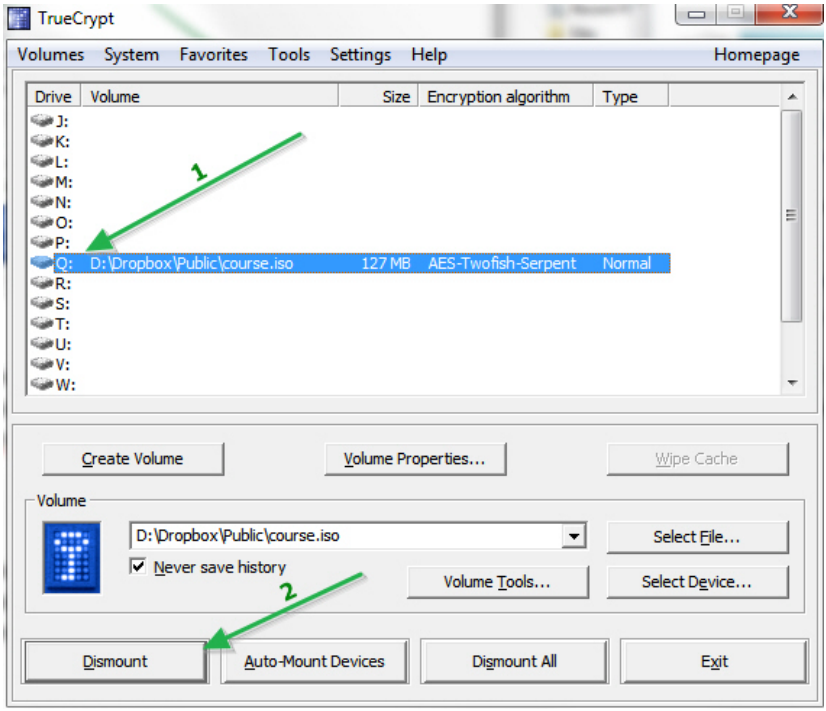


Рис. 2.21. Демонтирование виртуального диска с зашифрованной информацией

Хранение зашифрованного файла-контейнера в "облаке". Если файл-контейнер с зашифрованной информацией находится в каталоге, который синхронизируется с сервисом облачного хранения, то сразу после демонтирования этот файл будет синхронизирован с облаком. До демонтирования файл-контейнер блокируется TrueCrypt-ом, что делает синхронизацию невозможной.

Поскольку TrueCrypt шифрует файл блоками по 128 бит, то синхронизироваться будут только те блоки, которые изменились в результате шифрования информации. Сервисы облачного хранения синхронизируют не файлы целиком, а только изменившиеся блоки, что дает значительный выигрыш по скорости передачи данных по сравнению с передачей целых файлов.

Необходимо принять во внимание, что совместная работа с файлом-контейнером возможна только в последовательном режиме. Это подразумевает обязательное завершение работы с контейнером одного пользователя (демонтирование виртуального диска в TrueCrypt) и синхронизацию с облаком перед началом работы с контейнером другого пользователя или с другого устройства. При одновременной работе двух пользователей Dropbox с одним файлом-контейнером создаст две версии этого файла и добавит к имени одного из них пометку "**device conflicted copy date**", где "**device**" - имя устройства, на котором обнаружена конфликтующая копия файла, а "**date**" - дата обнаружения конфликта (см. рис. 2.22).

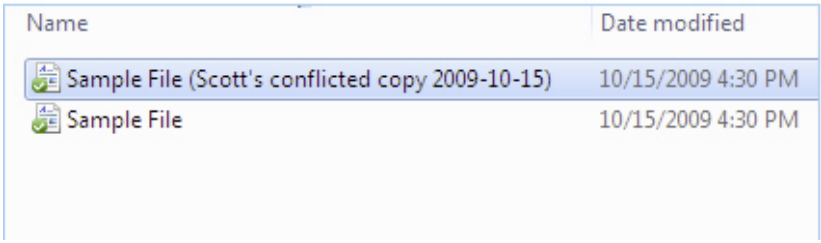


Рис. 2.22. Конфликтующие версии файлов в сервисе Dropbox

В этом случае придется провести дополнительную работу по поиску актуальной версии файла-контейнера и синхронизации данных между двумя версиями файла-контейнера.

Вывод – использование сочетания мер по защите персональных данных, метаданных и современных средств шифрования позволяет надежно защитить важную информацию от посторонних глаз при использовании сервисов облачного хранения.

11.2.4. Технология "Google Apps for Education"

Google Apps for Education (**GAfE – это набор облачных приложений, которые предоставляются компанией Google бесплатно для образовательных учреждений в рамках выбранного образовательным учреждением домена**. По данным на январь 2013 Google Apps для учебных заведений используют более 14 миллионов студентов и преподавателей.

История Google Apps берет начало в феврале 2006 года, когда был впервые представлен сервис электронной почты для использования с собственным доменным именем организации. Приложения Google для учебных заведений стали доступны в октябре 2006 и объединяли в себе, помимо почтовой службы, приложение для чата Google Talk, сервис Calendar и сервис создания веб-страниц Google Page Creator. По состоянию на январь 2013 года приложения Google Apps включают в себя инструменты для коммуникации и планирования, инструменты для коллективной работы, инструменты для повышения эффективности работы. Рассмотрим подробнее каждый из этих инструментов.

Инструменты для коммуникации и планирования:

- Электронная почта "**Gmail**" с размером почтового ящика в 25 ГБ на одного сотрудника/студента образовательного учреждения.
- Служба "**Google Talk**" с возможностью обмена текстовыми, аудио- и видео сообщениями, которые удобно использовать для, например, ведения удалённых лекционных занятий.
- Инструмент "**Календарь**" для управления расписаниями занятий, встреч, рабочим временем с возможностью ведения общего календаря между сотрудниками.

Инструменты для коллективной работы:

- Сервис "**Документы**", позволяющий создавать текстовые документы, таблицы, презентации и специальные опросные формы.
- Сервис "**Диск**" для безопасного хранения любых документов в "облаке" с возможностью общего доступа с бесплатным объемом 5 ГБ дискового пространства на одного пользователя.
- Сервис "**Сайты**" позволяет создавать веб-сайты для учащихся и преподавателей на основе шаблонов.
- Группы Google для создания списков рассылок и предоставления коллегам общего доступа к документам, сайтам и календарям.
- "**Видео для учебных заведений**" – сервис, позволяющий использовать учебные видео материалы в процессе обучения.

Инструменты для повышения эффективности работы:

- Служба поиска по письмам, сообщениям, документам.
- Единая централизованная консоль администрирования сервисов Google образовательного учреждения.

Все вышеперечисленные инструменты являются бесплатными при использовании в образовательных учреждениях. При этом Google Apps for education обладают следующими характерными особенностями:

- Осуществляется полное резервное копирование всех данных. Например, если сломается компьютер преподавателя или студента, работу можно будет продолжить через несколько секунд с другого устройства, на котором есть доступ к сети Интернет.
- Предоставляется надежное шифрование и безопасная аутентификация. Все данные при передаче между серверами Google и клиентскими устройствами шифруются с помощью протокола SSL.
- Обеспечивается высокий коэффициент готовности сервисов. Компания Google гарантирует работоспособность служб в течение 99,9% времени.
- Гарантируется соблюдение авторских прав и защита конфиденциальности данных ОУ. Политика конфиденциальности гарантирует, что компания Google не может публиковать или использовать не по назначению личную информацию, размещенную в облачных приложениях Google. Образовательное учреждение является единственным владельцем своих данных и полностью управляет ими.
- Возможность использовать ОУ собственного доменного имени при работе с Google Apps. Для каждого сотрудника учебного заведения регистрируется учетная запись электронной почты в виде **name@youruniversity.com**.

Чтобы начать использовать Google Apps для образовательных учреждений, необходимо проделать несколько шагов:

- Зарегистрировать образовательное учреждение.
- Подать заявку на подключение сервисов GAfE.

- Настроить GAfE в соответствии с особенностями учебного заведения.

Регистрация ОУ в GAfE. Для регистрации необходимо перейти по ссылке https://www.google.com/a/signup/?enterprise_product=GOOGLE.EDU и следовать инструкциям мастера регистрации, который предложит заполнить ряд стандартных форм (рис. 2.23).

В процессе регистрации Google Apps for Education потребуется подтвердить право собственности на доменное имя учебного заведения или зарегистрировать новое доменное имя. В связи с этим рекомендуется воспользоваться услугами соответствующего ИТ-специалиста, ответственного за регистрацию доменного имени вашей организации.

Если доменное имя у учебного заведения отсутствует, то его можно приобрести у любого поставщика интернет-услуг. В среднем абонентская плата за использование доменного имени составляет 8 долларов США за год использования. При этом следует учитывать, что процесс регистрации нового доменного имени может потребовать вплоть до 24 часов в связи с обновлением информации на корневых серверах DNS.

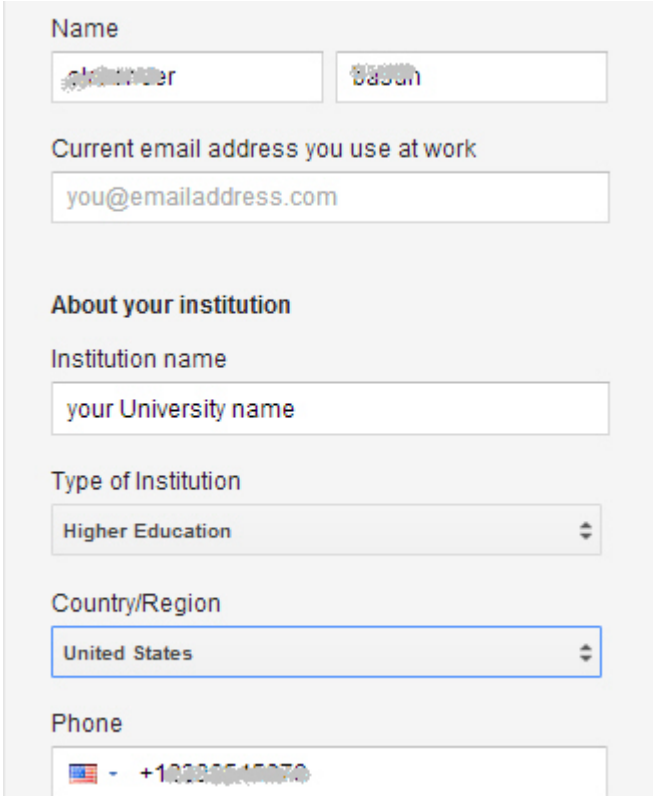


Рис. 2.23. Первый этап мастера регистрации Google Apps for Education

Заявка на подключение сервисов GAfE. После регистрации в Google Apps будет создана учетная запись с правами суперадминистратора с реквизитами входа, которые были указаны в процессе первичной регистрации. Необходимо зайти в Google Apps в режим управления доменом с правами суперадминистратора и подать заявку на подключение вашему учебному заведению бесплатных сервисов Google Apps for Education. **Важно заметить, что процесс рассмотрения заявки компанией Google на подключение бесплатных сервисов занимает одну-две недели.**

Настройка GAfE для нужд ОУ. Для осуществления первоначальной настройки и ознакомления с материалами по администрированию

Google Apps вашего учебного заведения необходимо выполнить все шаги мастера настройки. Если мастер первоначальной настройки не запустился автоматически при первом входе в систему управления доменом, необходимо сделать это вручную, используя главное меню Setup (рис. 2.24). В процессе выполнения мастер настроек поможет зарегистрировать учетные записи пользователей образовательного учреждения и настроить для них службы Google Apps. Этот процесс может потрбовать существенных временных затрат при большом количестве пользователей.

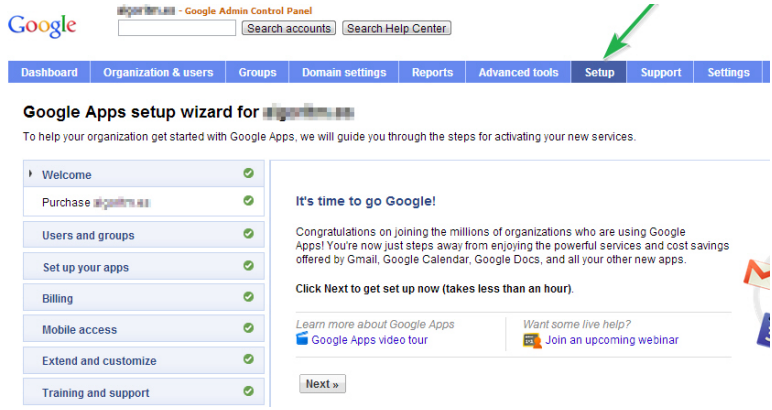


Рис. 2.24. Мастер первоначальной настройки Google Apps

Далее в главе рассматриваются основные бесплатные службы и сервисы Google Apps for Education.

Сервис электронной почты Gmail. Этот сервис представляет собой хранилище электронной почты и инструменты для поиска, помогающие учащимся быстро искать нужную информацию и отправлять мгновенные сообщения прямо из своих учетных записей. **Максимальный размер почтового ящика для каждого пользователя учебного заведения составляет 25 ГБ.**

Максимальный размер одного почтового сообщения составляет 25 МБ. При использовании возможностей сервиса "Google Drive" можно загрузить информацию в облачное хранилище и отправить пользователю ссылку на эти файлы. В этом случае получатель сможет скачать файл размером до 10 ГБ.

Интерфейс Gmail интуитивно понятен и по основным функциям не отличается от других почтовых сервисов (рис. 2.25). Одним из отличий является возможность использования ярлыков и фильтров сообщений.

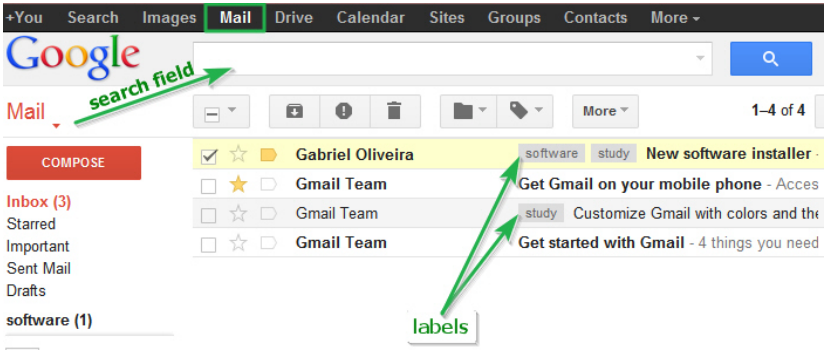


Рис. 2.25. Интерфейс Gmail. Использование ярлыков

Почтовый сервис Gmail достаточно эффективно борется со спамом и осуществляет антивирусную проверку всех почтовых сообщений. Еще одним большим плюсом Gmail в версии для образовательных учреждений является отсутствие рекламы.

Сервис сообщений Google Talk. С помощью сервиса сообщений учащиеся могут звонить своим знакомым и **отправлять им мгновенные сообщения бесплатно в любое время и в любой точке мира**.

Сервис сообщений Google Talk доступен из интерфейса Gmail. По умолчанию можно отправлять только текстовые сообщения в интерфейсе почтового ящика Gmail. Чтобы начать чат, нужно выбрать нужное имя из списка контактов и пригласить его к общению. После подтверждения откроется окно сообщений. Все указанные функции отмечены стрелками на рис. 2.26.

Для подключения голосовой и видео связи необходимо установить дополнительное приложение "Voice and video chat", которое доступно для всех популярных операционных систем (Windows XP и выше, Mac OS X 10.5 и выше и Linux). Для установки приложения нужно выбрать пункт меню "Add voice & video" в

меню изменения статуса пользователя. Для того, чтобы собеседник мог видеть и слышать сотрудника вашего учреждения, необходимо, чтобы компьютер сотрудника был оборудован веб-камерой и микрофоном.

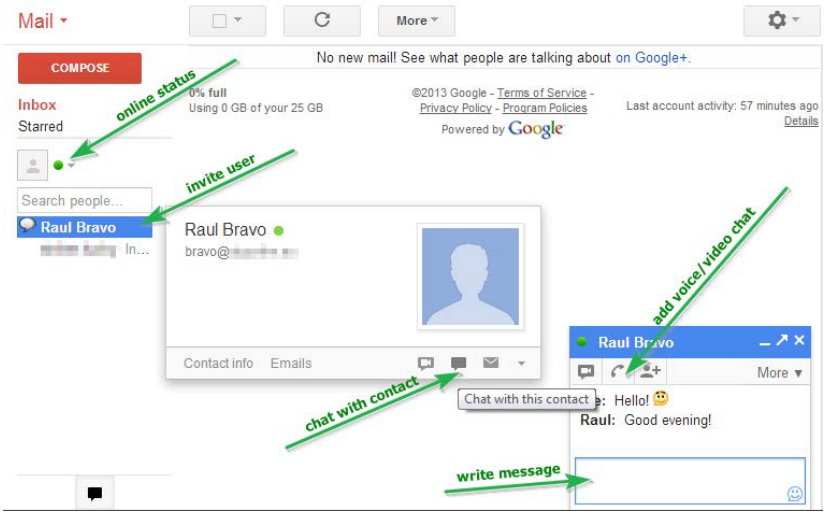


Рис. 2.26. Сервис сообщений в Gmail

Сервис "Календарь". Этот сервис поможет сотрудникам образовательного учреждения составлять расписания, планировать события в календарях, добавлять оповещения и напоминания о событиях, а также обмениваться этими календарями с коллегами. Для создания события в календаре необходимо выполнить следующие действия (подробно эти шаги представлены на рис. 2.27с указанием расположения соответствующих элементов управления пользовательского интерфейса):

- выбрать дату события;
- выбрать время события;
- ввести описание события и нажать кнопку "**Создать событие**".

Предоставление общего доступа к календарю. В некоторых случаях к календарю необходимо предоставить общий доступ другим сотрудникам. Например, руководителю образовательного учреждения

необходимо оценить загруженность преподавательского состава и спланировать общее собрание. Для такого рода целей сервис позволяет получить информацию о календарях других пользователей. Подробно процесс создания события в календаре описан на рис. 2.27.

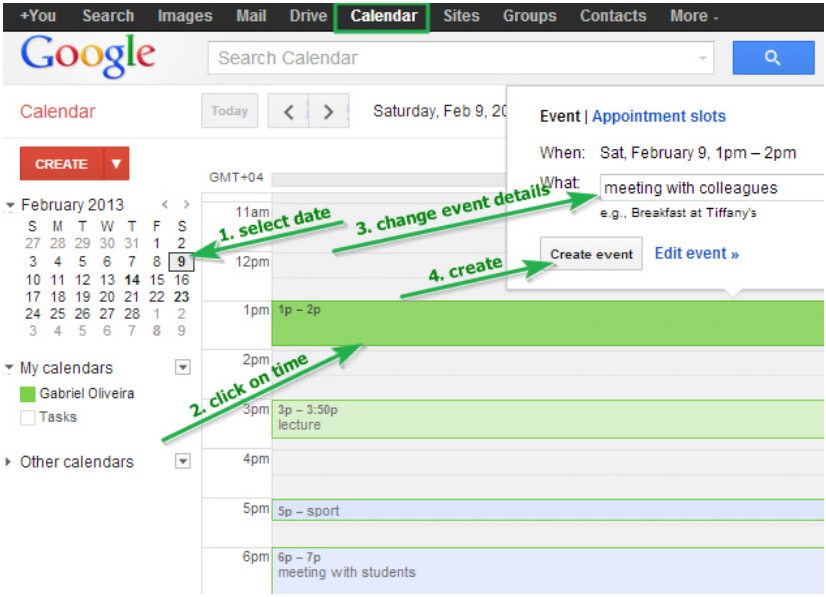


Рис. 2.27. Создание события в календаре Google

Для предоставления общего доступа к календарю необходимо перейти в режим настроек и установить галочку "**Share this calendar with others**" и выбрать режим предоставления общего доступа. Сервис предоставляет очень гибкие возможности по предоставлению общего доступа к календарю – можно настроить как индивидуальный, так и групповой доступ, а также установить определенные разрешения доступа к своему календарю (рис. 2.28).

Для просмотра общих календарей необходимо в меню "**Other calendars**" вести имя или адрес электронной почты сотрудника, к календарю которого необходимо подключиться. Если сотрудник предоставил общий доступ к своему календарю, то события отобразятся в текущем календаре. События из календарей других сотрудников будут подсвечены индивидуальными цветами. Видимость

календарей сотрудников можно изменить путем нажатия на цветной квадрат рядом с именем сотрудника. Как видно на рис. 2.29, события из календарей различных сотрудников отображаются в единой календарной сетке, но представлены разными цветами.

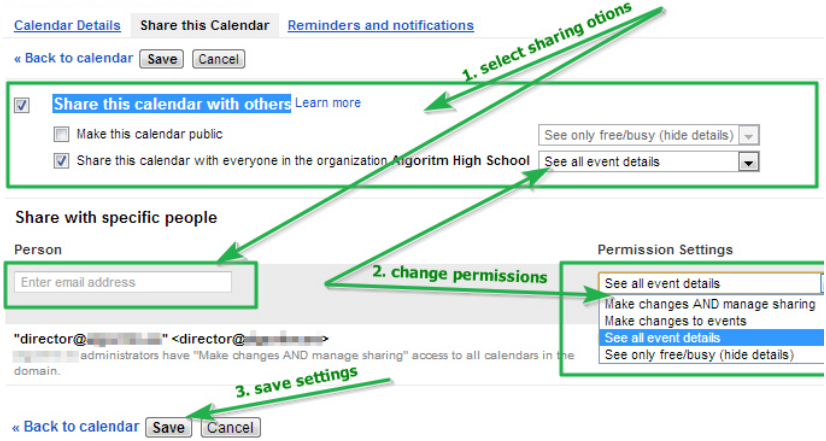


Рис. 2.28. Предоставление общего доступа к календарю

В сервисе "**Календарь**" реализована полезная функция добавления напоминаний. Есть два режима отправки напоминаний: посредством электронной почты и/или SMS. Настроить оповещения можно для каждого события индивидуально в режиме редактирования события или сразу для всех событий календаря в общих настройках (**Calendar Settings -> Calendars -> Reminders and notifications**). Настройка номера мобильного телефона для отправки SMS оповещений осуществляется в разделе "**Mobile setup**" настроек календаря.

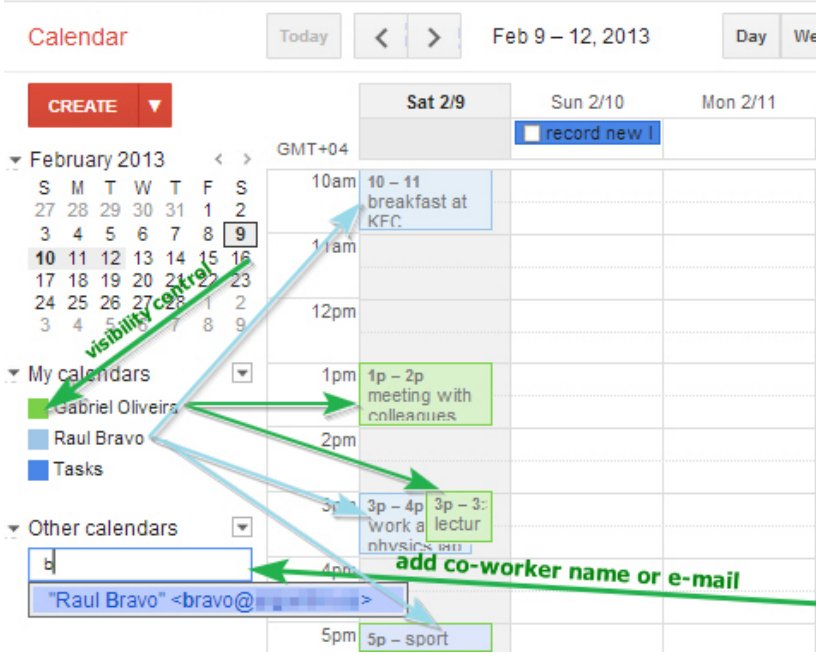


Рис. 2.29. Пример общего календаря Google Apps

Организация совместных мероприятий (например, семинарских занятий, выездных лабораторных работ и т.д). Отправка коллегам приглашений на мероприятия. Для организации общего мероприятия необходимо создать событие и в настройках добавить участников. Подробно процесс создания совместных мероприятий описан на рис. 2.30.

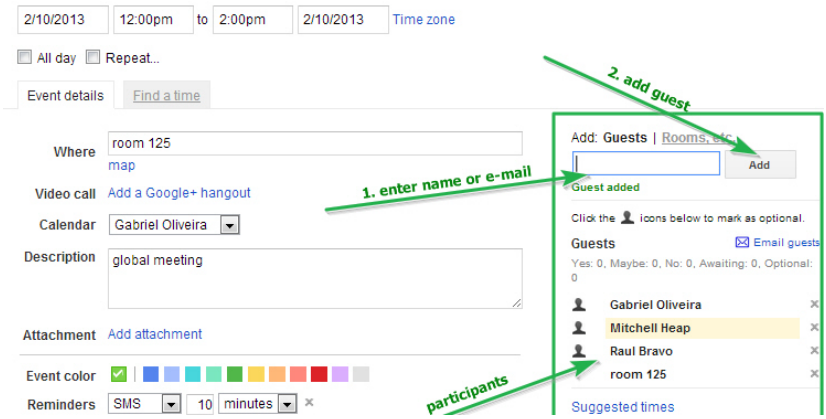


Рис. 2.30. Приглашение участников на мероприятие

Дополнительно можно указать место, где произойдет событие и даже увидеть это место на географической карте Google (в разделе настроек "**Where**" -> "**map**"). После сохранения события участники получат оповещение по электронной почте о предстоящем событии и смогут отправить ответ об участии. Сервис предоставляет возможность добавления комментариев к предстоящему событию.

Очевидно, что сервис "**Календарь**" – это очень удобное бесплатное средство личного планирования и организации совместной эффективной работы в коллективе одногруппников или коллег-педагогов. Его применение возможно при планировании мероприятий ОУ любого уровня: от аудиторных занятий, до открытых конференций, проходящих в ОУ.

Сервис "Диск". Сервис "**Диск**" позволяет организовать бесплатное облачное хранилище емкостью до 5 ГБ на каждого сотрудника образовательного учреждения, имеющего учетную запись в Google Apps for Education. Сервис предлагает два режима работы: 1) через веб-браузер и 2) через программу-клиент "Google Drive". Рассмотрим только первый режим, т.к. он позволяет осуществлять работу с сервисом "**Диск**" с любого компьютера, на котором установлен веб-браузер, не предъявляя возможно высокие требования к оборудованию и программному обеспечению, необходимому для корректной работы.

Для загрузки файлов в облачное хранилище через веб-браузер необходимо в разделе "**Drive**" Google Apps нажать кнопку "**Upload**" и выбрать режим загрузки (загрузка одного/нескольких файлов или каталог с файлами). Рис 2.31 предлагает подробную схему-инструкцию того, как именно можно загрузить новые файлы или папки в облачное хранилище.

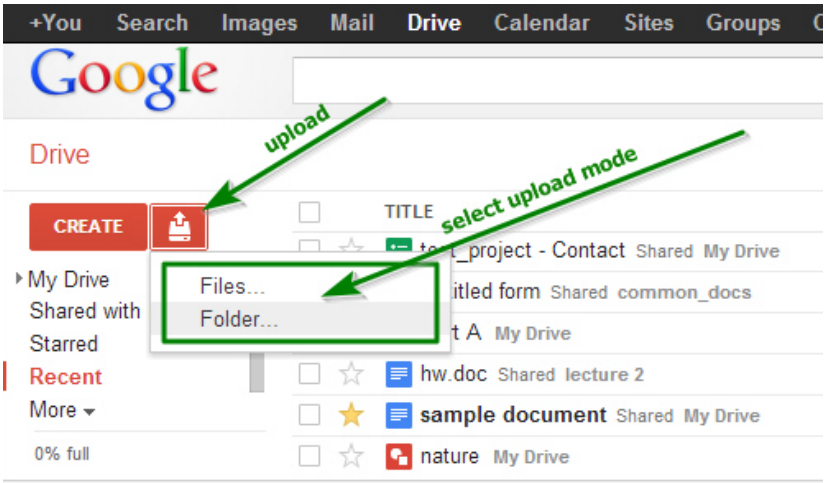


Рис. 2.31. Загрузка файлов в облачное хранилище Google Drive через веб-интерфейс

Время загрузки зависит от скорости Интернет-соединения и объема загружаемой информации. После окончания процесса передачи данных загруженные файлы отобразятся в общем списке. Работа с файлами в облаке не отличается от работы с файлами средствами любой операционной системы. Файлы можно просматривать, редактировать, переименовывать, удалять, предоставлять к ним общий доступ (не только коллегам из образовательного учреждения, но и публиковать в сети Интернет). Вопрос предоставления общего доступа подробно рассматривается в разделе, посвященном сервису "**Документы**". (см. рис. 2.32)

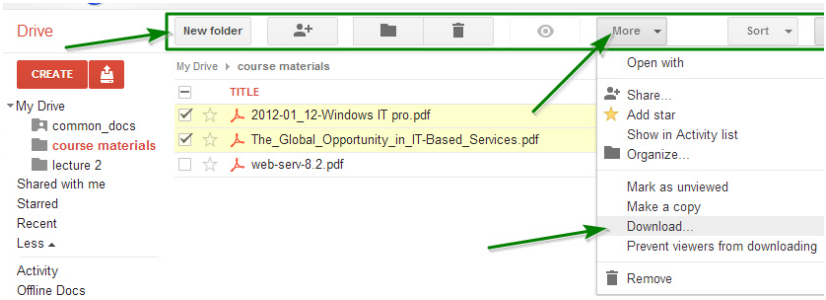


Рис. 2.32. Управление файлами через веб-интерфейс

Любой файл из облачного хранилища можно скачать на устройство пользователя через браузер. Для этого необходимо отметить галочками файлы/папки в списке "**My Drive**", которые необходимо скачать, после чего выбрать "**Download**" из меню "**More**".

Сервис "**Drive**" имеет ещё одну очень полезную функцию – преобразование файлов из одного формата в другой. Например, сервис может преобразовать электронную таблицу, в проприетарный формат .xlsx (Microsoft Office), в открытый формат .ods (Open Office) или в формат .pdf. Выполнение операции преобразования можно настроить при скачивании файлов в мастере "Convert and Download".

Подробный перечень форматов для конвертирования приведён на рис. 2.33, где также показан внешний вид элемента управления, использующегося для выполнения операции конвертирования.

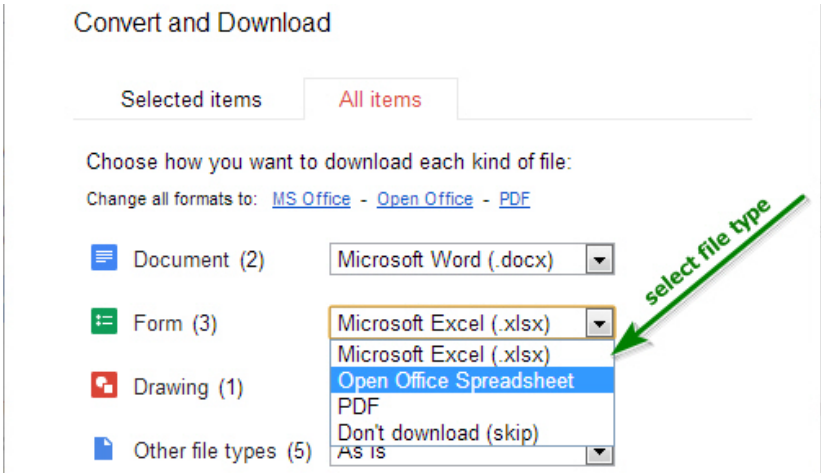


Рис. 2.33. Мастер преобразования форматов при скачивании файлов из облака

При использовании нескольких электронных устройств у пользователей (например, рабочий компьютер в образовательном учреждении и планшет – дома) очень часто возникает проблема синхронизации общей информации (конспектов, лекций, программного обеспечения и т.д.), хранимой на этих устройствах и поддержания её в актуальном состоянии. Для решения этой задачи рекомендуется использовать программное обеспечение "Google Drive". Установить это приложение можно из раздела "My Drive" сервиса "Drive" Google Apps for Education (см. рис. 2.34).

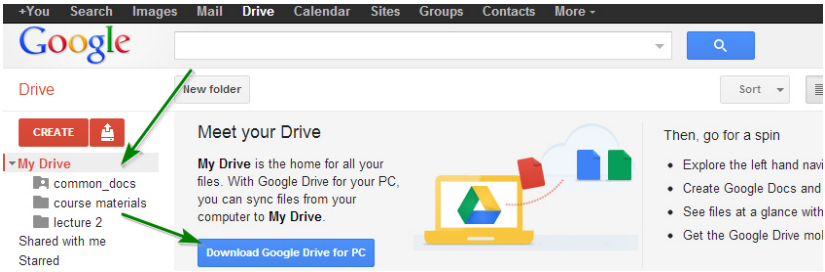


Рис. 2.34. Скачивание "Google Drive"

После скачивания и установки приложения на устройство пользователя, необходимо запустить приложение, используя учетную запись Google Apps for Education. Программа предложит указать каталог, содержимое которого будет синхронизироваться с облаком (см. рис. 2.35).

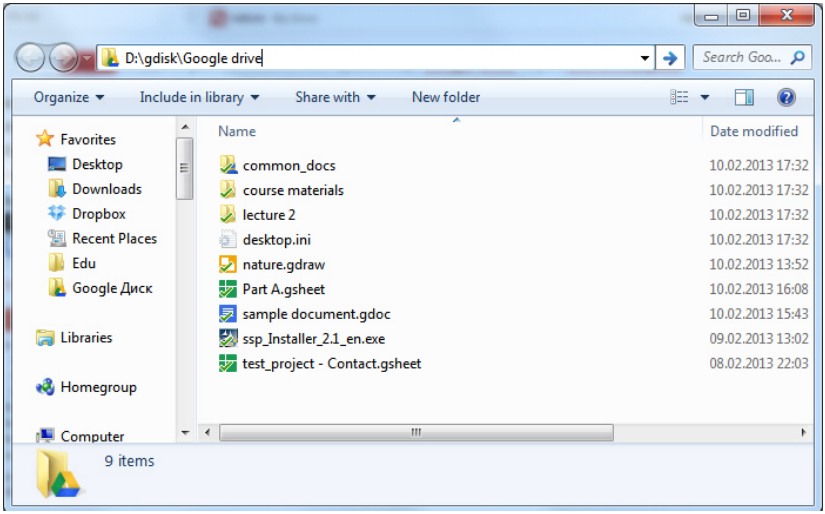


Рис. 2.35. Каталог Google Drive в операционной системе Windows 7, синхронизированный с облаком

Синхронизация происходит автоматически между всеми устройствами пользователя, на которых запущен клиент "Google Drive", и облачным сервисом при наличии доступа к сети Интернет.

Сервис "Документы". Сервис "**Документы**" позволяет создавать и редактировать текстовые документы, электронные таблицы и презентации непосредственно в веб-браузере. **Не требуется тратить время на установку и поддержку офисного программного обеспечения на компьютерах сотрудников.** При помощи сервиса "**Документы**" **в учебном заведении можно организовать совместную работу над документами в пределах группы или всего учебного заведения в режиме реального времени.** Помимо этого, окончательные версии документов можно публиковать для пользователей со всего мира. Дополнительно к этим функциям сервис "**Документы**" позволяет создавать формы для проведения опросов и

анкетирования среди учащихся. Инструмент "**Рисование**" поможет при создании графических схем и редактировании изображений.

Создание и редактирование документов. Работа с документами осуществляется через интерфейс Google Drive. Для создания нового документа необходимо нажать "**Create**" и выбрать тип создаваемого документа (см. рис. 2.36).

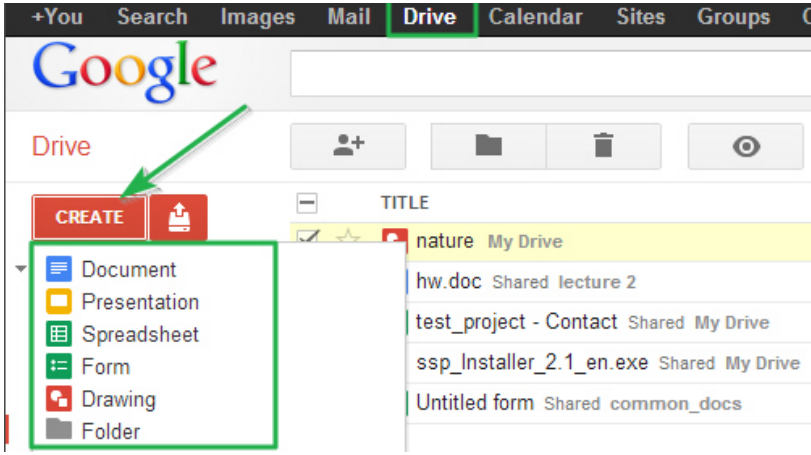


Рис. 2.36. Создание документов в сервисе Google Apps

Работа со всеми типами документов стандартизирована и является интуитивно понятной. Для редактирования текста, электронных таблиц и презентаций используются обычные инструменты редактирования, аналогичные инструментам любого офисного пакета (Microsoft Office, Open Office, Lotus Symphony и т.д.). Пример работы с текстовым документов приведён на рис. 2.37.

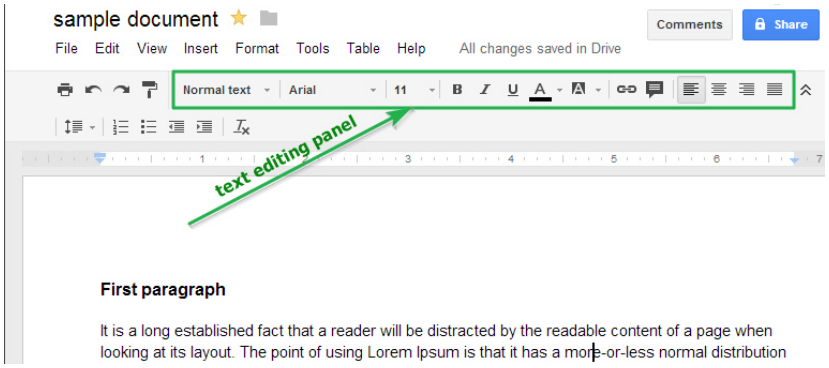


Рис. 2.37. Редактирование текстового документа

Сохранение документа происходит автоматически при внесении изменений в документ. Для каждого документа, который редактируется в облаке, ведется "**История версий**". Можно в считанные секунды вернуться к необходимой версии документа и продолжить редактирование документа с нужного момента.

Для просмотра версий документа необходимо в режиме редактирования документа выбрать "**All changes saved in Drive**". Для восстановления необходимой версии в окне "**Revision history**" нужно выбрать версию и нажать "**Restore this revision**". Указанная версия документа будет восстановлена. При коллективном редактировании документа правки, внесенные сотрудниками, подсвечиваются разными цветами (у каждого сотрудника – свой цвет, см. рис. 2.38).

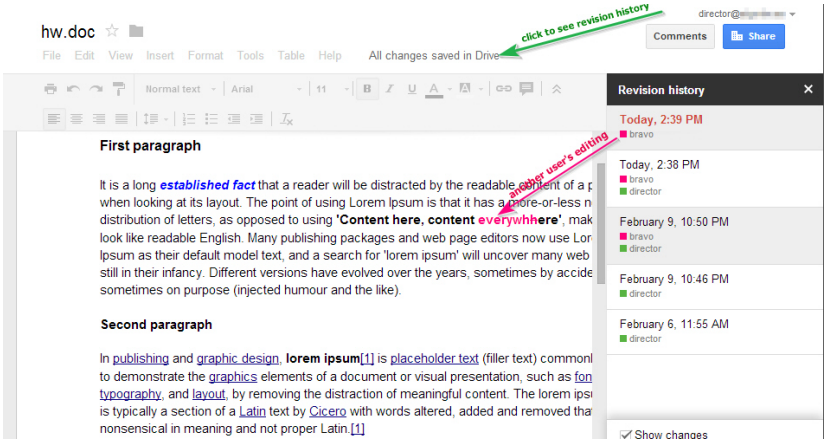


Рис. 2.38. История версий документа при коллективном редактировании

Коллективная работа с документами. Для организации возможности коллективной работы с документом необходимо нажать на кнопку "**Share**", находясь в режиме редактирования документа, либо из интерфейса "**Drive**" отметив галочками файлы, к которым необходимо предоставить коллективный доступ (см. рис. 2.39).

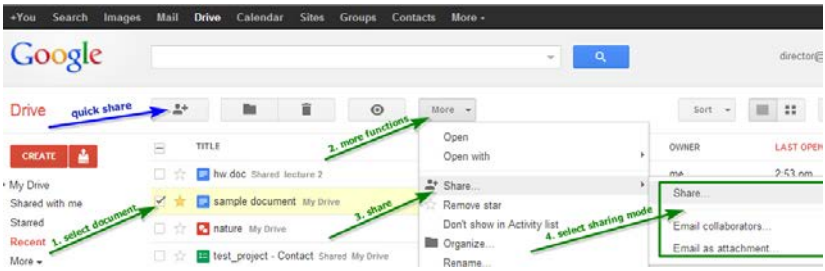


Рис. 2.39. Предоставление общего доступа к документам

После этого необходимо указать, кому и с какими правами необходимо предоставить доступ. Например, можно предоставить доступ к документу всем сотрудникам учебного заведения в режиме чтения. Тогда документ сможет увидеть каждый сотрудник учебного заведения в своей учетной записи Google Apps в разделе "**Drive**". Этот режим

полезен для публикации общих документов, например, положений, регламентов, учебных планов и т.д. (см. рис. 2.40).

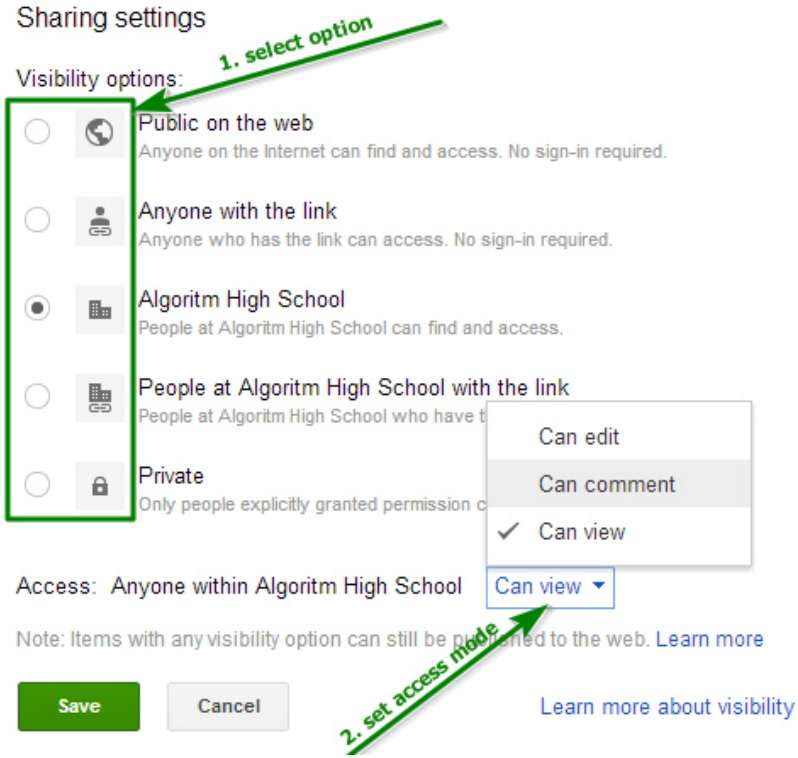


Рис. 2.40. Настройки общего доступа к файлам

Если требуется ограничить доступ к файлу в рамках определенной группы, то можно отправить избранным сотрудникам ссылку на общий файл. В этом случае доступ к информации получат только те пользователи, которые перейдут по специальной ссылке, которая будет автоматически направлена им по электронной почте (см. рис. 2.41).

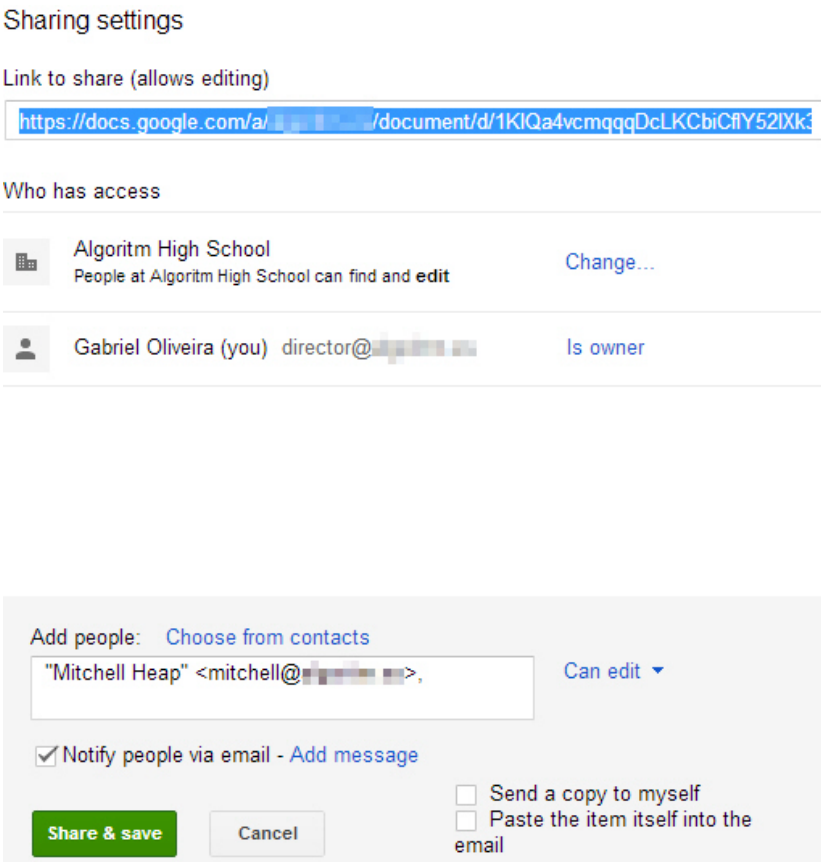


Рис. 2.41. Настройки общего доступа для выбранных пользователей

Сервис позволяет предоставить глобальный доступ к файлу через сеть Интернет любому пользователю. Для доступа к файлу достаточно просто перейти по ссылке. Регистрация в сервисе Google Apps при этом не требуется. Такой режим общего доступа может быть удобен при предоставлении информации будущим абитуриентам, у которых еще нет учетной записи в Google Apps for Education образовательного учреждения (например, список вопросов для подготовки к сдаче вступительных экзаменов). Дополнительным преимуществом сервиса "**Документы**" является встроенная проверка орфографии в редактируемых документах.

Создание опросов и проведение анкетирования. Еще одной полезной функцией сервиса "**Документы**" является возможность проведения опросов и анкетирования. Для создания опроса/анкеты необходимо создать новый документ с типом "**Form**" в разделе "**Disk**" (см. рис. 2.42).

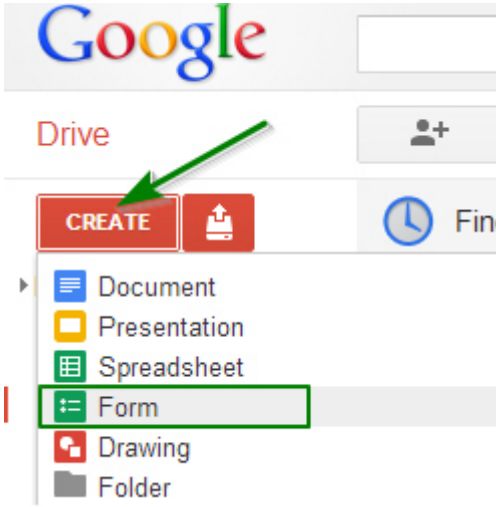


Рис. 2.42. Добавление новой формы

После конфигурирования анкеты (добавления вопросов), её нужно сохранить и предоставить общий доступ тем сотрудникам, среди которых необходимо провести опрос. Опрос может быть как анонимным, так и персонализированным (имя пользователя записывается при прохождении опроса/заполнении анкеты).

Результаты опроса/анкетирования можно посмотреть в виде графика или в виде таблицы в меню "**See responses**". Проведение опросов помогает улучшить качество образовательного процесса. Например, при помощи опросов можно узнать, понравился ли курс студентам, какие темы представляли наибольший интерес в процессе обучения и т.д. Посредством опросов можно принять верное решение по сложным вопросам, которое будет одобрено всем коллективом. Анкетирование поможет организовать сбор сведений о студентах/сотрудниках, например, при заселении в кампусе. Пример создания опросника можно видеть на рис. 2.43.

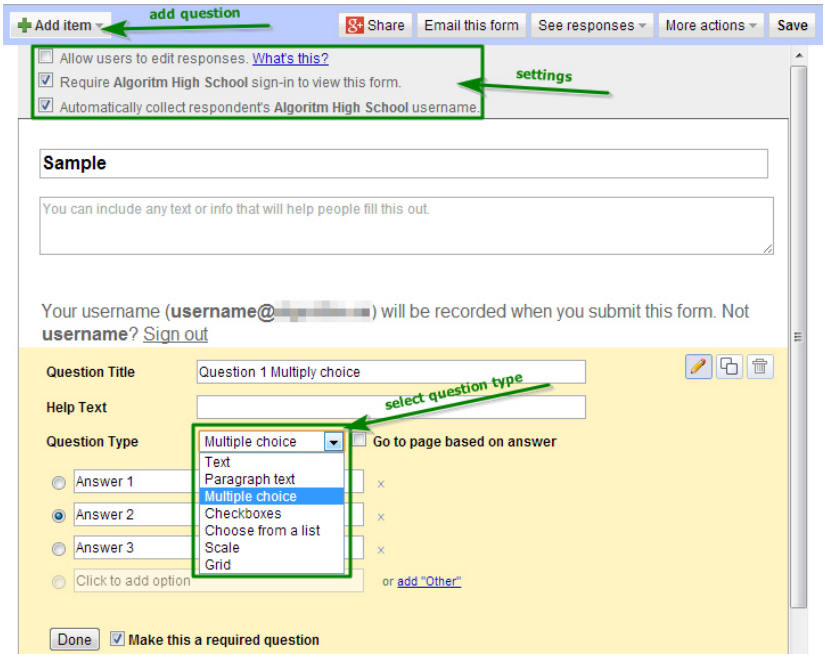


Рис. 2.43. Настройка анкеты/опроса

Сервис "Сайты". Этот сервис предназначен для создания мини-сайтов для совместной работы и централизованного хранения связанных между собой документов. В качестве примеров использования сервиса "**Сайты**" могут служить, например, сайт отдельного факультета/подразделения учебного заведения или wiki приоритетного научного проекта. Для создания собственных сайтов образовательному учреждению выделяется 10 ГБ дискового пространства в облаке, плюс 500 МБ для хранения файлов каждого сотрудника. Чтобы создать новый сайт, необходимо перейти в сервис "**Сайты**" и выбрать "**Create**". Проще всего создать сайт на основе готового шаблона, который можно выбрать из внушительного списка, представленного в галерее шаблонов.

На рис. 2.44 с помощью стрелок подробно показан процесс создания нового сайта из шаблона. Порядок выполнения действия показан нумерацией соответствующих стрелок.

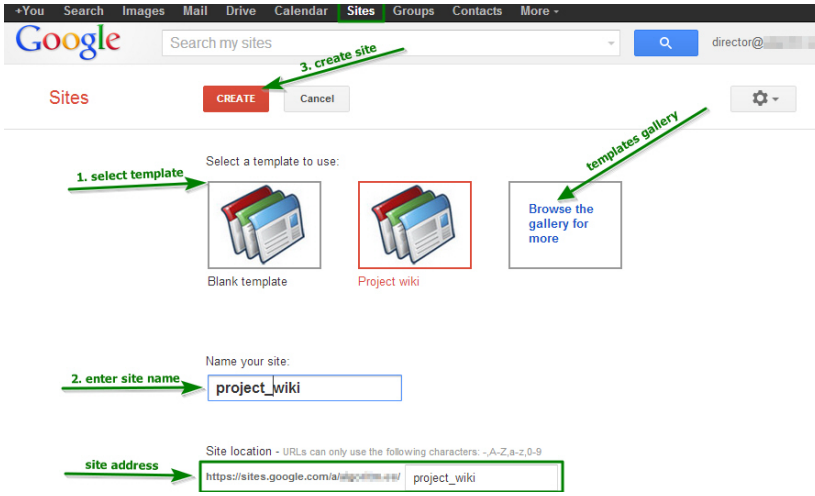


Рис. 2.44. Создание сайта на основе шаблона

После выбора шаблона и ввода названия сайта необходимо нажать "**Create**" для запуска процедуры генерации сайта. Генерация сайта на основе шаблона может занять одну-две минуты. Как только сайт будет создан, появится возможность редактировать существующие страницы и добавлять новые (в качестве подробного пример см. рис. 2.45).

увеличить изображение
Рис. 2.45. Интерфейс редактирования веб-страницы

Процесс редактирования веб-страниц не вызывает особых сложностей, поскольку используется тот же самый интерфейс, что и при

редактировании документов в сервисе "**Documents**". Предоставление общего доступа к сайту осуществляется так же, как и предоставление общего доступа к документам – необходимо выбрать пункт "**Share**" интерфейса, затем настроить видимость и права доступа к сайту (см. рис. 2.46).

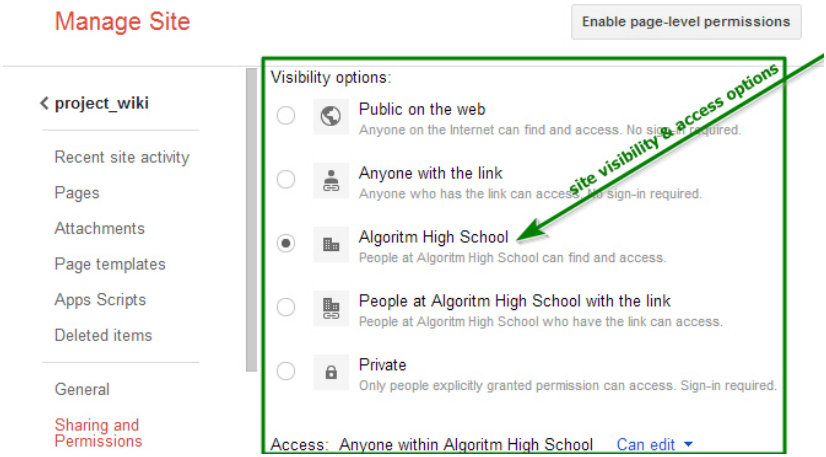


Рис. 2.46. Управление настройками сайта

Сервис "**Сайты**" позволяет быстро и просто создавать локальные сайты учебного заведения. Этот инструмент позволяет преподавателям сконцентрироваться на создании качественных учебных материалов, сокращая при этом время на их оформление и публикацию, что позволяет сделать процесс обучения более продуктивным.

Служба поиска. В настоящее время компания Google является мировым лидером по предоставлению поисковых услуг в сети Интернет. Поэтому во всех приложениях Google Apps присутствует инструмент поиска, будь то сервис электронной почты, календарь или документы. Поиск осуществляется не только по имени файла, электронного письма или события календаря, но и по содержимому этих объектов. Даже если ученик или преподаватель забыл, как называется тот или иной документ, найти его можно по одному или нескольким словам, которые содержатся в этом документе.

Как и ранее, пример того как выглядит интерфейс **GAfE**, использующийся при работе службы поиска, представлен на рис. 2.47. Активные элементы, использующиеся при поиске, а также результаты поиска отмечены стрелками.

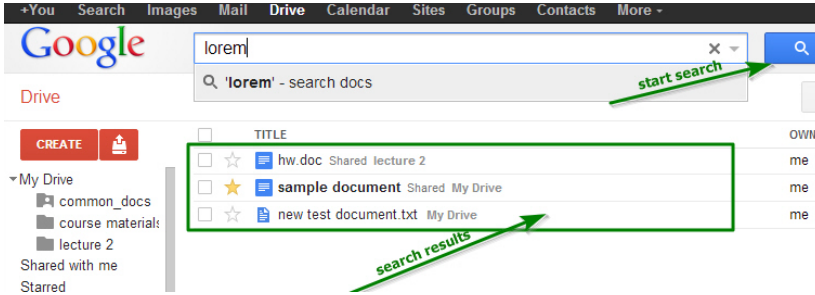


Рис. 2.47. Поиск документов по содержимому

Сложностей с поиском информации не возникает, поскольку в Google Apps for Education используется тот же интерфейс, что и на сайте поисковой машины Google. Стоит обратить особое внимание на то, что поиск осуществляется только в рамках отдельной службы. Таким образом, чтобы найти документ, необходимо перейти в сервис "**Drive**", а для поиска электронного письма или сообщения, соответственно, - в сервис "**Mail**". Поиск по всем объектам учебного заведения возможен в сервисах "**Drive**", "**Calendar**", "**Sites**", который можно осуществить в режиме расширенного поиска.

11.2.5. Технология "Microsoft Live@Edu"

Этот параграф посвящен сервису фирмы Microsoft Live@Edu (MLE), который практически идентичен GAfE по составу и качеству предлагаемых ОУ "облачных" услуг. Ввиду схожести подходов и принципов, используемых в MLE и GAfE, многие детали использования MLE будут опущены, как уже описанные в предыдущем параграфе.

Для использования учебной организацией службы Microsoft Live@edu необходимо заключить соглашение между данным образовательным учреждением (ОУ) и корпорацией Microsoft, в котором будут оговариваться условия использования данной службы.

Для использования службы Live@edu ОУ должно быть законным образовательным учреждением, основным видом деятельности которого является предоставление или администрирование образовательных услуг для конечных пользователей (выпускников, учащихся, преподавателей, штатных и бывших сотрудников, волонтеров или других лиц, связанных с данным ОУ). Для каждого конечного пользователя ОУ должен зарегистрировать профиль учетной записи, с помощью которого конкретный пользователь может получить доступ к Live@edu.

При заключении соглашения **Microsoft предоставит образовательному учреждению для его конечных пользователей (студентов и преподавателей) управляемую службу Outlook в режиме онлайн, включая электронную почту, использующую домен образовательного учреждения.** Рис 2.48 иллюстрирует, как выглядит страница настроек Outlook Live в окне браузера. Стрелка указывает на пример доменного имени, которое будет использоваться при отправке/получении почты: это имя не включает в себя каких-либо указаний на Microsoft, поэтому в случае прекращения действия соглашения ОУ сможет продолжить использование почтового доменного имени без изменений.

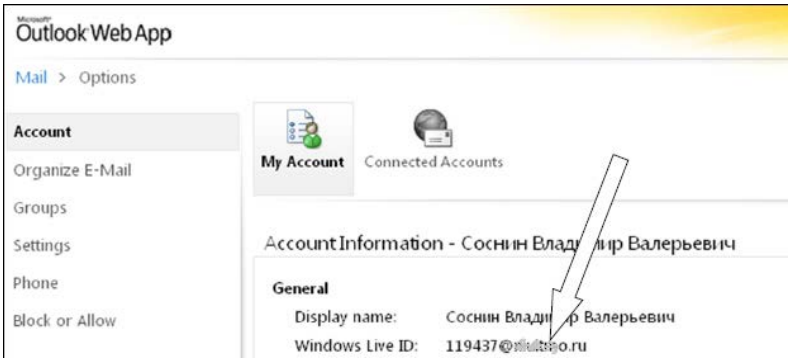


Рис. 2.48. Окно настройки параметров учётной записи Live@Edu

Кроме непосредственно онлайн-сервиса Live@Edu, Microsoft **на бесплатной основе предоставляет ОУ специальные программные средства, позволяющие автоматизировать управление профилями учетных записей конечных пользователей. Наличие таких средств выгодно отличает MLE от GAfE, т.к. позволяет существенно**

сэкономить временные затраты при первичной настройке облачных сервисов.

На рис. 2.49 представлено главное окно управление MLE, с которого начинает работу студент (преподаватель) после ввода аутентификационной информации. **Стрелкой указаны четыре основных сервиса, предоставляемых бесплатно в рамках MLE.** Все эти элементы хорошо известны пользователям программы Microsoft Outlook, поэтому таким пользователям не составит труда разобраться в работе Outlook Web App. Сложность в освоении может представлять лишь часть функционала, связанная с обеспечением совместной работы сразу нескольких пользователей.

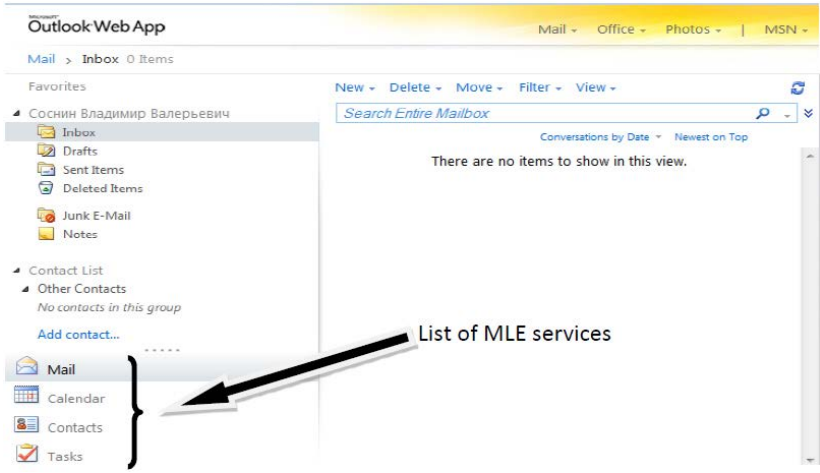


Рис. 2.49. Окно настройки параметров учётной записи Live@Edu

Ввиду интуитивной понятности интерфейса MLE покажем на примере, как работают лишь основные элементы управления. **На рис. 2.50 демонстрируется, как можно в рамках MLE создавать и редактировать некоторые виды документов: текстовые файлы Word, таблицы Excel и презентации PowerPoint. Этот набор можно считать достаточным для большинства образовательных целей ОУ.** Отметим, что в MLE (в отличие от GAfE) нет возможности сохранять документы в формате Open Document.

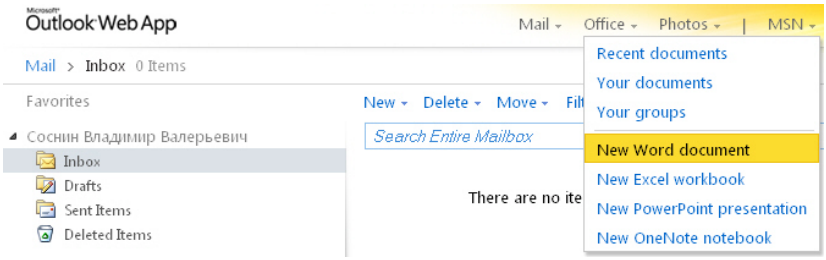


Рис. 2.50. Работа с документами в MLE

Выполнив команду "**Office->New Word document**" из основного меню, можно создать для редактирования простейший текстовый документ. Окно редактирования при этом будет выглядеть почти аналогично обычному офлайн-редактору MS Word 2010 (рис. 2.51).

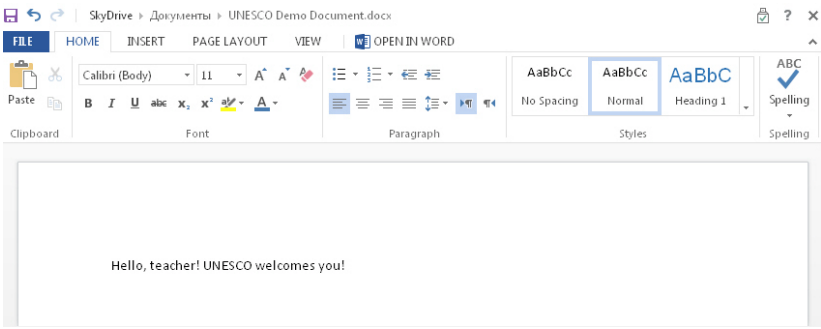


Рис. 2.51. Редактирование текстовых документов в MLE

После завершения редактирования созданного текстового документа **его можно сохранить в "облачном" хранилище данных Microsoft SkyDrive.** Делается это аналогично уже описанному Google Disk. Рис 2.52 иллюстрирует, как выглядит список созданных в "облаке" документов. Эти документы можно загрузить на компьютер для офлайн-редактирования, а можно предоставить к ним онлайн-доступ сотрудникам ОУ для совместного редактирования, аналогично тому, как это делается в сервисе облачного хранения Google Disk, описанном в предыдущем параграфе.

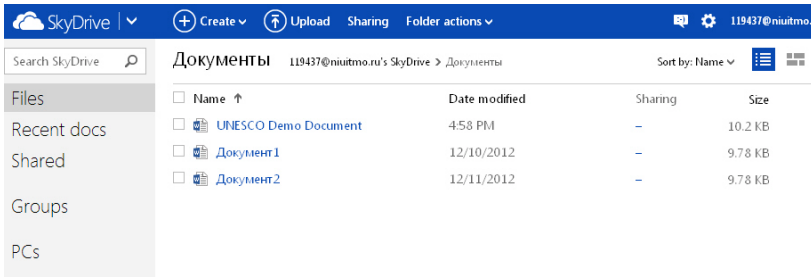


Рис. 2.52. Сервис "облачного" хранения данных SkyDrive

Остановимся на описании возможностей совместной работы подробнее, т.к. именно эта часть функционала MLE оказывается наиболее востребованной в многопользовательской среде, которой является ОУ. На рис. 2.53 показан процесс создания новой группы пользователей: можно указать имя группы, описание назначения группы и выбрать эл.адрес, соответствующий этой группе. В дальнейшем все сообщения, отправленные на этот адрес, будут доставлены только членам группы.

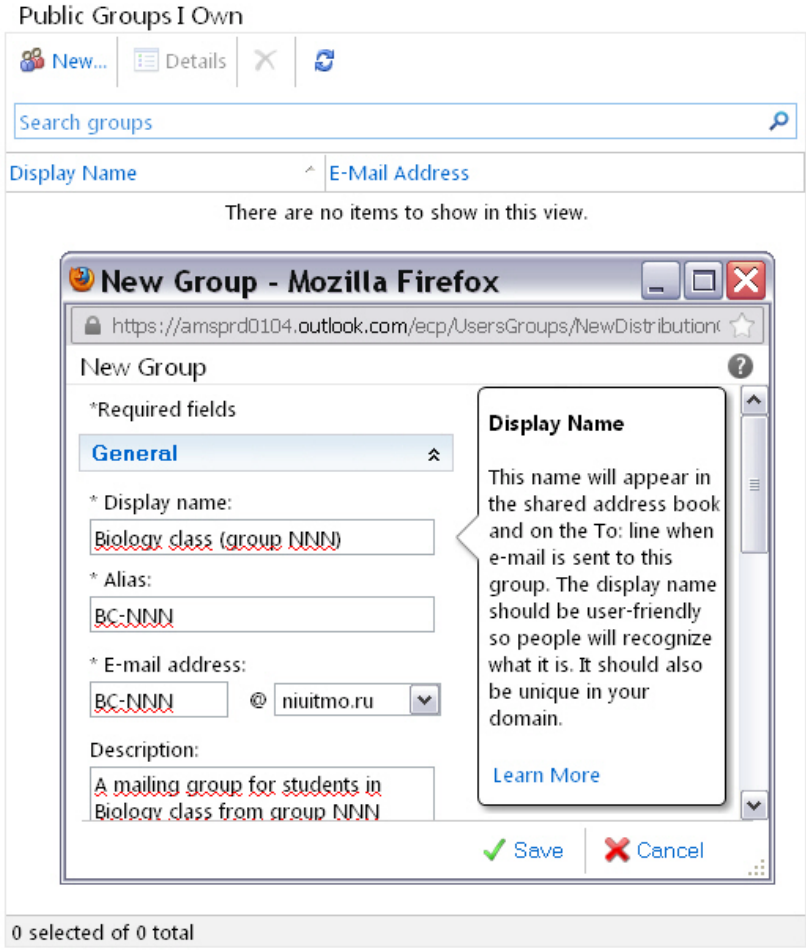


Рис. 2.53. Создание группы в Live@Edu

Каждая созданная группа может содержать неограниченное количество членов, которые могут быть наделены следующими ролями:
Владелец группы. Пользователь, создавший группу, является её владельцем. Такой пользователь может удалить или переименовать группу, а также добавить или убрать новых пользователей в группу. Кроме этого, владелец может настраивать специфические разрешения для пользователей, чтобы обеспечить возможность

дифференцированного доступа к рассылке групповых сообщений. Например, владелец может ввести этап премодерации сообщений от определенных пользователей, чтобы иметь возможность проверять посылаемые в группу сообщения до того, как все члены группы получат эти сообщения. Владелец группы может создать произвольное количество совладельцев группы, которые будут обладать всеми теми же правами, что и владелец. На рис. 2.54 показано диалоговое окно, в котором при нажатии на кнопку **Add** можно добавить в группе новых владельцев. Аналогично выглядит элемент управления, использующийся для добавления новых пользователей-невладельцев в группу.

- **Член группы**. Этот вид участия в работе группы предполагает возможность отправлять сообщения в группу, а также читать все разрешенные групповые сообщения (конкретными разрешениями управляет владелец группы).
- **Модератор группы**. Возможности модератора группы включают в себя уже упомянутую выше премодерацию отправляемых в группу сообщений. Именно модератор принимает решение о том, можно ли размещать в группе то или иное сообщение. Кроме того, модератор может использовать различные меры воздействия на авторов отклоненных.
- **Отправитель.** Для того чтобы отправить сообщение в группу, необязательно быть её членом. Любой владелец адреса эл.почты может послать сообщение в любую группу. Однако это не гарантирует фактического получения этого сообщения членами группы, т.к. по усмотрению владельца группы может использоваться этап премодерации сообщений.

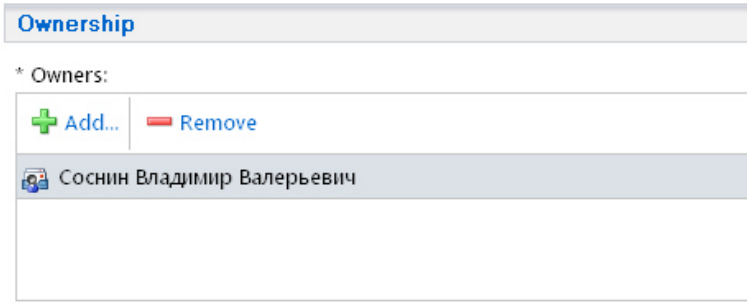


Рис. 2.54. Окно добавления владельцев в группу

Кроме управления потоком рассылки, создание группы может преследовать целый ряд целей: членам группы можно предоставлять исключительные права доступа к документам, имена групп можно эффективно использовать в правилах сортировки почтовых сообщений (обычно для сообщений, приходящих от доверенной группы, рекомендуется отключить проверку спам-фильтрами, чтобы исключить случаи их ложно-положительного срабатывания. Последним важных элементом MLE является календарь. Мы не будем подробно останавливаться на описании его функционала, т.к. он практически полностью повторяет функционал GAfE. Приведём лишь на рис. 2.55 пример того, как выглядит создание общего календаря, разделяемого несколькими пользователями. Для этого необходимо просто ввести эл.адрес пользователя, доступ к календарю которого требуется получить. Однако при этом требуется, чтобы указанный пользователь разрешил совместное использование своего календаря другим сотрудникам.

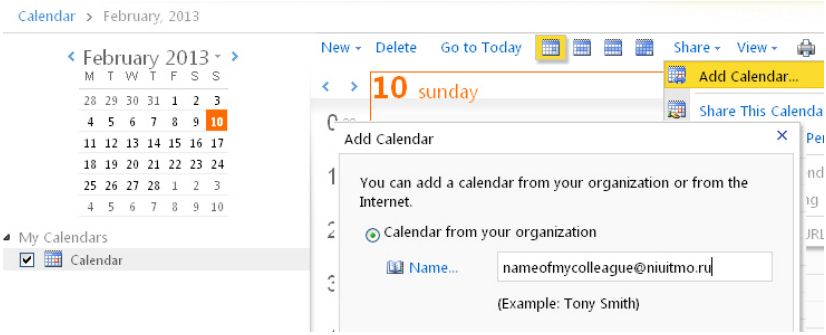


Рис. 2.55. Создание общего календаря событий Live@Edu

Подчеркнём важность элемента "Календарь" для образовательного процесса в любом ОУ. Если расписание учебных занятий реализовано с помощью этого элемента, каждый студент и преподаватель практически мгновенно может получить доступ к расписанию занятий любой учебной группы. Преподаватель может менять расписание занятий, и эти изменения автоматически попадут в расписание студентов. Кроме того, есть возможность настроить систему оповещений о предстоящих событиях, что позволит заблаговременно узнавать о возможных изменениях в учебном процессе.

11.3. Рекомендации по выбору и использованию облачных услуг

В разделе приводятся советы по принятию решения о выборе конкретного поставщика услуг облачных вычислений. Советы связаны со сравнением функциональности и стоимости сервисов, предоставляемых разными поставщиками, со сравнением программно-аппаратных платформ разных поставщиков, а также с анализом возможных технических работ, состав которых может зависеть от выбора поставщика услуг. Кроме того, рассматриваются юридические проблемы, связанные с оформлением договора о предоставлении услуг облачных вычислений.

11.3.1. Преимущества облачных вычислений для образовательных учреждений и учащихся

Использование облачных вычислений в области образования имеет много положительных сторон. Особо значимыми из них можно считать следующие преимущества:

- **Экономические преимущества.** Использование облачных технологий не требует капитальных затрат на создание и обслуживание собственных центров обработки данных, закупку серверного и сетевого оборудования для создания собственной IT-инфраструктуры. Также не требуются закупка и установка дорогостоящего программного обеспечения, регулярные обновления платформ и систем. Все эти расходы ложатся на поставщика облачного решения. В результате снижается нагрузка на технический персонал, что позволяет задействовать тех же научных сотрудников в других, более полезных для учреждения проектах.
- **Гибкая масштабируемость (эластичность).** Благодаря такой характеристике облачных сервисов, как эластичность, у образовательного учреждения имеется возможность постепенно наращивать объем используемых услуг без значительных предварительных вложений. В периоды пиковых нагрузок (например, во время сессий), не требуется планировать введение дополнительных информационных мощностей, поскольку облачные сервисы могут

масштабироваться автоматически и практически неограниченно.

- **Высокая доступность.** Облачные сервисы доступны в течение 99,5% времени, а некоторые провайдеры гарантируют доступность на уровне 99,9% . Это очень удобно для преподавателей и обучающихся, поскольку они могут реализовать возможности по обучению практически в любое время и не зависеть от локальных информационно-образовательных ресурсов учреждения. В результате это приводит к колоссальной экономии времени. Кроме того, постоянная доступность снимает преграды по получению дистанционного образования, например, в удаленных регионах, где на процесс обучения может влиять разница во времени. Высокая доступность образовательных ресурсов благоприятно влияет на рейтинг образовательного учреждения.
- **Уменьшение воздействия на окружающую среду.** Во многих странах объявлен курс на энергосберегающие ("зеленые") технологии, которые наносят меньший вред окружающей среде, чем традиционные. В соответствии с "зеленой" концепцией центры обработки данных должны использовать энергосберегающие технологии при проектировании и эксплуатации. Как показывает практика, для уменьшения воздействия на окружающую среду выгоднее использовать облачные услуги, которые используют "зеленые" технологии, чем внедрять такие технологии в локальной IT-инфраструктуре. Например, компания Google заявляет об увеличении энергоэффективности в 80 раз при использовании её облачных технологий (Google Apps for education).
- **Удовлетворение потребностей конечных пользователей.** Для конечных пользователей облачные технологии предоставляют еще больше преимуществ. Очень удобно, когда данные доступны из любого места, где есть Интернет и с любого устройства, будь то персональный компьютер, смартфон или планшет. Пользователям не нужно заботиться о резервных копиях, данные безопасно хранятся в "облаке". Облачная инфраструктура гарантирует сохранность данных. Если говорить о стандартном офисном пакете, который поставляется учебным заведениям бесплатно и может использоваться для решения очень широкого круга задач, то учащимся не потребуется приобретать, устанавливать и обновлять его на своих компьютерах. Единственное

приложение, которое будет требовать обновления – это веб-браузер.

- **Концентрация на ключевых задачах.** В любой сфере образования главная задача образовательных учреждений – концентрация усилий на образовании и исследованиях. При использовании облачных технологий сокращаются издержки на развертывание и поддержку используемых в работе приложений, высвобождаются человеческие ресурсы, которые могут быть задействованы в образовательном процессе.

11.3.2. Риски, связанные с использованием облачных вычислений

Облачные технологии появились сравнительно недавно, поэтому у некоторых складывается ощущение недоверия к ним. Не каждый руководитель ИТ-подразделений согласится отдать часть сервисов на аутсорсинг и/или разместить ключевые данные, необходимые для функционирования организации, а также конфиденциальную информацию у третьих лиц. Рассмотрим основные риски, связанные с использованием облачных технологий.

- **Безопасность данных.** Основным риском считается безопасность данных. Многие компании и организации считают, что их данные находятся в большей безопасности, если они хранятся в локальной информационной среде, когда физически можно увидеть носители с данными или оборудование, которое непосредственно участвует в обработке данных. Некоторым руководителям трудно даже представить себе ситуацию, что данные, с которыми идет работа в настоящий момент, могут физически находиться в дата-центрах, расположенных на других континентах и в других странах. На самом деле, особых поводов для опасения нет. Данные при передаче по открытым каналам данных шифруются, дополнительно к этому при заключении контракта с провайдером подписывается соглашение о неразглашении конфиденциальных данных. Основным риском для провайдеров является то, что при нарушении конфиденциальности или пропаже данных хотя бы одного пользователя (учащегося или сотрудника образовательного учреждения), этот пользователь может подать в суд на поставщика облачных услуг, что может привести к огромным

судебным издержкам и сильно испортить репутацию. Поэтому в условиях современной рыночной конкуренции каждый из поставщиков облачных технологий старается приложить все усилия, чтобы гарантировать безопасность и сохранность данных.

- **Снижение доступности.** Как говорилось выше, многие поставщики облачных услуг гарантируют доступность своих сервисов в течение 99,5% времени. Но, к сожалению, нельзя забывать о существовании, например, хакерских атак типа "отказ в обслуживании" (DoS-атаки), которые могут снизить общее время доступности. Для дополнительной минимизации рисков следует пользоваться услугами нескольких Интернет-провайдеров.
- **Привязка к поставщику.** Еще одним серьезным риском является привязка к определенному поставщику облачных услуг. Поскольку расходы по миграции из локальной среды в облако значительны, то, в случае, если поставщик перестанет удовлетворять потребности образовательного учреждения по каким-либо критериям (увеличится плата за использование, на рынке появится более хороший и дешевый сервис и т.д.), то сменить его будет достаточно проблематично. Денежные и временные затраты могут быть колоссальными. Поэтому необходимо ответственно отнестись к выбору поставщика облачных услуг.

Рассмотрим эти и некоторые другие виды рисков на примере пользовательского соглашения, заключаемого фирмой Microsoft с учебными заведениями для предоставления услуг Live@Edu.

Нежелательная реклама. В соответствии с упомянутым соглашением Microsoft имеет исключительное право на продажу и размещение рекламы в предоставленных образовательному учреждению службах. Само Учреждение такого права не имеет. При этом возможны следующие виды рекламы:

- графическая реклама (рекламные баннеры);
- текстовая реклама (бегущая строка);
- видео- и аудиореклама;
- рассылка рекламной корреспонденции.

Хотя в соглашении оговаривается, что демонстрируемая реклама может касаться лишь различных продуктов и услуг Microsoft, всё же такие условия работы сервиса Live@Edu могут оказаться неприемлемыми для некоторых учебных заведений.

Чрезмерные меры безопасности. В соответствии с соглашением Microsoft оставляет за собой право блокировать доставку любого типа электронной почты или других сообщений Live@Edu, если это необходимо для защиты служб или клиентов Microsoft или предотвращения нарушения ОУ условий лицензионного соглашения.

Разумной целью данного условия мог бы быть сервис фильтрации спама, однако Microsoft трактует право блокировки почты существенно шире: заблокированным может стать, например, любое письмо, которое Microsoft посчитает угрозой для защиты своих клиентов. К сожалению, такое нестрогое определение позволяет блокировать практически любые письма пользователей Live@Edu. И хотя на практике блокировка писем связана в большей части случаев именно с фильтрацией спам-сообщний, ОУ следует иметь в виду, что заблокированным может оказаться любое электронное письмо, чего бы не произошло при использовании традиционного не облачного почтового сервера в ОУ.

Сбор служебных данных. Microsoft может автоматически собирать определенные сведения о производительности службы Live@Edu на компьютерах пользователей. Кроме того, Microsoft может компилировать, использовать и разглашать сводную статистику о пользователях, если такое разглашение не позволяет установить личность отдельных пользователей.

Обычно данная информация обычно не является конфиденциальной, поэтому не содержит рисков для работы ОУ. Однако руководству ОУ следует обдумать возможные проблемы, косвенно связанные с разглашением подобной информации.

Конфиденциальная информация. При передаче персональных данных за пределы страны ОУ, Microsoft же обязуется соблюдать принципы соглашения Safe Harbour (изложенные Министерством торговли США), в отношении сбора, использования и хранения данных, полученных из других стран. Кроме этого, все персональные данные пользователей ОУ защищены в соответствии с Директивой

95/46/ЕС Европейского парламента и Совета. Если такая политика обработки данных в Microsoft не соответствует задачам ОУ, следует рассматривать её в качестве одного из существенных рисков.

Кроме исполнения указанных директив, Microsoft обязуется не использовать конфиденциальную информацию ОУ без письменного согласия этой стороны в течение пяти лет после ее получения, кроме как для осуществления взаимных деловых отношений. Также Microsoft обязуется не разглашать конфиденциальную информацию ОУ, кроме случаев, когда это касается получения совета у юридических или финансовых консультантов или по требованию законодательства. Майрософт обязуется уведомлять ОУ об обнаружении несанкционированного использования или разглашения конфиденциальной информации и будет оказывать всяческое содействие другой стороне для возврата конфиденциальной информации и предотвращения ее дальнейшего несанкционированного использования.

Раскрытие персональных данных конечных пользователей (включая содержимое сообщений) может происходить в следующих случаях:

- Для соблюдения требований закона, предоставления ответа на судебный запрос или выполнения требования судебных органов.
- Для защиты прав или собственности Microsoft или клиентов Microsoft.
- Для защиты сотрудников и клиентов Microsoft или других лиц.

Вышеперечисленные особенности политики обработки и передачи конфиденциальных данных означают достаточно низкий уровень защищенности конфиденциальной информации ОУ, что может существенно повлиять на решение об использовании облачных услуг Microsoft.

Отказ от гарантий и ответственности. Microsoft может время от времени обновлять условия использования своих служб. Кроме того, Microsoft и ее поставщики не гарантируют пригодность сервисов Live@edu для использования в целях образовательного учреждения, а также не несут ответственности за любые потери, которые могут произойти в результате несанкционированного использования Служб Microsoft. Более того, все материалы, программное обеспечение и

службы Live@Edu предоставляются поставщиками услуг "как есть", и Microsoft отказывается от любых гарантий относительно них, в том числе от любых подразумеваемых гарантий товарной пригодности, пригодности для определенной цели, принадлежности права, отсутствия нарушения прав иных правообладателей или результатов, которые могут быть получены учреждением и конечными пользователями в связи с использованием службы Live@edu.

"Microsoft и ее поставщики не несут ответственности за любые убытки (в том числе убытки в связи с упущенной выгодой, прерыванием деловой деятельности, потерей деловой информации или другие материальные убытки) возникающие вследствие использования или невозможности использования любой службы Microsoft, даже если корпорация Microsoft была предупреждена о возможности таких убытков или в случае если полученная компенсация не покрывает нанесенный учреждению ущерб. Максимальная совокупная ответственность одной стороны перед другой ни при каких обстоятельствах не может превышать десяти тысяч долларов".

Указанный ряд особенностей соглашения о предоставлении облачных услуг может показаться очень невыгодным для стороны ОУ в виду практически полного отказа Майкрософт от какой-либо ответственности и декларирования возможности в одностороннем порядке менять условия соглашения. Однако такая практика широко распространена при продаже программного обеспечения и является скорее правилом, чем исключением. Поэтому соответствующий раздел в соглашении хотя и должен быть принят к сведению, однако не должен оказывать существенное влияние на оценку рисков использования облачных технологий.

11.3.3. Рекомендации по выбору поставщика облачных услуг

Перед тем как выбрать провайдера облачных услуг, необходимо разработать перечень критериев, которые удовлетворяли бы потребностям организации по следующим направлениям:

- **Функциональность.** В этом направлении важно учесть требования пользователей к возможностям программ, используемых при работе в облаке. Для офисных пакетов

ключевым будет список поддерживаемых форматов и их совместимость с другими обычными и облачными приложениями, возможность экспорта в другие форматы. Немаловажно оценить и максимальный объем хранилища, предоставляемый каждому пользователю. Для прочих систем следует проанализировать дополнительные функциональные возможности, которые добавляют "изюминку" в эти решения (например, SMS-уведомления о переносе времени предстоящего экзамена и т.п.) и могут быть полезны для образовательного учреждения.

- **Платформа.** В настоящее время разработчики веб-браузеров стремятся к корректному отображению содержимого веб-страниц вне зависимости от вида операционной системы, используемой конкретным конечным пользователем. Но все же образовательному учреждению следует с особым вниманием отнестись к выбору основной платформы, для которой поставщик облачных услуг гарантирует полноценное функционирование. Возможно, придется рекомендовать пользователям установить и перейти на определенный вид операционных систем для обеспечения лучшей совместимости и быстродействия.
- **Технические особенности.** Скорее всего, перед переносом рабочих процессов в облако потребуется провести работы по автоматизации некоторых рутинных действий, например, написать скрипты по автоматической регистрации пользователей в облаке и загрузить первоначальную информацию.
- **Удобство и доступность для пользователей.** При выборе провайдера облачных услуг следует внимательно отнестись к удобству использования пользователями той или иной системы. Продуманность и лаконичность пользовательского интерфейса позволят сотрудникам более эффективно выполнять поставленные перед ними задачи. Многие поставщики облачных решений предоставляют бесплатный тестовый период. Рекомендуется организовать контрольную группу из преподавателей и учеников, которые помогли бы оценить удобство использования системы. Не следует забывать про возможность работы в выбранной системе для людей с ограниченными возможностями, это тоже не маловажный этический аспект.
- **Договор.** Необходимо проанализировать стандартный договор, предоставляемый провайдером. Особое внимание

следует обратить на следующие моменты: срок действия договора, штрафы за досрочное расторжение, возможность миграции данных во внешние системы, первоначальная и последующая стоимость услуг. В договоре об уровне сервисного обслуживания должны быть указаны размеры компенсаций, выплачиваемых в случае сбоев при эксплуатации систем. Особо важно оценить объем гарантий в тех случаях, когда услуги предоставляются бесплатно. Перед началом использования услуг рекомендуется ознакомиться с отзывами других пользователей о системе. Несмотря на то, что облачные услуги достаточно просты в использовании, следует обратить внимание на возможные варианты оказания технической поддержки. Может оказаться, что выгоднее заказать платную поддержку у поставщика облачных услуг, чем поддерживать конечных пользователей самостоятельно.

- **Расходы.** При планировании бюджета на переход к облачным технологиям следует оценить не только расходы на сами облачные услуги, но и оценить все сопутствующие затраты. Это могут быть расходы на управление, координацию и техническую реализацию проекта миграции в облако, различные юридические консультации, связанные с заключением договора, работы по первоначальному обучению пользователей работе в системе и т.д.

11.3.4. Организационно-правовые изменения

Развитие облачных сервисов в образовании не может продолжаться без определенных организационно-правовых изменений.

Организациям, которые планируют внедрять облачные технологии, прежде всего необходимо пересмотреть политику в отношении безопасности и конфиденциальности данных. Все риски по утрате и разглашении конфиденциальных данных должны регулироваться путем прямых договоренностей с провайдерами. Рекомендуется проконсультироваться с юристами и внести соответствующие изменения в договор об оказании услуг.

Персонал, ранее занятый обслуживанием IT-инфраструктуры организации, следует переориентировать на установку, настройку и контроль облачных услуг. Системным администраторам придется смириться с тем, что конечные пользователи больше не будут

следовать внутренним регламентам по использованию информационных и технических ресурсов, как это происходит при традиционной организации работы без использования облачных вычислений.

Что касается прав интеллектуальной собственности на данные, обрабатываемые в облаке, то необходимо абсолютно четко оговорить их в договоре. В соглашении о предоставлении облачных услуг должно быть указано, что права собственности на данные, размещенные и созданные в облаке должны оставаться у пользователя. Если все образовательные материалы будут размещены в "облаке" образовательной организации, то, скорее всего, потребуется иное оформление прав интеллектуальной собственности.

В европейских странах отдельные образовательные учреждения (школы, колледжи, университеты) обсуждают условия предоставления облачных услуг на уровне национальных и региональных органов управления образованием. Дополнительные преимущества здесь заключаются в том, что несколько образовательных учреждений могут стать частью одного "облака", что облегчит сотрудничество и ускорит обмен опытом между такими учреждениями.

Рассмотрим возможные организационно-правовые изменения в работе образовательного учреждения (ОУ), которые могут возникнуть при использовании облачных услуг компании Microsoft, на примере сервиса Live@Edu (в качестве источника используется [13]).

Фильтрация спама. В лицензионном соглашении между ОУ и Майкрософт указано, что Microsoft может отфильтровывать из электронной почты нежелательную почту и вредоносные программы. Это позволяет ОУ существенно сократить расходы на антивирусное программное обеспечение, однако повлечёт за собой организационно-правовые изменения в порядке работы ИТ-отдела ОУ, которому может потребоваться перенастраивать существенную часть используемого антивирусного программного обеспечения с возможной необходимостью расторгнуть договор на обеспечение антивирусной защиты с третьими лицами.

Родительское согласие. В соответствии с соглашением Microsoft не берет на себя обязанность получать родительское согласие на использование Live@edu каждым конечным пользователем

(студентом, преподавателем), т.е. ОУ обязано получить все законные разрешения у родителей и опекунов конечных пользователей на использование ими служб Microsoft. Это может потребовать существенных затрат времени и административного ресурса, что можно отнести к существенным изменениям в организационно-правовом поле работы ОУ. Все необходимые мероприятия, связанные с получением родительского согласия следует планировать заранее, т.к. затраты на их проведение достаточно сложно спрогнозировать.

Поддержка пользователей. Поддержка служб Microsoft осуществляется следующим образом: Microsoft будет предоставлять ОУ такой же уровень поддержки, какой предоставляется другим коммерческим конечным пользователям каждой из таких прочих служб Microsoft для потребителей, с помощью веб-сайта, расположенного по адресу http://support.live.com. Поддержку Microsoft осуществляет бесплатно, но оставляет за собой право в будущем перейти на предоставление платной поддержки.

Это означает, что в ОУ может потребоваться существенным образом менять локальную службу поддержки пользователей. Теперь большинство функций поддержки по обеспечению работы с электронной почтой и прочими услугами Live@edu можно делегировать Майкрософт, что позволяет упразднить некоторые должности в ОУ. Однако важно заметить условие соглашения, в соответствии с которым Майкрософт в неопределенном будущем может перейти на предоставление платной поддержки. Это означает, что решения о судьбе локальной службы поддержки ОУ должны быть взвешенными и должны учитывать возможные сценарии поведения поставщика облачных услуг.

Технические проблемы. В случае возникновения проблем с системой, сетью, сервером или важным приложением, находящимся под прямым единоличным управлением Microsoft, делающая невозможным использование службы Live@edu для большинства конечных пользователей или вызывающая значительные задержки и длящаяся более пятнадцати минут Microsoft в кратчайшие сроки обязан предпринять соответствующие действия для ограничения продолжительности и масштабов возникшей проблемы и приложить коммерчески обоснованные усилия для восстановления доступа к основной службе Live@edu и ее функциональности.

Это означает, как и в предыдущем случае, что ОУ имеет возможность существенно сократить затраты на поддержку пользователей, в данном случае – техническую. Подобные изменения неизбежно повлекут за собой ряд организационно-правовых изменений в работе ОУ, о чём следует заранее озаботиться руководству ОУ.

Обязательства ОУ. В соответствии с соглашением ОУ принимает следующие обязательства при работе с облачными технологиями Майкрософт:

- Учреждение обязано не реже чем раз в полгода обновлять профили учетных записи конечных пользователей. Это нужно, чтобы можно было дифференцировать учетные записи действительных учащихся и выпускников.
- ОУ несет ответственность за наличие у него прав на свои домены, используемые для предоставления служб Live@edu.
- ОУ запрещается использовать, копировать, воспроизводить, кэшировать, хранить или раскрывать пароли любому пользователю, за исключением владельца учетной записи.
- Чтобы предотвратить несанкционированный доступ к Паролям, административному ПО должен быть ограничен только специально назначенными сотрудниками ОУ.
- Учреждение обязуется следить за обновлениями, предоставляемыми Microsoft, и немедленно устанавливать их. ОУ не должно продолжать предоставление пользователям доступа к основной службе Live@edu, пока обновления не установлены.
- Учреждение обязуется немедленно сообщить Microsoft о любой существующей бреши в системе безопасности, и стороны должны по взаимному согласию предпринять соответствующие меры для немедленного ослабления любой действующей угрозы и предотвращения прогнозируемых угроз.

Каждый из перечисленных пунктов влечёт за собой большое количество организационно-правовых изменений. Например, требуется объяснить ИТ-персоналу новые обязанности, связанные с мониторингом и обновлением состояний профилей учётных записей на достаточно регулярной основе. Кроме того, требуется обеспечить защиту от несанкционированного доступа к аутентификационным данным Live@Edu. Следует также следить на обновлениями указанных

программных продуктов, а также отслеживать наличие брешей в системе безопасности этих продуктов, что требует затрат как машинного времени, так и времени обслуживающего персонала.

Меры безопасности. Отдельным параграфом в соглашении оговаривается внедрение ОУ следующих мер безопасности в связи с использованием основной службы Live@edu. Каждая из этих мер может потребовать существенных организационно-правовых изменений в работе ОУ, поэтому перечислим их ниже. Меры безопасности должны соответствовать следующим стандартам или превосходить их:

- Доступ к серверам, используемым Учреждением в связи с развертыванием службы Live@edu, должен быть защищен с помощью механизмов контроля доступа и ограничен назначенной группой сотрудников ОУ или подрядчиков, имеющих обязательства по соблюдению конфиденциальности.
- Ключ шифрования или другие учетные данные, которые должны быть установлены на серверах в связи с развертыванием службы Live@edu, по окончании установки должны быть уничтожены или закрыты в помещении с ограниченным доступом.
- Учреждение должно использовать криптографические протоколы для защиты сведений, отправляемых через Интернет.
- Учреждение должно следить за обновлениями и устанавливать все служебные обновления поставщиков, в том числе обновления для системы безопасности, обновления протокола и исправления.

Стороны обязуются принимать меры предосторожности для защиты конфиденциальной информации другой стороны, не меньшие, чем меры, принимаемые стороной для защиты собственной конфиденциальной информации.

Если Microsoft предъявит Учреждению обоснованное требование внедрить дополнительные процедуры и требования безопасности, стороны должны договориться о сроках такого внедрения.

Товарные знаки. Если образовательное учреждение решает использовать фирменную символику, то оно должно выдать Microsoft

бесплатную лицензию на использование этой символики. ОУ должно предоставить гарантии, что данная символика является оригинальной и не нарушает авторские права. Этот факт означает, что ОУ необходимо заблаговременно обеспокоиться регистрацией фирменной символики, т.к. подобные организационно-правовые изменения могут затронуть все виды деятельности ОУ (например, фирменная символика присутствует на фирменных бланках учреждения, поэтому изменение этой символики требует длительного времени).

Срок действия соглашения. Срок действия соглашения истекает через два года после даты вступления в силу; однако если одна из сторон не предоставит письменное уведомление о прекращении действия соглашения не позднее, чем за тридцать дней до завершения текущего срока, срок действия продлевается еще на один год.

Любая сторона может немедленно прекратить или приостановить действие соглашения, если другая сторона нарушит любую существенную гарантию, заявление, условие или обязательство указанное в соглашении и не сможет исправить такое нарушение в течение тридцати дней после получения письменного уведомления.

Юридические аспекты. Если ОУ организовано или создано в Соединенных Штатах или Японии, спорные правовые вопросы между Майкрософт и ОУ будут толковаться и рассматриваться согласно законодательству юрисдикции, в которой организовано или создано ОУ. В противном случае соглашение будет толковаться и рассматриваться согласно законодательству штата Вашингтон, исключая нормы коллизионного права. Это означает, что юристам ОУ может понадобиться ознакомиться с законодательством штата Вашингтон, чтобы предусмотреть риски, связанные с возможными правовыми последствиями неправомерных действий. Выполнить это достаточно сложно, т.к. в большинстве случаев требует от юриста уверенное владение английским языком, что может потребовать от ОУ нанимать дополнительный персонал.

Изменение соглашения. Важной проблемой является то, что Microsoft оставляет за собой право время от времени вносить изменения в заключенное соглашение на свое усмотрение. Но не менее чем за 30 дней ОУ будет прислано уведомление о вносимых изменениях. Если ОУ не согласно с ними, Microsoft может разорвать Соглашение. К сожалению, этот пункт сообщения требует основательной оценки

рисков, т.к. возможные запланированные организационно-правовые изменения могут оказаться недостаточными или ненужными в результате одностороннего изменения соглашения со стороны Майкрософт.

11.3.5. Правовые особенности использования облачных систем хранения данных

При использовании сервиса Dropbox необходимо принять во внимание следующие правовые аспекты.

Право собственности на материалы, размещенные на сервисе облачного хранения.

Согласно правилам обслуживания (Terms of Service), полные права интеллектуальной собственности на все материалы, размещенные на сервисе, сохраняются за пользователем.

Хотя сервис не претендует на права собственности относительно материалов, размещаемых в "облаке", согласие пользователя с правилами обслуживания предоставляет сервису ограниченные права только для предоставления своих услуг. Например, это могут быть такие услуги:

- предоставление общего доступа к материалам пользователя с его согласия;
- реализация функций предварительного просмотра изображений и документов для пользователя;
- реализация технических аспектов функционирования сервиса, например, выбор плана резервного копирования для обеспечения сохранности данных.

Ограниченные права могут быть переданы доверенным третьим сторонам, с которыми сотрудничает Dropbox (например, компании Amazon, которая предоставляет дисковое пространство для сервисов Dropbox).

Помимо этого, в некоторых случаях, оговоренных в политике конфиденциальности, информация, размещаемая на сервисе, может быть передана третьим лицам, в т.ч. правоохранительным органам.

Политика конфиденциальности. В рамках политики конфиденциальности Dropbox собирает и хранит следующую информацию:

- персональную информацию, указанную при регистрации (имя, номер телефона, адрес электронной почты, адреса в социальных сетях и т.п.);
- файлы и сведения об ассоциации файлов, хранящихся на сервисе облачного хранилища (если файл, загруженный на сервис одним пользователем, частично совпадает с файлом, загруженный другим пользователем, то сохраняется не дубликат файла, а только изменения);
- данные журналов событий, в которые записываются сведения об устройствах, с которых производился доступ к услугам облачного хранения, сведения о программном обеспечении, установленном на этих устройствах и действия пользователя во время использования сервиса (IP-адрес устройства, информация о браузере, идентификационный номер устройства, сведения об установленной операционной системе, провайдере Интернет-услуг и т.д.);
- cookie (куки) – небольшие файлы, которые передаются на устройство пользователя и служат для сбора информации и улучшения функций сервиса (например, для сохранения идентификатора пользователя, для лучшего понимания, как пользователь взаимодействует с сервисом, для мониторинга использования и оптимизации сетевого трафика).

Таким образом, Dropbox неявно анализирует содержимое информации, размещаемой в облаке, а также хранит различную дополнительную информацию, которая может повлиять на конфиденциальность пользователя. Этот факт следует учитывать при работе с сервисом.

Например, cookie (куки) можно отключить в настройках браузера, это лишь ограниченно скажется на функциональности сервиса.

Хранение метаданных. При размещении материалов в облаке необходимо помнить о том, что в некоторых файлах может храниться

дополнительная информация, комментирующая этот файл (метаданные). Это могут сведения об авторе, ключевые слова, название организации, даты создания/редактирования и прочие сведения. Например, некоторые устройства (смартфоны, планшеты, фотоаппараты и пр.), снабженные датчиком системы глобального позиционирования GPS, могут добавлять сведения о местоположении устройства (геолокационную информацию) при фотосъемке к получающимся изображениям.

Существует специальный стандарт EXIF [http://ru.wikipedia.org/wiki/EXIF], позволяющий добавлять к изображениям и прочим медиафайлам дополнительную информацию. Стандарт является очень гибким и поддерживается всеми производителями цифровой техники.

Таким образом, при предоставлении общего доступа к файлам на сервисах облачного хранения необходимо помнить, что метаданные файлов также будут доступны всем пользователям, которым предоставлен доступ.

Условия раскрытия информации третьим лицам. Хотя Dropbox обещает не предоставлять пользовательскую информацию третьим лицам, есть ситуации, в которых такая информация может быть предоставлена:

- по постановлению суда;
- если эта информация может угрожать жизни или здоровью любого человека;
- для предотвращения мошенничества в отношении Dropbox или его пользователей;
- для защиты прав собственности Dropbox.

Поэтому необходимо использовать сервис облачного хранения таким образом, чтобы не нарушать действующее законодательство. Согласно [2], пользователь несет полную ответственность за свое поведение, содержание файлов и папок, и связь с другими людьми во время использования услуг сервиса.

Срок хранения информации. Согласно политике конфиденциальности, Dropbox хранит информацию пользователя до

тех пор, пор пока активна его учетная запись. Для удаления учетной записи, необходимо зайти в настройки учетной записи на сайте Dropbox и выбрать пункт меню "**Удалить учетную запись**" (рис. 3.1).

При удалении учетной записи стоит помнить, что Dropbox оставляет за собой право сохранить и использовать пользовательскую информацию, необходимую для выполнения юридических обязательств (разрешения споров в суде, исполнения договоренностей и т.д.). В связи с этим в процессе удаления информации могут быть задержки, а резервные версии файлов могут существовать после удаления. Файлы, находящиеся в общем доступе с другими пользователями, не могут быть удалены с сервиса облачного хранения.

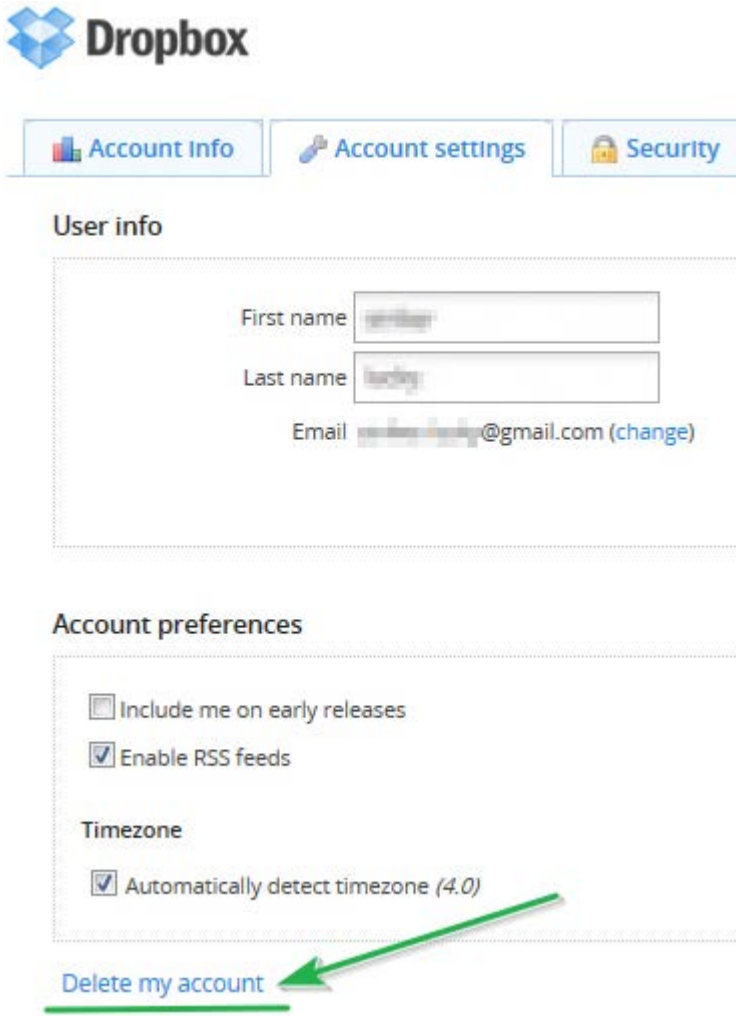


Рис. 3.1. Удаление учетной записи Dropbox

Ответственность пользователя при использовании Dropbox. Материалы в электронном виде (файлы), размещенные в облаке, могут являться интеллектуальной собственностью других лиц. Поэтому пользователям запрещается скачивать, копировать, загружать на сервис и предоставлять общий доступ к информации, если у них нет прав на эту информацию. Запрещается размещать на облачном сервисе

любые вредоносные программы (программы-шпионы, вирусы и т.д.). Пользователь несет полную ответственность за размещение и использование материалов на сервисе.

Пользователь также несет ответственность за защиту и сохранность всех своих материалов, которые были размещены в облаке. Dropbox не будет нести ответственность за потерю или повреждение файлов, а также за любые издержки, связанные с резервным копированием и восстановлением пользовательских данных.

Изменение условий обслуживания. Следует обязательно помнить о том, что сервис оставляет за собой право изменить условия обслуживания без предварительного уведомления пользователя, а также удалить любые материалы пользователя по своему усмотрению.

Прекращение обслуживания. Администрация сервиса Dropbox оставляет за собой право приостановить или прекратить предоставление услуг облачного хранения в любое время с или без предварительного уведомления (например, в случае не соблюдения пользователем правил использования сервиса, либо по постановлению суда или в случае возникновения опасности для других пользователей и т.д.).

Выводы. Перед принятием решения об использовании Dropbox в образовательном учреждении необходимо проанализировать возможные риски, которые связаны с правовыми аспектами использования сервиса.

Персонал образовательного учреждения должен быть обязательно ознакомлен с условиями обслуживания и политикой конфиденциальности сервиса Dropbox перед началом его использования.

11.3.6. Будущее облачных технологий в образовании

Как показывает практика, первыми облачные технологии осваивает мелкий и средний бизнес, а также различные стартап проекты. Образовательные учреждения в силу специфики своих внутренних процессов смогут перевести свои ключевые сервисы в "облако" еще не

скоро. Ведь одно дело – это отдать на аутсорсинг такую услугу, как электронная почта и совершенно другое – перевести в "облако" сложные системы управления обучением (LMS).

Мы живем в век информационных технологий, которые меняются очень динамично. Поставщики облачных вычислений с каждым годом улучшают предлагаемые услуги и расширяют их перечень. В бесплатных облачных продуктах от Microsoft и Google уже имеются функции, которые есть в LMS некоторых учебных заведений. Это будет способствовать постепенной миграции учебных заведений в "облака".

Помимо бесплатных облачных приложений, существуют специализированные LMS, такие как Blackboard, Moodle и Desire2Learn, которые можно уже сейчас беспрепятственно развернуть в "облаке". Поэтому кажется сомнительным, что в будущем учебные организации захотят устанавливать и сопровождать LMS самостоятельно на своих внутренних ресурсах, если облачные провайдеры смогут предоставить доступ к безопасным, легкодоступным и более дешевым аналогам традиционных LMS.

Большую роль в переходе к облачным технологиям сыграют конечные пользователи. Наряду с повсеместным ростом скорости Интернета, доступ к информации в "облаке" будет становиться все быстрее, проще и удобнее. Веб-приложения будут совершенствоваться, персональные данные можно будет безбоязненно хранить в "облаке", а не на собственных устройствах, которые могут в самый неподходящий момент выйти из строя. Таким образом, часть запросов на использование облачных технологий будет исходить не только от образовательных организаций, но и от конечных пользователей.

В современных **образовательных учреждениях наметилась очень четкая тенденция на развитие технологий дистанционного образования.** Основное направление – это публикация в открытом бесплатном доступе материалов некоторых или всех курсов образовательного учреждения. Публикуемые материалы включают планы курсов, конспекты лекций, домашние задания, экзаменационные вопросы. Для некоторых курсов доступны видеозаписи лекций. Одним из самых ярких примеров развития дистанционного образования является **Массачусетский технологический институт с проектом OpenCourseWare, на котором опубликовано более двух тысяч**

учебных курсов. И хотя этот проект не выдаёт пользователям сертификаты и дипломы и не предоставляет возможности связаться с сотрудниками института, опубликованная информация может быть использована преподавателями других учебных заведений.

Другим примером является международный проект Coursera, на котором публикуются полноценные курсы 33-х университетов, включающие в себя видеолекции с субтитрами, текстовые конспекты лекций, домашние задания, тесты и итоговые экзамены. Единственное, что необходимо для участия в этом образовательном проекте – учетная запись электронной почты и доступ к сети Интернет. С течением времени подобных проектов будет появляться больше, они будут развиваться и совершенствоваться, предлагать инновационные, более эффективные технологии обучения.

Учитывая все вышесказанное, можно уверенно сказать, что за облачными технологиями в образовании — будущее. Облачные технологии обладают огромным потенциалом и открывают широкие возможности не только для образовательных учреждений, но и для любого человека, который заинтересован в получении качественного образования.

11.3.7. Рабочая программа

Цель освоения дисциплины "Применение облачных вычислений в образовании" – научить студентов педагогического направления использовать в практике преподавания современные технологии облачных вычислений.

Структура дисциплины обобщенно представлена в табл. 4.1. Как видно из этой таблицы, на проведения курса отводится 20 часов, которые включают в себя проведение лекций, лабораторных работ и включают самостоятельную работу студентов (СРС) при подготовке к лекциям и лабораторным работам. О количестве часов, выделенных на каждый вид работ, можно узнать из табл. 4.1. Общая трудоёмкость дисциплины составляет 20 часов.

Таблица 4.1. Виды образовательной деятельности					
№ раздела	**Наименование раздела дисциплины**	**Виды учебной нагрузки и их трудоемкость, часы**			
		Лекции	**Лабораторные работы**	**СРС**	**Всего часов**
1	Теоретические основы облачных вычислений	1	0	2	3
2	Основы работы с облачными сервисами	2	7	4	13
3	Выбор облачных услуг и связанные с этим риски	1	1	2	4
ИТОГО:		**4**	**8**	**8**	**20**

Содержание (дидактика) дисциплины представлено тремя разделами, названия которых указаны в табл. 4.1. Рассмотрим детально каждый раздел, давая подробное описание входящих в него учебных тем.

Раздел 1. Теоретические основы облачных вычислений. Даётся определение технологий облачных вычислений. Приводится классификация этих технологий с подробным пояснением и примерами. Описывается отличи облачных вычислений от технологий Web2.0.

Раздел 2. Основы работы с облачными сервисами. Делается обзор наиболее популярных технологий облачных сервисов. Приводятся примеры работы с Microsoft Live@Edu, Google Apps For Education, а также популярных сервисов облачного хранения данных.

Раздел 3. Выбор облачных услуг и связанные с этим риски. Приводятся рекомендации по использованию конкретных облачных сервисов в образовательных учреждениях. Объясняются преимущества и недостатки этого подхода, отдельно освещаются вопросы организационно-правовых изменений, которые могут произойти в результате внедрения облачных технологий в образовательный процесс.

Курс включает в себя ряд лекций и лабораторных работ. Их перечень и краткая характеристика даны соответственно в табл. 4.2 и табл. 4.3.

Таблица 4.2. Темы лекций

№ п/п	Номер раздела	Тема лекции и перечень обсуждаемых вопросов
1	1	Введение в облачные вычисления: классификация, характеристики основных видов, отличие от Web 2.0
2	2	Обзор наиболее популярных облачных сервисов. Основы работы с облачными системами хранения
3	2	Основы работы с Google Apps for Education, Microsoft Live@Edu, облачным Moodle
4	3	Рекомендации по использованию облачных услуг, обзор преимуществ, недостатков и перечень рисков

Таблица 4.3. Темы лабораторных работ

№ п/п	Номер раздела	Наименование лабораторной работы и краткое описание решаемых проблем
1	1	**Аналитический обзор 3-4 сервисов облачных услуг, появившихся за последний год.** Студенты с помощью поисковых систем в Интернет находят появившиеся за прошедший год облачные проекты, соотносят их с предложенной на лекции классификацией и формулируют рекомендации по использованию рассмотренного сервиса в системе образования
2	2	**Основы работы с облачными сервисами хранения данных на примере Dropbox.** Студенты под руководством преподавателя создают учетные записи Dropbox, учатся добавлять/удалять файлы и совместно работать с файлами в облаке Dropbox, осваивают методы обеспечения конфиденциальности
3	2	**Основы работы с Moodle в облаке.** Студенты под руководством преподавателя создают простые учебные курсы в системе Moodle и размещают их в специализированном облаке
4	3	**Анализ организационно-правовых последствий применения облачных услуг.** Под руководством

		преподавателя студенты составляют перечень организационно-правовых изменений, которые потребуется сделать в работе учебных заведений, в которых преподают студенты-педагоги, на примере самостоятельного анализа случайно-взятого лицензионного соглашения о предоставлении облачных услуг

Курсовой проект. Кроме лабораторных работ к окончанию обучение требуется выполнить курсовой проект, в котором применяются на практике полученные навыки работы с "облачными" технологиями. Курсовой проект должен быть выполнен по теме той специальности, которую преподает в школе или вузе слушатель курса "Облачные вычисления в образовании". В рамках курсового проекта студент составляет комплексный бизнес-план по переходу на использование облачных услуг на примере отдельно взятой облачной технологии. Цель работы – отразить наиболее полно отразить все виды косвенных затрат при использовании облачных вычислений для получения адекватной реальности оценки экономической эффективности рассмотренного в курсовой работе облачного сервиса.

Формы контроля освоения дисциплины. Аттестация студентов производится преподавателями, ведущими лабораторные работы по дисциплине в следующих формах:

1. тестирование по теме, изученной на лекции;
2. выполнение и защита лабораторных работ;
3. оценка личностных качеств студента.

Финальная аттестация студентов производится по окончании обучения в форме экзамена и в виде защиты курсового проекта. Фонды оценочных средств, включающие типовые задания, тесты и методы контроля, позволяющие оценить работу студентов по данной дисциплине, перечислены в Приложении 2, в котором также приводятся критерии оценивания и таблица планирования результатов обучения.

Учебно-методическое и информационное обеспечение дисциплины включает в себя следующие элементы:

- данное пособие как основа для подготовки к лекциям;
- набор тестов по каждой из изучаемых тем в формате Moodle XML;
- мультимедиа данные с видеоинструкциями по следующей лекции данного курса: " "2. Характеристика основных типов "облачных" услуг" ".

Материально-техническое обеспечение дисциплины рассмотрим независимо в применении к каждой из форм проведения занятий. Для проведения лекционных занятий требуется:

- комплект электронных презентаций/слайдов по теме лекции;
- аудитория, оснащённая презентационной техникой (проектор, экран, компьютер/ноутбук, звуковые колонки).

Для проведения лабораторных занятий и приёма защиты курсовых проектов студентов требуется:

- лаборатория, оснащённая компьютерами с установленной UNIX-подобной операционной системой (например, Linux) или операционной системой фирм Microsoft (Windows XP и более новые) или Apple (Mac OS X 10.5 и более новые);
- наличие на лабораторных компьютерах одного из следующих браузеров: Internet Explorer 7, Firefox 3.0.1, Chrome 3.0.195.27, Safari 3.1 (указанная версия браузеров является минимально необходимой, т.е. для получения "облачных" сервисов подойдут и более современные версии указанных программ);
- аудитория, оснащённая презентационной техникой (проектор, экран, компьютер/ноутбук, звуковые колонки).

Отметим, что большинство "облачных" сервисов можно также использовать с помощью браузеров, которые не были перечислены выше. Это связано с тем, что приведённый перечень содержит лишь общие элементы из большого количества поддерживаемых браузеров на платформах различных поставщиков облачных услуг (см., например, здесь: http://help.outlook.com/en-us/exchangelabshelp/bb899685).

11.3.8. Технологии и форма преподавания

Укажем основные рекомендации по организации и технологиям обучения для преподавателя. Преподавание дисциплины ведётся с применением образовательных технологий, основной из которых является ресурс http://lms.iite.unesco.org с Веб 2.0 содержимым, включающим в себя **электронный конспект, электронные тесты, а также возможности участия студента в виде комментатора.**

Формы организации учебного процесса.
Лекционные занятия предлагается проводить по одному из следующих типовых шаблонов:

- информационные лекции с использование презентаций;
- лекции с заранее запланированными ошибками;
- проблемные лекции;
- лекции с разбором конкретной ситуации.

Лабораторные занятия предлагается проводить по одному из следующих типовых шаблонов:

- контекстное и проблемное обучение;
- работа в команде;
- индивидуальная работа со студентом.

Самостоятельная работа студентов (СРС) предполагает использование электронной образовательной среды и открытых Интернет источников и предполагает опережающее изучение материала. Кроме того, возможно перекрестное выполнение лабораторных работ.

Оценивание уровня учебных достижений студента осуществляется в виде текущего и финального рубежного контроля. Фонды оценочных средств включают в себя комплект тестов по материалам лекций (указать кол-во шт.), а также перечень тем для курсовых проектов (указать кол-во шт.).

Критерии оценивания. При оценивании курсового проекта независимо оцениваются следующие этапы работы:

- сбор материала, определение структуры изложения и консультирование с преподавателем;
- выступление с докладом перед своей учебной группой под контролем преподавателя;
- проверка корректности, содержательности и самостоятельности выполненной работы преподавателем без участия студента.

Итоговая оценка рассчитывается как среднее между оценками за каждый из трёх перечисленных этапов.

Допуск к выполнению лабораторной работы происходит при условии наличия у студента отчета по выполненному заданию после короткого собеседования. **Отчет по лабораторной работе представляет собой описание проделанной работы и предоставленную ссылку на реализованный студентом облачный проект в соответствии с заданием.** Защита лабораторной работы проходит в форме ответов на вопросы преподавателя (в письменной и/или устной форме).

Если задание выполнено в полном соответствии с техническим заданием и рекомендуемым стилем, использует оптимальный набор "облачных" технологий, комплектуется множеством тестовых примеров и показывает на них верные результаты, представленные в удобной для восприятия форме, то студент получает максимальное количество баллов. Основаниями для снижения количества баллов в диапазоне от минимального до максимального являются:

- неполное соответствие техническому заданию;
- неверный выбор технологии "облачных" услуг;
- недостаточное количество тестовых примеров.

Отчет не может быть принят и подлежит доработке в случае:

- серьезного несоответствия техническому заданию;
- отсутствия минимально необходимого количества тестовых примеров;
- некорректной работы программы.

Приложение

Исторический обзор и современное состояние дел

Теория процессов объединяет несколько направлений исследований, каждое из которых отражает определённый подход к моделированию и анализу процессов. Ниже мы рассматриваем наиболее крупные из этих направлений.

1 Робин Милнер

Наибольший вклад в теорию процессов внёс выдающийся английский математик **Робин Милнер** (см. **[1]** - [5]).

Робин Милнер родился в 1934 г. в г. Плимуте (Англия) в семье военного офицера. В настоящее время (с 1995 г.) он работает профессором информатики в университете Кембриджа (http://www.cam.ac.uk). С января 1996 г. по октябрь 1999 г. Милнер занимал должность руководителя Компьютерной Лаборатории в университете Кембриджа.

В 1971-1973 г. Милнер стажировался в Лаборатории искусственного интеллекта в Стэнфордском университете. С 1973 г. по 1995 г. он работал в отделении информатики (Computer Science Department) Университета Эдинбурга (Шотландия), в котором в 1986 г. он совместно с коллегами основал Лабораторию основ компьютерных наук (Laboratory for Foundation of Computer Science).

С 1971 до 1980, в Стэнфорде и затем в Эдинбурге, он работал в области автоматизации рассуждений. Совместно с коллегами он разработал Логику Вычислимых Функций (Logic for Computable Functions, LCF), развивающую подход Д. Скотта к построению теоретических основ понятия вычислимости, и предназначенную для автоматизации формальных рассуждений. Эта работа послужила основой для прикладных систем, разработанных под руководством Милнера.

В 1975-1990 Милнер руководил группой, которая разрабатывала Standard ML - широко используемый в индустрии и образовании язык программирования, семантика которого была полностью формализована (ML является сокращением словосочетания "meta-language"). В языке Standard ML был впервые реализован алгоритм вывода полиморфных типов. Главное достоинство Standard ML - возможность оперирования с логическими доказательствами и наличие средств автоматизации построения логических доказательств.

Около 1980 г. Милнер разработал Исчисление Взаимодействующих Систем (Calculus for Communicating Systems, CCS), одно из первых алгебраических исчислений для анализа параллельных процессов.

В конце 1980-х совместно с двумя коллегами он разработал тг-исчисление, которое является основной моделью поведения мобильных взаимодействующих систем.
В 1988 г. Милнер был избран членом Королевского Общества. В 1991 г. ему была присуждена высшая награда в области информатики - премия имени А.М.Тьюринга.
Главную цель своей научной деятельности сам Милнер определяет как построение теории, объединяющей понятие вычисления с понятием взаимодействия.

2 Исчисление взаимодействующих систем (CCS)

Исчисление Взаимодействующих Систем (Calculus of Communicating Systems, CCS) было впервые опубликовано Милнером в 1980 г. в книге [89]. Стандартным учебником по CCS является книга [92].
В [89] отражены результаты исследований Милнера, проведённые им в период с 1973 по 1980 г.
Основные работы Милнера по моделям параллельных процессов, выполненные в этот период:
• работы [84], [85], в них Милнер исследует денотационную семантику параллельных процессов
• [83], [88], здесь, в частности, введено понятие потокового графа (flow graph) с синхронизированными портами,
• [86], [87], в этих работах появилось современное CCS.
Используемая в CCS модель взаимодействия параллельных процессов, основанная на понятии передачи сообщений, взята Милнером из работы Хоара [71].
В статье [66] исследуются сильная и наблюдаемая эквивалентности и вводится логика Хеннесси - Милнера.
Понятия, введённые в CCS, развивались и в рамках других подходов, среди которых следует отметить в первую очередь
• тг-исчисление ([53], [97], [94]), и
• структурную операционную семантику (SOS), это направление было создано Г. Плоткиным, и опубликовано в работе [104].
Более подробная информация исторического характера о CCS может быть найдена в [105].

3 Теория взаимодействующих последовательных процессов (CSP)

Теория взаимодействующих последовательных процессов (Communicating sequential processes, CSP) была разработана английским математиком Тони Хоаром (C.A.R. Hoare) (р. в 1934). Эта теория возникла в 1976 г. и была опубликована в [71]. Более полное изложение CSP содержится в книге [73].

В CSP изучается модель взаимодействия параллельных процессов, основанная на передаче сообщений. Рассматривается синхронное взаимодействие между процессами.
Одним из ключевых понятий CSP является понятие охраняемой команды, которое Хоар позаимствовал из работы Дейкстры [52].
В [72] рассматривается модель CSP, основанная на теории трасс. Недостатком этой модели является отсутствие методов изучения дедлокового поведения. Этот недостаток устранён в другой модели CSP (failure model), введённой в [46].

4 Алгебра взаимодействующих процессов (ACP)

Ян Бергстра (Jan Bergstra) и Ян Биллем Клоп (Jan Willem Klop) в 1982 г. в работе [37] ввели термин "процессная алгебра" (process algebra), понимая по этим теорию первого порядка с равенством, в которой предметные переменные принимают значения в множестве процессов. Затем разработанные ими подходы привели к созданию нового направления в теории процессов - алгебры взаимодействующих процессов (Algebra of Communicating Processes, ACP), которое изложено в работах [39], [40], [34].
Главным объектом изучения в ACP являются логические теории, в которых испольуются функциональные символы, соответствующие операциям над процессами (a., +, и т.д.)
В [30] проводится сравнительный анализ различных точек зрения на понятие процессной алгебры.

5 Процессные алгебры

Термин **процессная алгебра (ПА),** введённый Бергстрой и Клопом, постепенно стал использоваться в двух значениях:
• в первом значении данный термин обозначает произвольную теорию первого порядка с равенством, областью интерпретации которой является некоторое множество процессов, и • во втором значении данный термин обозначает большой класс направлений, каждое из которых связано с некоторой алгебраической теорией, описывающей свойства процессов, в этом значении данный термин используется, например, в названии книги "Справочная книга по процессной алгебре" (Handbook of Process Algebra) [42]
Ниже мы перечисляем наиболее важные источники, относящиеся к ПА в обоих значениях этого термина.

1. Справочная книга по ПА [42].
2. Краткий обзор основных результатов в ПА: [19].
3. Исторические обзоры: [27], [28], [15].
4. Различные подходы, связанные с понятием эквивалентности процессов: [101], [59], [57], [58], [56].
5. ПА с семантикой частичных порядков: [44].

6. ПА с рекурсией: [91], [47].
7. SOS-модель для ПА: [21], [38].
8. Алгебраические методы верификации: [63].
9. ПА с данными (действия и процессы параметризованы элементами данных)
• ПА с данными */i-CRL*
• [62] (есть программное средство для верификации на основе излагаемого подхода).
• PSF [79] (есть программное средство).
• Язык формальных описаний LOTOS [45].
10. ПА с временем (действия и процессы параметризованы моментами времени)
• ПА с временем на основе CCS: [114], [99].
• ПА с временем на основе CSP: [107]. Учебник: [109].
• ПА с временем на основе ACP: [29].
• Интеграция дискретного и плотного времени, относительного и абсолютного времени: [32].
• Теория ATP: [100].
• Учёт времени в БМ: [33].
• Программное средство UPPAAL [74]
• Программное средство KRONOS [116] (временные автоматы) .
• */i-CRL* с временем: [111] (эквациональные рассуждения).
11. Вероятностные ПА (действия и процессы параметризованы вероятностями).
Данные ПА предназначены для комбинированного исследования систем, при котором одновременно производится верификация, и анализ производительности.
• Пионерская работа: [64].
• Вероятностные ПА, основанные на CSP: [76]
• Вероятностные ПА, основанные на CCS: [69]
• Вероятностные ПА, основанные на ACP: [31].
• ПА TIPP (и связанное с ней програмное средство): [60].
• ПА EMPA: [43].
• В работах [50] и [23] рассмотрено одновременное использование обычной и вероятностной альтернативной композиции процессов.
• В работе [51] рассмотрено понятие аппроксимации вероятностных процессов.
12. Программные средства, относящиеся к ПА
• Concurrency Workbench [98] (ПА типа CCS).
• CWB-NC [117].
• CADP [54].
• CSP: FDR http://www.fsel.com/

6 Мобильные процессы

Мобильные процессы описывают поведение распределённых систем, во время функционирования которых может изменяться

- конфигурация связей между их компонентами, и
- структура этих компонентов. Основные источники:

1. тг-исчисление (Milner и др.):
- старый справочник: [53],
- стандартный справочник: [97],
- учебники: [94], [8], [10], [9]
- страница в Википедии: [14]
- реализация тг-исчисления на распределённой вычислительной системе: [115].
- приложения тг-исчисления к моделированию и верификации криптографических протоколов: [12]
2. The ambient calculus: [48].
3. Action calculus (Milner): [93]
4. Биграфы: [95], [96].
5. Обзор литературы по мобильным процессам: [11].
6. Программное средство: Mobility Workbench [112].
7. Сайт www. cs . auc. dk/mobility

Другие источники:

- Лекция Р. Милнера "Computing in Space" [6], которую он прочитал на открытии здания им. Билла Гейтса, построенном для Компьютерной Лаборатории университета Кембриджа 1 мая 2002 г.

В лекции вводятся понятия "ambient" и "bigraph".

- Лекция Р. Милнера "Turing, Computing and Communication" [7].

7 Гибридные системы

Это системы, в которых

- значения одних переменных изменяются дискретно, и
- значения других переменных изменяются непрерывно.

Моделирование поведения таких систем производится с использованием дифференциальных и алгебраических уравнений.

Основные подходы:

- Гибридные процессные алгебры: [41], [49], [113].
- Гибридные автоматы [22] [77].

Для моделирования и верификации гибридных систем разработано программное средство HyTech [68].

8 Другие математические теории и программные средства, связанные с моделированием процессов

- Страница в Википедии по теории процессов [13].
- Теория сетей Петри [103].
- Теория частичных порядков [80].

• Темпоральная логика и model checking [106], [118].
• Теория трасс [108].
• Исчисление инвариантов [24].
• Метрический подход (в котором изучается понятие расстояния между процессами): [35], [36].
• SCCS [90].
• CIRCAL [82].
• MEIJE [25].
• Процессная алгебра Hennessy [65].
• Модели процессов с бесконечным числом состояний: [119], [120], [121], [122].
• Синхронно взаимодействующие автоматы: [123], [124], [125].
• Асинхронно взаимодействующие расширенные автоматы: **[126] [130].**
• Фрмальные языки SDL [131], Estelle [132], LOTOS [133].
• Формализм Statecharts, введенный D. НагеГом [134], [135] и входящий в язык проектирования UML.
• Модель взаимодействующих расширенных временных автоматов CETA (Communicating Extended Timed Automata) [136]-[140].
• A Calculus of Broadcasting Systems [17], [18].

11.9 Бизнес-процессы

1. BPEL (Business process execution language) [141].
2. BPML (Business Process Modeling Language) [16], [142].
3. Статья "Does Better Math Lead to Better Business Processes?" [143].
4. Страничка "тг-calculus and Business Process Management" [144].
5. Статья "Workflow is just a тг-process", Howard Smith and Peter Fingar, October 2003 [145].
6. "Третья волна" в моделировании бизнес-процессов: [146], [147].
7. Статья "Composition of executable business process models by combining business rules and process flows" [148].
8. Web services choreography description language [149].

Литература

[1] Web-страничка Р.Милнера
http://www.cl.cam.ac.uk/~rm135/
[2] Страничка Р.Милнера в Википедии
http://en.wikipedia.org/wiki/Robin_Milner
[3] Интервью Р. Милнера
http://www.dcs.qmul.ac.uk/~martinb/ interviews/milner/
[4] http://www.fairdene.com/picalculus/ robinmilner.html
[5] http://www.cs.unibo.it/gorrieri/ icalp97/Lauree_milner.html
[6] R. Milner: Computing in Space. *May, 2002.*
http://www.fairdene.com/picalculus/ milner-computing-in-space.pdf
[7] R. Milner: Turing, Computing and Communication. *King's College, October 1997.*
http://www.fairdene.com/picalculus/ milner-infomatics.pdf
[8] Учебное пособие по тг-исчислению: the pi-calculus, a tutorial.
http://www.fairdene.com/picalculus/ pi-c-tutorial.pdf
[9] J. Parrow: An introduction to the π-calculus. *[42], p. 479-543.*
[10] **D. Sangiorgi and D. Walker:** The π-calculus: A Theory of Mobile Processes. *ISBN 0521781779.*
http://us.cambridge.org/titles/ catalogue.asp?isbn=0521781779
[11] S. **Dal Zilio:** Mobile Processes: a Commented Bibliography.
http://www.fairdene.com/picalculus/ mobile-processes-bibliography.pdf
[12] M. **Abadi and A. D. Gordon:** A calculus for cryptographic protocols: The Spi calculus. *Journal of Information and Computation, 143:1-70, 1999.*
[13] Страница в Википедии "Process calculus"
http://en.wikipedia.org/wiki/Process_calculus
[14] Страница в Википедии по тг-исчислению
http://en.wikipedia.org/wiki/Pi-calculus
[15] **J.C.M.** Baeten: A brief history of process algebra, *Rapport CSR 04-02, Vakgroep Informatica, Technische Universiteit Eindhoven, 2004*
http://www.win.tue.nl/fm/0402history.pdf
[16] Business Process Modeling Language
http://en.wikipedia.org/wiki/BPML
[17] http: //en.wikipedia. org/wiki/ Calculus_of„Broadcasting_Systems
[18] K. V. S. **Prasad:** A Calculus of Broadcasting Systems, *Science of Computer Programming, 25, 1995.*
[19] L. Aceto: Some of my favorite results in classic process algebra. *Technical Report NS-03-2, BRICS, 2003.*
[20] L. Aceto, Z.T. Esik, W.J. Fokkink, and A. Ingolfsdottir (editors): Process Algebra: Open Problems and Future Directions. *BRICS Notes Series NS-03-3, 2003.*

[21] L. Aceto, W.J. Fokkink, and C. Verhoef: Structural operational semantics. *In [42], pp. 197-292, 2001.*
[22] **R. Alur, C Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine:** The algorithmic analysis of hybrid systems. *Theoretical Computer Science, 138:3-34, 1995.*
[23] **S. Andova:** Probabilistic Process Algebra. *PhD thesis, Technische Universiteit Eindhoven, 2002.*
[24] **K.R. Apt, N. Francez, and W.P. de Roever:** A proof system for communicating sequential processes. *TOPLAS, 2:359-385, 1980.*
[25] **D. Austry and G. Boudol:** Algebre de processus et synchronisation. *Theoretical Computer Science, 30:91-131, 1984.*
[26] **J.C.M. Baeten:** The total order assumption. *In S. Purushothaman and A. Zwarico, editors, Proceedings First North American Process Algebra Workshop, Workshops in Computing, pages 231-240. Springer Verlag, 1993.*
[27] **J.C.M. Baeten:** Over 30 years of process algebra: Past, present and future. *In L. Aceto, Z. T Esik, W.J. Fokkink, and A. Ingolfsdottir, editors, Process Algebra: Open Problems and Future Directions, volume NS-03-3 of BRICS Notes Series, pages 7-12, 2003.*
[28] http://www.win.**tue**.nl/fm/pubbaeten.html
[29] **J.C.M. Baeten and J.A. Bergstra:** Real time process algebra. *Formal Aspects of Computing, 3(2): 142-188, 1991.*
[30] J.C.M. Baeten, J.A. Bergstra, C.A.R. Hoare, R. Milner, J. Parrow, and R. de Simone: The variety of process algebra. *Deliverable ESPRIT Basic Research Action 3006, CONCUR, 1991.*
[31] J.C.M. Baeten, J.A. Bergstra, and S.A. Smolka: Axiomatizing probabilistic processes: ACP with generative probabilities. *Information and Computation, 121 (2)-.234-255, 1995.*
[32] **J.C.M. Baeten and C.A. Middelburg:** Process Algebra with Timing. *EATCS Monographs. Springer Verlag, 2002.*
[33] **J.C.M. Baeten, C.A. Middelburg, and M.A. Reniers:** A new equivalence for processes with timing. *Technical Report CSR 02-10, Eindhoven University of Technology, Computer Science Department, 2002.*
[34] **J.C.M. Baeten and W.P. Weijland:** Process Algebra. *Number 18 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1990.*
[35] **J.W. de Bakker and J.I. Zucker:** Denotational semantics of concurrency. *In Proceedings 14th Symposium on Theory of Computing, pages 153-158. ACM, 1982.*
[36] **J.W. de Bakker and J.I. Zucker:** Processes and the denotational semantics of concurrency. *Information and Control, 54:70-120, 1982.*

[37] **J.A. Bergstra and J.W. Klop:** Fixed point semantics in process algebra. *Technical Report IW 208, Mathematical Centre, Amsterdam, 1982.*
[38] **J.A. Bergstra and J.W. Klop:** The algebra of recursively defined processes and the algebra of regular processes. *In J. Paredaens, editor, Proceedings 11th ICALP, number 172 in LNCS, pages 82-95. Springer Verlag, 1984.*
[39] **J.A. Bergstra and J.W. Klop:** Process algebra for synchronous communication. *Information and Control, 60(l/3):109-137, 1984.*
[40] **J.A. Bergstra and J.W. Klop:** A convergence theorem in process algebra. *In J.W. de Bakker and J.J.M.M. Rutten, editors, Ten Years of Concurrency Semantics, pages 164-195. World Scientific, 1992.*
[41] **J.A. Bergstra and C.A. Middelburg:** Process algebra semantics for hybrid systems. *Technical Report CS-R 03/06, Technische Universiteit Eindhoven, Dept. of Comp. Sci., 2003.*
[42] **J.A. Bergstra, A. Ponse, and S.A. Smolka, editors:** Handbook of Process Algebra. *North-Holland, Amsterdam, 2001.*
[43] **M. Bernardo and R. Gorrieri:** A tutorial on EMPA: A theory of concurrent processes with non-determinism, priorities, probabilities and time. *Theoretical Computer Science, 202:1-54, 1998.*
[44] **E. Best, R. Devillers, and M. Koutny:** A unified model for nets and process algebras. *In [42], pp. 945-1045, 2001.*
[45] **E. Brinksma (editor):** Information Processing Systems, Open Systems Interconnection, LOTOS - A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour, *volume IS-8807 of International Standard. ISO, Geneva, 1989.*
[46] **S.D. Brookes, C.A.R. Hoare, and A.W. Roscoe:** A theory of communicating sequential processes. *Journal of the ACM, 31(3) -.560-599, 1984.*
[47] **O. Burkart, D. Caucal, F. Moller, and B. Steffen:** Verification on infinite structures. *In [42], pp. 545-623, 2001.*
[48] **L. Cardelli and A.D. Gordon:** Mobile ambients. *Theoretical Computer Science, 240:177-213, 2000.*
[49] P.J.L. Cuijpers and M.A. Reniers: Hybrid process algebra. *Technical Report CS-R 03/07, Technische Universiteit Eindhoven, Dept. of Comp. Sci., 2003.*
[50] **P.R. D'Argenio:** Algebras and Automata for Timed and Stochastic Systems. *PhD thesis, University of Twente, 1999.*
[51] **J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden:** Metrics for labeled Markov systems. *In J.C.M. Baeten and S. Mauw, editors, Proceedings CONCUR'99, number 1664 in Lecture Notes in Computer Science, pages 258-273. Springer Verlag, 1999.*

[52] **E.W. Dijkstra:** Guarded commands, nondeterminacy, and formal derivation of programs. *Communications of the ACM, 18(8):453- 457, 1975.*
[53] **U. Engberg and M. Nielsen:** A calculus of communicating systems with label passing. *Technical Report DAIMI PB-208, Aarhus University, 1986.*
[54] **J.-C. Fernandez, H. Garavel, A. Kerbrat, R. Mateescu, L. Mounier, and M. Sighireanu:** CADP (CAESAR/ALDEBARAN development package): A protocol validation and verification toolbox. *In R. Alur and T.A. Henzinger, editors, Proceedings CAV '96, number 1102 in Lecture Notes in Computer Science, pages 437-440. Springer Verlag, 1996.*
[55] **R.W. Floyd:** Assigning meanings to programs. *In J.T. Schwartz, editor, Proceedings Symposium in Applied Mathematics, Mathematical Aspects of Computer Science, pages 19-32. AMS, 1967.*
[56] **R.J. van Glabbeek:** The linear time - branching time spectrum II; the semantics of sequential systems with silent moves. *In E. Best, editor, Proceedings CONCUR '93, number 715 in Lecture Notes in Computer Science, pages 66-81. Springer Verlag, 1993.*
[57] R.J. van Glabbeek: What is branching time semantics and why to use it? *In M. Nielsen, editor, The Concurrency Column, pages 190-198. Bulletin of the EATCS 53, 1994.*
[58] **R.J. van Glabbeek:** The linear time - branching time spectrum I. The semantics of concrete, sequential processes. *In [42], pp. 3-100, 2001.*
[59] **R.J. van Glabbeek and W.P. Weijland:** Branching time and abstraction in bisimulation semantics. *Journal of the ACM, 43:555-600, 1996.*
[60] **N. Gotz, U. Herzog, and M. Rettelbach:** Multiprocessor and distributed system design: The integration of functional specification and performance analysis using stochastic process algebras. *In L. Donatiello and R. Nelson, editors, Performance Evaluation of Computer and Communication Systems, number 729 in LNCS, pages 121-146. Springer, 1993.*
[61] J.F. Groote: Process Algebra and Structured Operational Semantics. *PhD thesis, University of Amsterdam, 1991.*
[62] **J.F. Groote and B. Lisser:** Computer assisted manipulation of algebraic process specifications. *Technical Report SEN-R0117, CWI, Amsterdam, 2001.*
[63] **J.F. Groote and M.A. Reniers:** Algebraic process verification. *In [42], pp. 1151-1208, 2001.*
[64] **H. Hansson:** Time and Probability in Formal Design of Distributed Systems. *PhD thesis, University of Uppsala, 1991.*

[65] M. **Hennessy:** Algebraic Theory of Processes. *MIT Press, 1988.*
[66] **M. Hennessy and R. Milner:** On observing nondeterminism and concurrency. *In J.W. de Bakker and J. van Leeuwen, editors, Proceedings 7th ICALP, number 85 in Lecture Notes in Computer Science, pages 299-309. Springer Verlag, 1980.*
[67] **M. Hennessy and G.D. Plotkin:** Full abstraction for a simple parallel programming language. *In J. Becvar, editor, Proceedings MFCS, number 74 in LNCS, pages 108- 120. Springer Verlag, 1979.*
[68] **T.A. Henzinger, P. Ho, and H. Wong-Toi:** Hy-Tech: The next generation. *In Proceedings RTSS, pages 56-65. IEEE, 1995.*
[69] **J. Hillston:** A Compositional Approach to Performance Modelling. *PhD thesis, Cambridge University Press, 1996.*
[70] **C.A.R. Hoare:** An axiomatic basis for computer programming. *Communications of the ACM, 12:576-580, 1969.*
[71] **C.A.R. Hoare:** Communicating sequential processes. *Communications of the ACM, 21 (S)-.666-677, 1978.*
[72] C.A.R. Hoare: A model for communicating sequential processes. *In R.M. McKeag and A.M. Macnaghten, editors, On the Construction of Programs, pages 229-254. Cambridge University Press, 1980.*
[73] **C.A.R. Hoare:** Communicating Sequential Processes. *Prentice Hall, 1985.*
[74] **K.G. Larsen, P. Pettersson, and Wang Yi:** Uppaal in a nutshell. *Journal of Software Tools for Technology Transfer, 1, 1997.*
[75] **P. Linz:** An Introduction to Formal Languages and Automata. *Jones and Bartlett, 2001.*
[76] **G. Lowe:** Probabilities and Priorities in Timed CSP. *PhD thesis, University of Oxford, 1993.*
[77] **N. Lynch, R. Segala, F. Vaandrager, and H.B. Weinberg:** Hybrid I/O automata. *In T. Henzinger, R. Alur, and E. Sontag, editors, Hybrid Systems III, number 1066 in Lecture Notes in Computer Science. Springer Verlag, 1995.*
[78] **S. MacLane and G. Birkhoff:** Algebra. *MacMillan, 1967.*
[79] **S. Mauw:** PSF: a Process Specification Formalism. *PhD thesis, University of Amsterdam, 1991.* **http://carol.science.**uva**.nl/~psf/**
[80] A. Mazurkiewicz: Concurrent program schemes and their interpretations. *Technical Report DAIMI PB-78, Aarhus University, 1977.*
[81] J. **McCarthy:** A basis for a mathematical theory of computation. *In P. Braffort and D. Hirshberg, editors, Computer Programming and Formal Systems, pages 33-70. North-Holland, Amsterdam, 1963.*
[82] **G.J. Milne:** CIRCAL: A calculus for circuit description. *Integration, 1:121-160, 1983.*

[83] **G.J. Milne and R. Milner:** Concurrent processes and their syntax. *Journal of the ACM, 26(2) -.302-321, 1979.*
[84] **R. Milner:** An approach to the semantics of parallel programs. *In Proceedings Convegno di informatica Teoretica, pages 285-301, Pisa, 1973. Instituto di Elaborazione della Informazione.*
[85] **R. Milner:** Processes: A mathematical model of computing agents. *In H.E. Rose and J.C. Shepherdson, editors, Proceedings Logic Colloquium, number 80 in Studies in Logic and the Foundations of Mathematics, pages 157-174. North-Holland, 1975.*
[86] **R. Milner:** Algebras for communicating systems. *In Proc. AFCET/SMF joint colloquium in Applied Mathematics, Paris, 1978.*
[87] **R. Milner:** Synthesis of communicating behaviour. *In J. Winkowski, editor, Proc. 7th MFCS, number 64 in LNCS, pages 71-83, Zakopane, 1978. Springer Verlag.*
[88] **R. Milner:** Flowgraphs and flow algebras. *Journal of the ACM, 26(4)-.794-818, 1979.*
[89] **R. Milner:** A Calculus of Communicating Systems. *Number 92 in Lecture Notes in Computer Science. Springer Verlag, 1980.*
[90] **R. Milner:** Calculi for synchrony and asynchrony. *Theoretical Computer Science, 25:267-310, 1983.*
[91] **R. Milner:** A complete inference system for a class of regular behaviours. *Journal of Computer System Science, 28:439-466, 1984.*
[92] **R. Milner:** Communication and Concurrency. *Prentice Hall, 1989.*
[93] **R. Milner:** Calculi for interaction. *Ada Informatica, 33:707-737, 1996.*
[94] **R. Milner:** Communicating and Mobile Systems: the π-Calculus. *Cambridge University Press, ISBN 052164320, 1999.*
http://www.cup.org/titles/ catalogue.asp?isbn=0521658691
[95] **R. Milner:** Bigraphical reactive systems. *In K.G. Larsen and M. Nielsen, editors, Proceedings CONCUR '01, number 2154 in LNCS, pages 16-35. Springer Verlag, 2001.*
[96] **O. Jensen and R. Milner** Bigraphs and Mobile Processes. *Technical report, 570, Computer Laboratory, University of Cambridge, 2003.*
http://citeseer.ist.psu.edu/ j ensen03bigraphs.html
http://citeseer.ist.psu.edu/668823.html
[97] **R. Milner, J. lParrow, and D. Walker:** A calculus of mobile processes. *Information and Computation, 100:1-77, 1992.*
[98] **F. Moller and P. Stevens:** Edinburgh Concurrency Workbench user manual (version 7.1). http://www.dcs.ed.ac.uk/home/cwb/
[99] **F. Moller and C. Tofts:** A temporal calculus of communicating systems. *In J.C.M. Baeten and J.W. Klop, editors, Proceedings CONCUW90, number 458 in LNCS, pages 401-415. Springer Verlag, 1990.*

[100] **X. Nicollin and J. Sifakis:** The algebra of timed processes ATP: Theory and application. *Information and Computation, 114:131-178, 1994.*
[101] **D.M.R. Park:** Concurrency and automata on infinite sequences. *In P. Deussen, editor, Proceedings 5th GI Conference, number 104 in LNCS, pages 167-183. Springer Verlag, 1981.*
[102] **C.A. Petri:** Kommunikation mit Automaten. *PhD thesis, Institut fuer Instrumentelle Mathematik, Bonn, 1962.*
[103] **C.A. Petri:** Introduction to general net theory. *In W. Brauer, editor, Proc. Advanced Course on General Net Theory, Processes and Systems, number 84 in LNCS, pages 1-20. Springer Verlag, 1980.*
[104] **G.D. Plotkin:** A structural approach to operational semantics. *Technical Report DAIMI FN-19, Aarhus University, 1981.*
[105] **G.D. Plotkin:** The origins of structural operational semantics. *Journal of Logic and Algebraic Programming, Special Issue on Structural Operational Semantics, 2004.*
[106] **A. Pnueli:** The temporal logic of programs. *In Proceedings 19th Symposium on Foundations of Computer Science, pages 46-57. IEEE, 1977.*
[107] G.M. Reed **and** A.W. Roscoe: A timed model for communicating sequential processes. *Theoretical Computer Science, 58:249-261, 1988.*
[108] M. Rem: Partially ordered computations, with applications to VLSI design. *In J.W. de Bakker and J. van Leeuwen, editors, Foundations of Computer Science IV, volume 159 of Mathematical Centre Tracts, pages 1-44. Mathematical Centre, Amsterdam, 1983.*
[109] **S.A. Schneider:** Concurrent and Real-Time Systems (the CSP Approach). *Worldwide Series in Computer Science. Wiley, 2000.*
[110] D.S. **Scott and** C. **Strachey:** Towards a mathematical semantics for computer languages. *In J. Fox, editor, Proceedings Symposium Computers and Automata, pages 19-46. Polytechnic Institute of Brooklyn Press, 1971.*
[111] Y.S. **Usenko:** Linearization in *μCRL. PhD thesis, Technische Universiteit Eindhoven, 2002.*
[112] **B. Victor:** A Verification Tool for the Polyadic π-Calculus. *Licentiate thesis, Department of Computer Systems, Uppsala University, Sweden, May 1994. Report DoCS 94/50.*
[113] **T.A.C. Willemse:** Semantics and Verification in Process Algebras with Data and Timing. *PhD thesis, Technische Universiteit Eindhoven, 2003.*
[114] **Wang** Yi: Real-time behaviour of asynchronous agents. *In J.C.M. Baeten and J.W. Klop, editors, Proceedings CONCUR'90, number 458 in LNCS, pages 502- 520. Springer Verlag, 1990.*
[115] **L. Wischik:** New directions in implementing the π-calculus. *University of Bologna, August 2002.*
http://www.fairdene.com/picalculus/ implementing-pi-c.pdf

[116] **S. Yovine:** Kronos: A verification tool for real-time systems. *Journal of Software Tools for Technology Transfer, 1:123-133, 1997.*
[117] **D. Zhang, R. Cleaveland, and E. Stark:** The integrated CWB-NC/PIOAtool for functional verification and performance analysis of concurrent systems. *In H. Garavel and J. Hatcliff, editors, Proceedings TACAS'03, number 2619 in Lecture Notes in Computer Science, pages 431-436. Springer-Verlag, 2003.*
[118] **E. Clarke, O. Grumberg, D. Peled:** Model checking. *MIT Press, 2001.*
[119] J.Esparza: Decidability of model-checking for infinite-state concurrent systems, *Ada Informatica, 34:85-107, 1997.*
[120] **P.A.Abdulla, A.Annichini, S.Bensalem, A.Bouajjani, P.Habermehl, Y.Lakhnech:** Verification of Infinite-State Systems by Combining Abstraction and Reachability Analysis, *Lecture Notes in Computer Science 1633, pages 146-159, Springer- Verlag, 1999.*
[121] **K.L.McMillan:** Verification of Infinite State Systems by Compositional Model Checking, *Conference on Correct Hardware Design and Verification Methods, pages 219-234, 1999.*
[122] **O.Burkart, D.Caucal, F.Moller, and B.Steffen:** Verification on infinite structures, *In J. Bergstra, A. Ponse and S. Smolka, editors, Handbook of Process Algebra, chapter 9, pages 545-623, Elsevier Science, 2001.*
[123] **D. Lee and M. Yannakakis.** Principles and Methods of Testing Finite State Machines - a Survey. The Proceedings of the IEEE, 84(8), pp 1090-1123, 1996.
[124] **G. Holzmann.** Design and Validation of Computer Protocols. Prentice-Hall, Englewood Cliffs, N.J., first edition, 1991.
[125] **G. Holzmann.** The SPIN Model Checker - Primer and Reference Manual. Addison-Wesley, 2003.
[126] **S. Huang, D. Lee, and M. Staskauskas.** Validation-Based Test Sequence Generation for Networks of Extended Finite State Machines. In Proceedings of FORTE/PSTV, October 1996.
[127] **J. J. Li and M. Segal.** Abstracting Security Specifications in Building Survivable Systems. In Proceedings of 22-тв National Information Systems Security Conference, October 1999, Arlington, Virginia, USA.
[128] **Y.- J. Byun, B. A. Sanders, and C.-S. Keum.** Design Patterns of Communicating Extended Finite State Machines in SDL . In Proceedings of 8-th Conference on Pattern Languages of Programs (PLoP'2001), September 2001, Monticello, Illinois, USA.
[129] **J. J. Li and W. E. Wong.** Automatic Test Generation from Communicating Extended Finite State Machine (CEFSM)-Based Models. In

Proceedings of the Fifth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'02), pp. 181-185, 2002.
[130] S. Chatterjee. EAI Testing Automation Strategy. In Proceedings of 4-th QAI Annual International Software Testing Conference in India, Pune, India, February 2004
[131] ITU Telecommunication Standardization Sector (ITU-T), Recommendation Z.100, CCITT Specification and Description Language (SDL), Geneva 1994.
[132] Information Processing Systems - Open Systems Interconnection: Estelle, A Formal Description Technique Based on Extended State Transition Model, ISO International Standard 9074, June 1989.
[133] ISOIEC. LOTOS - A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour. International Standard 8807, 1988.
[134] **D. Harel.** A Visual Formalism for Complex Systems. Science of Computer Programming, 8:231-274, 1987.
[135] **D. Harel and A. Naamad.** The STATEMATE semantics of statecharts. ACM Transactions on Software Engineering and Methodology (Also available as technical report of Weizmann Institute of Science, CS95-31), 5(4):293-333, Oct 1996.
[136] **M. Bozga, J. C. Fernandez, L. Ghirvu, S. Graf, J. P. Krimm, and L. Mounier.** IF: An intermediate representation and validation environment for timed asynchronous systems. In Proceedings of Symposium on Formal Methods 99, Toulouse, number 1708 in LNCS. Springer Verlag, September 1999.
[137] M. Bozga, S. Graf, and L. Mounier. IF-2.0: A validation environment for componentbased real-time systems. In Proceedings of Conference on Computer Aided Verification, CAV'02, Copenhagen, LNCS. Springer Verlag, June 2002.
[138] M. Bozga, D. Lesens, and L. Mounier. Model-Checking Ariane-5 Flight Program. In Proceedings of FMICS'01, Paris, France, pages 211-227. INRIA, 2001.
[139] M. Bozga, S. Graf, and L. Mounier. Automated validation of distributed software using the IF environment. In 2001 IEEE International Symposium on Network Computing and Applications (NCA 2001). IEEE, October 2001.
[140] M. Bozga and Y. Lakhnech. IF-2.0 common language operational semantics. Technical report, 2002. Deliverable of the IST Advance project, available from the authors.
[141] http://www-128.ibm.com/developerworks/ library/specification/ws-bpel/
[142] http://www.bpml.org

[143] http://www.wfme.org/standards/docs/
better_maths_better_processes.pdf
[144] http://www.fairdene.com/picalculus/
[145] http://www.bpmi.org/bpmi-library/
2B6EA45491.workflow-is-just-a-pi-process.pdf
[146] http://www.fairdene.com/picalculus/ bpm3-apx-theory.pdf
[147] http://www.bpm3.com
[148] http://portal.acm.org/citation.cfm?id=1223649
[149] http://www.w3.org/TR/ws-cdl-10/

1. А.Е. Кононюк. Дискретно-непрерывная математика. Математическая логика. Кн.9. Ч.1 Киев: «Освіта України», 2017. 464с.
2. А.Е. Кононюк. Дискретно-непрерывная математика. Математическая логика. Кн.9. Ч.2 Киев: «Освіта України», 2017. 564с.
3. А.Е. Кононюк. Дискретно-непрерывная математика. Математическая логика. Кн.9. Ч.3 Киев: «Освіта України», 2017. 424с.
4. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 1. Киев: «Освіта України», 2017. 730с
5. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 2. Киев: «Освіта України»,2017.668 с
6. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 3, ч.1. Киев: «Освіта України», 2017. 608 с
7. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 3, ч.2. Киев: «Освіта України», 2017. 645 с
8. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 3, ч.3. Киев: «Освіта України», 2017. 645 с
9. А.Е. Кононюк. Дискретно-непрерывная математика. Алгоритмы. Кн.10. Ч.1 Киев: «Освіта України», 2017. 588с
10. 9. А.Е. Кононюк. Дискретно-непрерывная математика. Автоматы. Кн.11. Ч.1 Киев: «Освіта України», 2017. 658с
11. Дж. Ллойд. Системы тепловидения. Москва, 1978
12. А.Е. Кононюк. Общая теория познания и созидания. Кн.1. Киев: «Освіта України», 2013. 648 с. *ecat.diit.edu.ua:81/ft/index_ru.html*
13. А.Е. Кононюк. Общая теория познания и созидания. Кн.2, ч.1. Киев: «Освіта України», 2013. 544 с. *ecat.diit.edu.ua:81/ft/index_ru.html*
14. А.Е. Кононюк. Общая теория познания и созидания. Кн.2, ч.2. Киев: «Освіта України», 2013. 644 с. *ecat.diit.edu.ua:81/ft/index_ru.html*
15. А.Е. Кононюк. Информациология. Общая теория информации. Кн.1. Киев: «Освіта України», 2011. 476 с.
ecat.diit.edu.ua:81/ft/index_ru.html

16. А.Е. Кононюк. Информациология. Общая теория информации. Кн.2. Киев: «Освіта України», 2011. 476 с. *ecat.diit.edu.ua:81/ft/index_ru.html*
17. А.Е. Кононюк. Информациология. Общая теория информации. Кн.3. Киев: «Освіта України», 2011. 412 с. *ecat.diit.edu.ua:81/ft/index_ru.html*
18. А.Е. Кононюк. Информациология. Общая теория информации. Кн.4. Киев: «Освіта України», 2011. 488 с. *ecat.diit.edu.ua:81/ft/index_ru.html*

19. А.Е. Кононюк. Общая теория понятий. Кн.1. Киев: «Освіта України», 2014. 514с.

20. А.Е. Кононюк. Общая теория понятий. Кн.2. Киев: «Освіта України», 2014. 544с.

21. А.Е. Кононюк. Общая теория понятий. Кн.3. Киев: «Освіта України», 2014. 614с.

22. А.Е. Кононюк. Системология. Общая теория систем. Кн.1. Киев: «Освіта України», 2012. 564с. *ecat.diit.edu.ua:81/ft/index_ru.html*
23. А.Е. Кононюк. Системология. Общая теория систем. Кн.2. Ч.1. Киев: «Освіта України», 2014. 558с. *ecat.diit.edu.ua:81/ft/index_ru.html*
24. А.Е. Кононюк. Системология. Общая теория систем. Кн.2. Ч.2. Киев: «Освіта України», 2014. 658с. *ecat.diit.edu.ua:81/ft/index_ru.html*
25. А.Е. Кононюк. Системология. Общая теория систем. Кн.2. Ч.1. Киев: «Освіта України», 2014. 558с. *ecat.diit.edu.ua:81/ft/index_ru.html*
26. А.Е. Кононюк. Общая теория распознавания. Кн.1. Киев: «Освіта України», 2012. 584 с. *ecat.diit.edu.ua:81/ft/index_ru.html*
27. А.Е. Кононюк. Общая теория распознавания. Кн.2. Киев: «Освіта України», 2012. 588 с. *ecat.diit.edu.ua:81/ft/index_ru.html*
28. А.Е. Кононюк. Консалтология. Общая теория консалтинга. Кн.1. Киев: «Освіта України», 2013. 448 с. *ecat.diit.edu.ua:81/ft/index_ru.html*
29. А.Е. Кононюк. Консалтология. Общая теория консалтинга. Кн.2. Киев: «Освіта України», 2013. 412 с. *ecat.diit.edu.ua:81/ft/index_ru.html*
30. А.Е. Кононюк. Консалтология. Общая теория консалтинга. Кн.3. Киев: «Освіта України», 2013. 520 с. *ecat.diit.edu.ua:81/ft/index_ru.html*
31. А.Е. Кононюк. Консалтология. Общая теория консалтинга. Кн.4. Киев: «Освіта України», 2013. 508 с. *ecat.diit.edu.ua:81/ft/index_ru.html*
32. А.Е. Кононюк. Дискретно-непрерывная математика. Начала. Кн.1. Киев: «Освіта України», 2012. 652с. *ecat.diit.edu.ua:81/ft/index_ru.html*

33. А.Е. Кононюк. Дискретно-непрерывная математика. Множества. Кн.2. Ч.1. Киев: «Освіта України», 2012. 452с.
ecat.diit.edu.ua:81/ft/index_ru.html
34. А.Е. Кононюк. Дискретно-непрерывная математика. Множества. Кн.2. Ч.2. Киев: «Освіта України», 2013. 536 с.
ecat.diit.edu.ua:81/ft/index_ru.html
35. А.Е. Кононюк. Дискретно-непрерывная математика. Отношения. Кн.3. Ч. 1. Киев: «Освіта України», 2013. 552с.
ecat.diit.edu.ua:81/ft/index_ru.html
36. А.Е. Кононюк. Дискретно-непрерывная математика. Отношения. Кн.3. Ч. 2. Киев: «Освіта України», 2013. 548 с.
ecat.diit.edu.ua:81/ft/index_ru.html
37. А.Е. Кононюк. Дискретно-непрерывная математика. Алгебры. Кн.4. Ч.1. Киев: «Освіта України», 2011. 452с.
ecat.diit.edu.ua:81/ft/index_ru.html
38. А.Е. Кононюк. Дискретно-непрерывная математика. Алгебры. Кн.4. Ч.2. Киев: «Освіта України», 2011. 668 с.
ecat.diit.edu.ua:81/ft/index_ru.html

39. А.Е. Кононюк. Дискретно-непрерывная математика. Алгебры. Кн.4. Ч.3. Киев: «Освіта України», 2015. 488 с.
http://lib.sumdu.edu.ua/library/DocDescription?doc_id=640902

40. А.Е. Кононюк. Дискретно-непрерывная математика. Алгебры. Кн.4. Ч.4. Киев: «Освіта України», 2015. 548 с.

41. А.Е. Кононюк. Дискретно-непрерывная математика. Алгебры. Кн.4. Ч.5. Киев: «Освіта України», 2015. 528 с.

42. А.Е. Кононюк. Дискретно-непрерывная математика. Алгебры. Кн.4. Ч.6. Киев: «Освіта України», 2015. 608 с.

43. А.Е. Кононюк. Дискретно-непрерывная математика. Матрицы. Кн.5. Ч.1. Киев: «Освіта України», 2013. 612 с.
ecat.diit.edu.ua:81/ft/index_ru.html

44. А.Е. Кононюк. Дискретно-непрерывная математика. Матрицы. Кн.5. Ч.2. Киев: «Освіта України», 2013. 500 с.
ecat.diit.edu.ua:81/ft/index_ru.html

45. А.Е. Кононюк. Дискретно-непрерывная математика. Матрицы. Кн.5. Ч.3. Киев: «Освіта України», 2013. 520 с. *ecat.diit.edu.ua:81/ft/index_ru.html*
46. А.Е. Кононюк. Дискретно-непрерывная математика. Матрицы. Кн.5. Ч.4. Киев: «Освіта України», 2013. 508 с. *ecat.diit.edu.ua:81/ft/index_ru.html*
47. А.Е. Кононюк. Дискретно-непрерывная математика. Матрицы. Кн.5. Ч.5. Киев: «Освіта України», 2013. 672 с. *ecat.diit.edu.ua:81/ft/index_ru.html*
48. А.Е. Кононюк. Дискретно-непрерывная математика. Поверхности. Кн.6. Ч.1. Киев: «Освіта України», 2012. 652с. *ecat.diit.edu.ua:81/ft/index_ru.html*
49. А.Е. Кононюк. Дискретно-непрерывная математика. Графы. Кн.7. Ч.1 Киев: «Освіта України», 2014. 652с. *ecat.diit.edu.ua:81/ft/index_ru.html*
50. А.Е. Кононюк. Дискретно-непрерывная математика. Графы. Кн.7. Ч.2 Киев: «Освіта України», 2014. 552с. *ecat.diit.edu.ua:81/ft/index_ru.html*
51. А.Е. Кононюк. Дискретно-непрерывная математика. Графы. Кн.7. Ч.3 Киев: «Освіта України», 2015. 512с. *ecat.diit.edu.ua:81/ft/index_ru.html*
52. А.Е. Кононюк. Дискретно-непрерывная математика. Графы. Кн.7. Ч.4 Киев: «Освіта України», 2015. 552с. *ecat.diit.edu.ua:81/ft/index_ru.html*

53. А.Е. Кононюк. Дискретно-непрерывная математика. Графы. Кн.7. Ч.5 Киев: «Освіта України», 2015. 660с.

54. А.Е. Кононюк. Обобщенная теория моделирования. Кн.1. Ч.1 Киев: «Освіта України», 2012. 602с. *ecat.diit.edu.ua:81/ft/index_ru.html*
55. А.Е. Кононюк. Обобщенная теория моделирования. Кн.1. Ч.2 Киев: «Освіта України», 2012. 708с. *ecat.diit.edu.ua:81/ft/index_ru.html*
56. А.Е. Кононюк. Обобщенная теория моделирования. Кн.1. Ч.3 Киев: «Освіта України», 2012. 568с. *ecat.diit.edu.ua:81/ft/index_ru.html*
57. А.Е. Кононюк. Обобщенная теория моделирования. Кн.2. Киев: «Освіта України», 2012. 548с. *ecat.diit.edu.ua:81/ft/index_ru.html*
58. А.Е. Кононюк. Обобщенная теория моделирования. Кн.3. Ч.1 Киев: «Освіта України», 2012. 636с. *ecat.diit.edu.ua:81/ft/index_ru.html*
59. А.Е. Кононюк. Обобщенная теория моделирования. Кн.3. Ч.2 Киев: «Освіта України», 2012. 448с. *ecat.diit.edu.ua:81/ft/index_ru.html*
60. А.Е. Кононюк. Обобщенная теория моделирования. Кн.3. Ч.3 Киев: «Освіта України», 2013. 588с. *ecat.diit.edu.ua:81/ft/index_ru.html*

61. А.Е. Кононюк. Основы теории оптимизации. Кн.1. Киев: «Освіта України», 2011. 602с. *ecat.diit.edu.ua:81/ft/index_ru.html*
62. А.Е. Кононюк. Основы теории оптимизации. Кн.2. Ч.1. Киев: «Освіта України», 2011. 552с. *ecat.diit.edu.ua:81/ft/index_ru.html*
63. А.Е. Кононюк. Основы теории оптимизации. Кн.2. Ч.2. Киев: «Освіта України», 2011. 616с. *ecat.diit.edu.ua:81/ft/index_ru.html*
64. А.Е. Кононюк. Основы теории оптимизации. Кн.2. Ч.3. Киев: «Освіта України», 2012. 456с. *ecat.diit.edu.ua:81/ft/index_ru.html*
65. А.Е. Кононюк. Основы теории оптимизации. Кн.2. Ч.4. Киев: «Освіта України», 2012. 512с. *ecat.diit.edu.ua:81/ft/index_ru.html*
66. А.Е. Кононюк. Основы научных исследований. Кн.1. Киев: «Освіта України», 2011. 508с. *ecat.diit.edu.ua:81/ft/index_ru.html*
67. А.Е. Кононюк. Основы научных исследований. Кн.2. Киев: «Освіта України», 2011. 452с. *ecat.diit.edu.ua:81/ft/index_ru.html*
68. А.Е. Кононюк. Основы научных исследований. Кн.3. Киев: «Освіта України», 2011. 456с. *ecat.diit.edu.ua:81/ft/index_ru.html*
69. А.Е. Кононюк. Основы научных исследований. Кн.4. Киев: «Освіта України», 2011. 456с. *ecat.diit.edu.ua:81/ft/index_ru.html*

70. А.Е. Кононюк. Общая теория коммуникаций. Кн.1. Киев: «Освіта України», 2014. 488с.

71. А.Е. Кононюк. Нейроні мережі і генетичні алгоритми. Киев: «Корнійчук», 2010. 448с. *ecat.diit.edu.ua:81/ft/index_ru.html*

72. Кононюк А. Е. Обобщенная теория познания и созидания. [В 2 кн.] Кн. 1 : Начала / А. Е. Кононюк. — Киев : Освіта України, 2013. *ecat.diit.edu.ua:81/ft/index_ru.html*

73. Кононюк А. Е. Обобщенная теория познания и созидания. [В 2 кн.] Кн. 2 : Теория познания. Ч. 1 / А. Е. Кононюк. — Киев : Освіта України, 2013 *ecat.diit.edu.ua:81/ft/index_ru.html*

74. Кононюк А. Ю. Вища математика. (Модульна технологія навчання) : навчальний посібник : в 2 кн. / А. Ю. Кононюк. — Київ : КНТ, 2009 — Кн. 1. — 2009. — 702 с. *ecat.diit.edu.ua:81/ft/index_ru.html*

75. Кононюк А. Ю. Вища математика. (Модульна технологія навчання) : навчальний посібник : в 2 кн. / А. Ю. Кононюк. — Київ : КНТ, 2009 Кн. 2. — 2009. — 790 с. *ecat.diit.edu.ua:81/ft/index_ru.html*

76. А.Е. Кононюк. Дискретно-непрерывная математика. Поверхности. Кн.6. Ч.2. Киев: «Освіта України», 2012. 652с. http://www.dut.edu.ua/ua/lib/127/category/96/view/1297

77. А.Е. Кононюк. Дискретно-непрерывная математика. Пространства. Кн.8. Ч.1. Киев: «Освіта України», 2016. 748 с. http://www.dut.edu.ua/ua/lib/1/category/96/view/1439

78. А.Е. Кононюк. Дискретно-непрерывная математика. Пространства. Кн.8. Ч.2. Киев: «Освіта України», 2016. 480с. http://lib.sumdu.edu.ua/library/DocDescription?doc_id=640775

79. А.Е. Кононюк. Истины и информация (фундаментальная теория представления истин и информации). К.1. Киев: «Освіта України», 2016. 568с.
80. А.Е. Кононюк. Истины и информация (фундаментальная теория представления истин и информации). К.2. Киев: «Освіта України», 2016. 558с.

81. А.Е. Кононюк. Истины и информация (фундаментальная теория представления истин и информации). К.3. Киев: «Освіта України», 2016. 588с.
82. А.Е. Кононюк. Истины и информация (фундаментальная теория представления истин и информации). К.4. Киев: «Освіта України», 2016. 552с
83. А.Е. Кононюк. Истины и информация (фундаментальная теория представления истин и информации). К.5. Киев: «Освіта України», 2016. 836 с
84. А.Е. Кононюк. Истины и информация (фундаментальная теория представления истин и информации). К.6. Киев: «Освіта України», 2016. 576 с

85. А.Е. Кононюк. Дискретно-непрерывная математика. Математическая логика. Кн.9. Ч.1 Киев: «Освіта України», 2017. 464с.
86. А.Е. Кононюк. Дискретно-непрерывная математика. Математическая логика. Кн.9. Ч.2 Киев: «Освіта України», 2017. 564с.
87. А.Е. Кононюк. Дискретно-непрерывная математика. Математическая логика. Кн.9. Ч.3 Киев: «Освіта України», 2017. 424с.
88. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 1. Киев: «Освіта України», 2017. 730с

89. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 2. Киев: «Освіта України»,2017.668 с
90. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 3, ч.1. Киев: «Освіта України», 2017. 608 с
91. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 3, ч.2. Киев: «Освіта України», 2017. 645 с
92. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн.4. Киев: «Освіта України», 2017. 655 с
93. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн.5. Киев: «Освіта України», 2018. 615 с
94. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн.6. Киев: «Освіта України», 2018. 645 с
95. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн.7. Киев: «Освіта України», 2018. 665 с
96. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн.8. Киев: «Освіта України», 2018. 684 с
97. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн.9. Киев: «Освіта України», 2018. 655 с
98. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн.10. Киев: «Освіта України», 2018. 678 с
99. А.Е. Кононюк. Основы теории облачных технологий. Киев: «Освіта України», 2018. 710 с
100. А.Е. Кононюк. Фундаментальная теория облачных технологий. Кн.1. Киев: «Освіта України», 2018. 710 с

т□