
С. М. Авдошин, А. А. Набебин



Дискретная математика. Формально-логические системы и языки



Москва, 2018

УДК 510.6
ББК 22.12
А18

Рецензенты:

Калягин В. А. — доктор физико-математических наук, профессор НИУ ВШЭ
Петренко А. К. — доктор технических наук, профессор,
заведующий отделом «Технологии программирования»
Института системного программирования РАН

Научный редактор:

Захаров В. А. — доктор физико-математических наук,
профессор МГУ им. М. В. Ломоносова

А18 Авдошин С. М., Набебин А. А.
Дискретная математика. Формально-логические системы и языки. – М.: ДМК Пресс, 2018. – 390 с.: ил.

ISBN 978-5-97060-622-3

Книга содержит основные сведения из формально-логических систем. Это функции алгебры логики (булевы функции), теорема Поста о функциональной полноте, k -значные логики, производные булевых функций, аксиоматические исчисления высказываний, предикатов, секвенций, резолюций и язык программирования Пролог. Рассматриваются монадическая логика, конечные автоматы и представимые ими языки, темпоральная логика, аксиоматический язык программирования OBJ3.

В основу книги положен многолетний опыт преподавания авторами дисциплины «Дискретная математика» на факультете бизнес-информатики, на факультете компьютерных наук Национального исследовательского университета Высшая школа экономики и на факультете автоматизации и вычислительной техники Национального исследовательского университета Московский энергетический институт.

Книга предназначена для студентов бакалавриата, обучающихся по направлениям 09.03.01 «Информатика и вычислительная техника», 09.03.02 «Информационные системы и технологии», 09.03.03 «Прикладная информатика», 09.03.04 «Программная инженерия», а также для ИТ-специалистов и разработчиков программных продуктов.



УДК 510.6
ББК 22.12

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

ISBN 978-5-97060-622-3

© Авдошин С. М., Набебин А. А., 2018
© Оформление, издание, ДМК Пресс, 2018

Содержание



Предисловие	9
Введение	13
Часть I. АЛГЕБРА ЛОГИКИ И ПРЕДИКАТЫ	27
Глава 1. Алгебра логики	28
1.1. Функции алгебры логики	28
1.2. Формулы. Реализация функций формулами	29
1.3. равносильные преобразования формул	31
1.4. Нормальные формы. Совершенные нормальные формы	33
Совершенные нормальные формы	34
1.5. Минимизация нормальных форм	36
1.5.1. Алгоритм Куайна построения сокращенной ДНФ	38
1.5.2. Алгоритм построения сокращенной ДНФ с помощью КНФ	39
1.5.3. Построение всех тупиковых ДНФ	41
1.5.4. Алгоритм минимизации функций в классе ДНФ	42
1.5.5. Алгоритм минимизации функций в классе КНФ	43
1.5.6. Алгоритм минимизации функций в классе нормальных форм	43
1.6. Минимизация частично определенных функций	45
1.6.1. Алгоритм минимизации частично определенных функций в классе ДНФ	46
1.6.2. Алгоритм минимизации частично определенных функций в классе КНФ	46
1.7. Двойственные функции. Принцип двойственности	49
1.8. Линейные функции	50
1.9. Монотонные функции	53
1.10. Теорема Поста о функциональной полноте	55
1.11. Предполные классы	58
Глава 2. Функции k-значной логики	60
2.1. Функции и отношения	60
2.2. Самодвойственные функции	63
2.3. Монотонные функции	63
2.4. Линейные функции	64
2.5. Функции, сохраняющие разбиение	64
2.6. Классы типа $\mathbb{C}$	64

2.7. Классы типа $\mathbb{B}$	65
2.8. Сравнение функций двужначной и многозначной логик.....	66
Глава 3. Производные булевой функции в синтезе логических схем.....	67
3.1. Производная булевой функции.....	67
3.2. Синтез логических схем методом каскадов.....	70
3.3. Разложение булевой функции в ряд.....	77
Глава 4. Синтез схем из функциональных элементов.....	80
4.1. Схема из функциональных элементов	80
4.2. Функции Шеннона	82
4.3. Элементарные методы синтеза схем	82
4.4. Синтез мультиплексоров	84
4.5. Элементы функциональной декомпозиции	86
4.6. Обнаружение неисправностей в схемах.....	91
Глава 5. Аксиоматическое исчисление высказываний	94
5.1. Определение исчисления высказываний	94
5.2. Теорема дедукции в исчислении высказываний	98
5.3. Производные правила вывода	100
5.4. Тавтологически истинные и доказуемые формулы.....	105
5.5. Разрешимость, непротиворечивость, полнота, независимость аксиом.....	108
5.6. Формулировка исчисления высказываний с единственным правилом вывода – правилом заключения.....	111
Глава 6. Логика предикатов.....	113
6.1. Предикаты, кванторы.....	113
6.2. Общезначимость, выполнимость, невыполнимость, опровержимость формул логики предикатов.....	115
6.3. Равносильность формул	119
6.4. Нормальные формы	121
6.4.1. Префиксная нормальная форма	121
6.4.2. Стандартная форма Сколема	122
6.5. Проблема разрешимости в логике предикатов.....	125
6.5.1. Проблема разрешимости $\exists$ -формул.....	126
6.5.2. Проблема разрешимости $\forall$ -формул.....	127
6.5.3. Проблема разрешимости монадической логики	128
6.6. Отношения	130
6.7. Суперпозиция функций.....	132
6.8. Операции Мальцева над функциями.....	133
6.9. Алгебра отношений (реляционная алгебра).....	133
6.10. Алгебра отношений k -значной логики.....	135

Глава 7. Аксиоматическое исчисление предикатов	136
7.1. Определение исчисления предикатов	136
7.2. Теорема дедукции в исчислении предикатов	138
7.3. Непротиворечивость исчисления предикатов	141
7.4. Семантическая полнота исчисления предикатов относительно класса общезначимых формул	142
7.4.1. Непротиворечивые расширения исчисления предикатов	143
7.4.2. Формализмы G и G_k	145
Глава 8. Исчисление секвенций	151
8.1. О правилах вывода в секвенциальном исчислении высказываний	151
8.2. Секвенциальное исчисление высказываний	155
8.3. Секвенциальное исчисление предикатов	158
Глава 9. Метод резолюций в логике предикатов и Пролог	161
9.1. Метод резолюций в логике высказываний	161
9.1.1. Семантическое дерево	162
9.1.2. Правило резолюции	164
9.2. Эрбрановы универсум, базис, интерпретация	167
9.3. Семантические деревья. Теорема Эрбрана	170
9.4. Унификация	174
9.5. Метод резолюций в логике предикатов	177
9.6. Основы Пролога	182
9.6.1. Унификация в Прологе	189
9.6.2. Декларативный и операторный смысл Пролог-программы	191
9.6.3. Бэктрекинг и оператор отсечения	193
9.6.4. Объявление операторов	194
9.7. Примеры программ и вычислений в Прологе	196
9.7.1. Принадлежность элемента списку	196
9.7.2. Первый элемент в списке	199
9.7.3. Последний элемент в списке	200
9.7.4. Следующий элемент в списке	201
9.7.5. Соединение списков	204
9.7.6. Обращение списка	205
9.7.7. Выравнивание списка	206
9.7.8. Добавление элемента в начало списка	206
9.7.9. Удаление первого вхождения данного элемента из списка	207
9.7.10. Удаление всех вхождений данного элемента из списка	207
9.7.11. Замена элемента в списке	210
9.7.12. Быть подсписком в списке	210
9.7.13. Включение множеств	212

9.7.14. Равенство множеств	213
9.7.15. Объединение множеств	213
9.7.16. Пересечение множеств	215
9.7.17. Разность множеств.....	215
9.7.18. Декартово произведение множеств	215
9.7.19. Множество всех подмножеств данного множества	216
9.7.20. Удаление всех повторов элементов в списке.....	216
9.7.21. Принадлежность множества списку подмножеств	216
9.7.22. Удаление всех повторов подмножеств в данном списке множеств.....	216
9.7.23. Удаление повторов атомов в списке списков атомов.....	217
9.7.24. Последовательное порождение нумерованных атомов.....	217
9.7.25. Программа построения сокращенной ДНФ по конъюнктивной нормальной форме	217
9.7.26. Программа построения сокращенной ДНФ по конъюнктивной нормальной форме без отрицаний.....	219
9.8. Некоторые встроенные предикаты Пролога.....	220
9.8.1. Средства управления	220
9.8.2. Классификация термов	221
9.8.3. Унификация термов	221
9.8.4. Сравнение термов.....	221
9.8.5. Арифметические функции	222
9.8.6. Арифметические предикаты	222
9.8.7. Обработка термов.....	223
9.8.8. Работа со строками.....	223
9.8.9. Составление списков	223
9.8.10. Взаимодействие с базой данных	224
9.8.11. Стандартные ввод и вывод	225
9.8.12. Доступ к файлам.....	226
9.8.13. Стандартный доступ к файлам в Прологе.....	226
9.8.14. Движение в файле.....	227
9.8.15. Исполнение системных функций	227
9.8.16. Отладчик (Debugger).....	227
Часть II. МОНАДИЧЕСКАЯ ЛОГИКА И КОНЕЧНЫЕ АВТОМАТЫ	230
Глава 10. Конечные автоматы	231
10.1. Автоматы Мили и Мура	231
10.2. Источники	235
10.3. Регулярные языки и регулярные выражения.....	240
10.3.1. Операции Клини и регулярные языки	240
10.3.2. Алгебра регулярных выражений Клини	242
10.4. Теоремы замкнутости для класса автоматов представимых языков	243

10.5. Минимизация числа состояний автомата с выходом	248
10.5.1. Склеивание неразличимых состояний.....	250
10.5.2. Алгоритм минимизации автомата.....	250
10.5.3. Алгоритм разбиения множества состояний на классы неотличимых состояний	254
10.5.4. Алгоритм проверки эквивалентности автоматов	255
Глава 11. Автоматы и сверхязыки	257
11.1. Макроавтоматы.....	257
11.2. Конкатенация языка и сверхязыка	259
11.3. Сверхитерация автоматных языков.....	261
11.4. Детерминизация макроисточника.....	264
Глава 12. Проблема униформизации	267
12.1. Языки и операторы	267
12.2. Игры.....	270
12.3. Стратегии	273
12.4. Униформизация конечноавтоматных языков	276
12.4.1. Порядковые векторы и порядковые стратегии	276
12.4.2. Теоремы о порядковых стратегиях.....	278
12.4.3. Пример построения выигрывающего автомата	282
Глава 13. Монадическая логика натуральных чисел	285
13.1. Монадическая логика	285
13.2. Выразимость в монадической логике	287
13.2.1. Макроисточники и монадическая логика	289
13.2.2. Регулярные языки и монадическая логика	289
13.2.3. Общерегулярные языки и монадическая логика	290
13.3. Специальная префиксная форма	290
13.4. Синтез автомата по формуле монадической логики.....	292
13.5. Алгоритмическая разрешимость монадической логики.....	294
Глава 14. Темпоральная логика.....	296
14.1. Пропозициональная темпоральная логика.....	296
14.2. Интерпретация формул.....	297
14.3. Общезначимость, выполнимость, опровержимость, невыполнимость	299
14.4. Другие темпоральные операторы.....	302
14.5. Аксиоматическая система	304
14.6. Спецификация свойств формулами.....	305
14.7. Спецификация взаимодействия и параллелизма.....	306

Глава 15. Аксиоматический язык программирования OBJ3	310
15.1. Описание языка	310
15.2. Спецификация объекта	311
15.3. Сорта и подсорта	312
15.4. Импорт модулей	312
15.5. Встроенные сорта	314
15.6. Декларация атрибутов	314
15.7. Приоритет	315
15.8. Параметризованное программирование	315
15.9. Теории	316
15.9.1. Программная спецификация FIELD алгебраического поля	316
15.9.2. Программная спецификация PROPC пропозиционального исчисления	317
15.9.3. Программная спецификация SET-NAT множества натуральных чисел	318
15.9.4. Программная спецификация obj WORDTREE дерева слов	319
15.9.5. Программная спецификация WORDSTACK словарного стека	320
15.9.6. View	321
15.9.7. Инстанциация	321
15.9.8. Параметризованная теория линейного векторного пространства	321
15.9.9. Параметризованный модуль obj ORD-PAIR пар вида (натуральное число, слово)	322
15.9.10. Параметризованный модуль SEQUENCE[X :: ELEMS] списков натуральных чисел и списков слов	323
Приложение 1. Логика высказываний и предикатов. Пролог	325
Приложение 2. Конечные автоматы	357
Приложение 3. Анализ конечных автоматов	364
Приложение 4. Синтез конечных автоматов	380
Литература	383
Обозначения	386

Предисловие



Философская логика есть наука о законах и формах отражения в мышлении развития объективного мира, о закономерностях познания истины.

Основная задача философской логики есть формулировка законов и принципов, соблюдение которых является необходимым условием получения истинных умозаключений.

Формальная логика есть наука, изучающая формы мысли (понятия, суждения, умозаключения, доказательства) со стороны их логической структуры, то есть отвлекаясь от конкретного содержания мыслей и вычлняя лишь общий способ связи частей этого содержания.

Начало формальной логики положил Аристотель.

Математическая логика есть область знания, в которой формальная логика изучается математическими методами.

Основные задачи математической логики есть: 1) построение формально-логических (аксиоматических) исчислений; 2) изучение связи логических исчислений с теми содержательными областями знаний, которые служат их интерпретациями и моделями.

Одно из крупнейших достижений математики первой половины XX века – оформление математической логики и теории алгоритмов в самостоятельные дисциплины. Три крупных результата определили характер всех последующих исследований этого направления: теорема Геделя о полноте аксиоматического исчисления предикатов относительно всех тождественно истинных формул такого исчисления (полнота относительно интерпретации); существование алгоритмически неразрешимых проблем, в частности теорема Черча об алгоритмической неразрешимости исчисления предикатов; теорема Геделя о неполноте аксиоматической арифметики относительно множества ее истинных формул.

Уже в древности предпринимались попытки строго изложить математические факты, которые стремились вывести из немногочисленных исходных посылок – аксиом. Замечательным примером такого подхода к изложению математических сведений были «Начала» древнегреческого математика Евклида, в особенности его геометрия. Изучались и законы правильного вывода следствий из исходных посылок (логика Аристотеля).

Первые попытки создать строгие математические теории восходят к работам Буля, Фреге, Пеано, других математиков. Исследователи руководствовались целью заложить такие основания математики, выделить такие аксиомы и правила вывода, с помощью которых можно было бы формально доказать любое содержательно истинное математическое утверждение. Подобные утверждения можно было бы доказывать автоматически. Эта связываемая с именем Гильберта программа в полной мере не удалась: австрийский мате-

матик Курт Гедель показал, что всякая достаточно богатая формальная система (например, аксиоматическая арифметика) не полна, т. е. в ней найдутся содержательно истинные утверждения, недоказуемые в системе. Пессимист угрюмо заметил бы: «Я же говорил вам, что из этой затеи Гильберта ничего хорошего не выйдет. Пойду наколю себе дров. Хотя я хорошо знаю заранее, что из этой моей затеи тоже ничего не выйдет». Оптимист скажет другое: «Мы надеялись на чудо. Его не произошло. Всевышний не дает нам того, чего мы очень хотим. Зато мы узнали много интересного, нам открылись огромные просторы для деятельности. О, сколько задач у нас впереди! За дело, коллеги!»

Аксиоматическая арифметика не полна. Тем не менее интерес к математической логике не убывает, исследования формально-логических систем продолжаются. Успешно описываются довольно большие фрагменты математических дисциплин. Использование алгебраического языка при проектировании узлов компьютера общеизвестно. В последнее время аппарат математической логики стал широко применяться в современных системах представления знаний.

Самое удивительное применение математическая логика нашла в вычислительной математике: формализм логического вывода в логике предикатов первого порядка был взят за основу при построении универсального языка программирования Пролог (акроним от PROgramming in LOGic). С появлением Пролог-подобных языков программирования математическая логика, которая совсем недавно была логикой сугубо теоретической, стала логикой вычисляющей. Классическая гильбертова логика доказательств истинных формул логики предикатов первого порядка для этой практической цели оказалась неподходящей. В 1965 г. Дж. А. Робинсон в качестве формализма, удобного для исполнения на компьютере, предложил использовать разработанный им метод резолюций. Замечательное открытие Робинсона оказалось необычайно плодотворным и уже в 70-х годах привело к построению универсального языка программирования Пролог и трансляторов для него. Это списковый язык для теоретико-множественных вычислений. Иногда его называют языком искусственного интеллекта. Задуманный как универсальный язык программирования для перевода с одного естественного языка на другой, Пролог оказался удобным инструментом при построении вычислительных моделей, возникающих при решении задач на графах, проектировании конечных автоматов, построении баз данных, баз знаний, экспертных систем, в задачах лингвистики при работе с естественными и искусственными языками (причем в числе последних могут выступать и языки программирования), в символьных преобразованиях, теории игр, системах представления знаний и т. д.

Несколько необычным в Прологе может показаться отсутствие привычного в традиционных языках программирования оператора присваивания. Вместо него в Прологе реализован более общий алгоритм унификации, построенный на основе операции сравнения с образцом. Необычен в Прологе также и отсутствующий в традиционных языках бэктрекинг, позволяющий обходить де-

рево вывода и собирать все возможные решения задачи. Реализация рекурсии в Прологе практически не отличается от общепринятой. В остальном Пролог довольно удобен, особенно в задачах, где основными обрабатываемыми объектами являются символ и список символов (множество).

Существует несколько версий трансляторов с Пролога: MicroProlog, DEC-System/Prolog, C-Prolog, IC-Prolog, M-Prolog, Sigma-Prolog, Chalcedony-Prolog, FF-Prolog, UNSW/Prolog, Prolog-6, Prolog-1 и Prolog-2 фирмы Expert Systems International, Quintus-Prolog, Turbo-Prolog, Arity-Prolog, SWI-Prolog, Visual-Prolog и другие. При написании программ мы ориентировались на версию языка SWI-Prolog (<http://www.swi-prolog.org/>).

Другое интересное направление применения математической логики есть разработка языков формальных спецификаций преобразований в алгебраических объектах и в анализе работы компьютерных программ.

Современный арсенал языков и инструментов, использующихся в данной области, известен как формальные методы разработки программ. Классическими методами и нотациями здесь являются VDM и Z. Если говорить о формальных моделях, то наиболее популярными являются Alloy, B и TLA. Среди средств моделирования и анализа программ на обычных языках программирования лидерами являются Isabelle (<http://isabelle.in.tum.de/>) и Frama-C (<http://frama-c.com/jessie.html>).

Для спецификаций преобразований в аксиоматически заданных алгебраических объектах, таких как полугруппы, группы, кольца, поля, линейные пространства и т. д., разработан функциональный язык программирования OBJ3. Сравнивая, например, OBJ3-программу для вычислений в полях, можно увидеть, что OBJ3-программа есть запись аксиом поля в терминах языка OBJ3. Вычисление преобразований в поле осуществляется в соответствии с аксиомами поля. OBJ3-программа моделирует эти преобразования. Иногда OBJ3 называют аксиоматическим языком программирования. Аксиоматический подход в языках программирования значительно облегчает задачу верификации программ. Предложенные в аксиоматических языках программирования подходы могут использоваться при создании других языков. Например, развитая в OBJ3 модульная система программ внедрена в языках программирования Ada, ML, C++, LOTOS (Language Of Temporal Ordering Specification). Параметризация из OBJ3 имплементирована в C#. Для решения аналогичных задач алгебраических спецификаций можно использовать родственный к OBJ3 язык программирования SafeOBJ из семейства OBJ (<https://cafeobj.org/>).

Книга написана по материалам лекций авторов по дисциплинам «Дискретная математика» и «Математическая логика и теория алгоритмов», читаемых на факультете бизнес-информатики, на факультете компьютерных наук Национального исследовательского университета Высшая школа экономики и на факультете автоматизации и вычислительной техники Национального исследовательского университета Московский энергетический институт. Эти курсы (или им аналогичные) начинали в МЭИ Д. А. Поспелов, В. Н. Вагин, В. П. Кутепов,

А. А. Болотов, А. Б. Фролов, Е. А. Щегольков, повлиявшие на выбор и характер излагаемого авторами материала.

Настоящая книга является второй в серии задуманных авторским коллективом книг по дискретной математике. Она предназначена студентам бакалавриата, изучающим академический курс «Дискретная математика».

В предлагаемой книге авторы сосредоточились на изложении основ математической логики и связанных с ней формальных языков, представимых конечными автоматами.

Основные теоретические и практические положения, изложение и анализ практических алгоритмов, иллюстрируемых большим числом примеров, позволяют сформировать прочную теоретическую базу, необходимую для дальнейшей работы практикующих программистов и ИТ-специалистов.

В приложении предлагаются задачи, которые могут быть использованы как для проведения практических занятий, так и для самостоятельной работы.

Авторы выражают глубокую благодарность рецензентам Калягину В. А., Петренко А. К. и научному редактору книги Захарову В. А. за замечания, позволяющие существенно улучшить качество книги. Мы также благодарны преподавателям департамента программной инженерии НИУ ВШЭ Ахметсафиной Р. З., Бересневой Е. Н., Горденко М. К., Гринкругу Е. М., Дворянскому Л. В., Дегтяреву К. Ю., Каленковой А. А., Ломазовой И. А., Подбельскому В. В., Шилову В. В., а также Амосову А. А., Вагину В. Н., Дубинскому Ю. А., Куликовой Н. Л., Фролову А. Б. из НИУ МЭИ за стимулирующие беседы. Авторы благодарят студентов Сапожкова Е. Д., Тимофееву Е. Э., Чичилеву Н. И. за активное участие в составлении и апробации задач и упражнений приложения. Авторы благодарны также корректору Синяевой Г. И. за внимательное прочтение рукописи и устранение ошибок.



Введение

1. Множество

Понятие множества неопределимо. Это простейшее исходное понятие человечество сформировало из опыта всего своего исторического развития. То же можно сказать о смысле простейшего отношения принадлежности: элемент a принадлежит множеству A (обозначение $a \in A$) – и о смысле отношения тождества (совпадения, равенства) двух элементов a и b из некоторого множества (обозначение $a = b$). Другими словами, предполагается, что читатель умеет распознавать совпадение или несовпадение двух элементов и устанавливать факт принадлежности или непринадлежности элемента множеству.

Пусть U есть некоторое множество. A есть подмножество множества U , если всякий элемент из множества A принадлежит множеству U . Множество U универсально (универсум), если все рассматриваемые множества есть подмножества множества U .

Пусть A, B, C есть произвольные подмножества множества U ; a, b, c есть элементы множества U . Обозначим символом $\emptyset$ пустое множество, то есть множество без элементов.

Основными неопределяемыми отношениями в теории множеств являются следующие отношения:

- $a = b$, элементы a и b равны (совпадают);
- $a \in A$, элемент a принадлежит множеству A .

Пусть знак $\leftrightarrow$ означает «если и только если»; а знаки $\&$, $\vee$, $\neg$, $\rightarrow$, $\forall$, $\exists$ есть логические знаки конъюнкции, дизъюнкции, отрицания, импликации, квантора общности и квантора существования. Используем их в общепринятом содержательном смысле. Знак $\exists!$ означает квантор существования единственного элемента.

Обозначим через $a \notin A$ отношение «элемент a не принадлежит множеству A » и через $a \neq b$ отношение «элементы a и b не равны (не совпадают)».

Введем далее следующие отношения.

- $A \subseteq B \leftrightarrow \forall a (a \in A \rightarrow a \in B)$, отношение включения множеств, при этом множество A называется подмножеством множества B , а множество B называется надмножеством множества A ;
- $A \supseteq B \leftrightarrow B \subseteq A$;
- $A = B \leftrightarrow A \subseteq B \& A \supseteq B$, отношение равенства множеств;
- $A \subset B \leftrightarrow A \subseteq B \& A \neq B$, отношение строгого включения множеств;
- $A \supset B \leftrightarrow B \subset A$.

Обозначим через $P(A)$ (или 2^A) множество всех подмножеств множества A . Введем следующие операции над множествами.

- $A \cup B = \{x \in U : x \in A \vee x \in B\}$, объединение множеств A и B ;
- $A \cap B = \{x \in U : x \in A \ \& \ x \in B\}$, пересечение множеств A и B ;
- $A - B = \{x \in U : x \in A \ \& \ x \notin B\}$, разность множеств A и B ;
- $\bar{A} = U - A$, дополнение ко множеству A ;
- $A \div B = (A \cup B) - (A \cap B)$, симметрическая разность множеств A и B ;
- $A \times B = \{(a, b) : a \in A \ \& \ b \in B\}$, декартово произведение множеств A и B .

Под натуральным числом понимаем количество элементов конечного множества. Количество элементов пустого множества есть 0.

Распространим декартово произведение на несколько сомножителей:

$$A_1 \times A_2 \times \dots \times A_n = \{(a_1, a_2, \dots, a_n) : a_1 \in A_1, a_2 \in A_2, \dots, a_n \in A_n\}.$$

Определим декартову степень множества:

$$A^n = A \times A \times \dots \times A \text{ (} n \text{ раз)}, A^0 = \emptyset.$$

Множества $\emptyset$ и A называются несобственными (тривиальными) подмножествами множества A . Если $A \subset B$ & $A \neq \emptyset$, то A есть собственное подмножество множества B .

Иногда пишут $A \cdot B$ или AB вместо $A \cap B$.



Примем следующие обозначения.

Множество натуральных чисел $\mathbb{N} = \{0, 1, 2, \dots\}$.

Множество положительных натуральных чисел $\mathbb{N}_+ = \{1, 2, \dots\}$.

Множество целых чисел $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$.

Множество $\mathbb{Z}_n = E_n = \{0, 1, 2, \dots, n-1\}$.

Множество рациональных чисел $\mathbb{Q} = \{\frac{m}{n} : m \in \mathbb{Z}, n \in \mathbb{N}_+\}$.

Множество вещественных чисел $\mathbb{R} = (-\infty, +\infty)$.

Множество неотрицательных вещественных чисел $\mathbb{R}_+ = [0, +\infty)$.

Множество комплексных чисел $\mathbb{C} = \{x + i y : x \in \mathbb{R}, y \in \mathbb{R}\}$, здесь $i^2 = -1$.



2. Функция

Определение. Пусть A и B есть два множества. Функция $f: A \rightarrow B$ есть отображение, которое каждому элементу x из A ставит в соответствие некоторый элемент y из B . Это обстоятельство записывается как $y = f(x)$.

Замечание. В этом определении функция f всюду определена. Частично определенная функция $f: A \rightarrow B$ есть отображение, которое каждому элементу из множества A сопоставляет не более одного элемента из множества B . Всяду определенная функция является частным случаем частично определенной функции.

Если $f(a) = b$, то элемент b есть образ элемента a , элемент a есть прообраз элемента b . Область определения функции f есть множество $D(f) = \{a \in A : \exists b \in B (f(a) = b)\}$. Область значений функции f есть множество $R(f) = \{b \in B : \exists a \in A (f(a) = b)\}$.

Иногда множество $R(f)$ обозначают как $Im(f)$ или $f(A)$. Полный прообраз элемента $b \in B$ есть множество $f^{-1}(b) = \{a \in A : f(a) = b\}$. Полный прообраз множества $C \subseteq B$ есть множество $f^{-1}(C) = \{a \in A : f(a) \in C\}$.

Сужение функции f , заданной на множестве A , на подмножество S множества A есть функция g – такая, что $\leftrightarrow a \in S (g(a) = f(a))$.

Расширение функции f , заданной на множестве A , на надмножество T множества A есть функция h – такая, что $\leftrightarrow a \in A (h(a) = f(a))$.

Функцию с конечной областью определения удобно задавать таблицей. Например, пусть множества $A = \{1, 2, 3, 4\}$, $B = \{1, 2, 3, 4, 5\}$, функция $f = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & & 1 & 2 \end{pmatrix}$.

Здесь $f(1) = 3$, $f(2)$ не определено, $f(3) = 1$, $f(4) = 2$. Порядок столбцов несуществен.

Область определения $D(f) = \{1, 3, 4\}$, область значений $R(f) = Im(f) = f(A) = \{1, 2, 3\}$.

Определение. Функция $\varphi: A \rightarrow B$ есть *взаимно-однозначное отображение* (1-1-отображение) между множествами A и B , если

- 1) $\forall b \in B \exists a \in A (f(a) = b)$,
- 2) $\forall a \in A \forall b \in A (a \neq b \rightarrow f(a) \neq f(b))$.

Замечание. Последнее условие можно заменить на условие

- 2') $\forall a \in A \forall b \in A (f(a) = f(b) \rightarrow a = b)$.

Функции $f: A \rightarrow B$ и $g: C \rightarrow D$ равны, если $A = C$, $B = D$, $\forall x \in A (f(x) = g(x))$.

Функция $I_A: A \rightarrow A$, для которой $\forall x \in A (I(x) = x)$, называется *тождественной* функцией.

Функция $f: A \rightarrow B$ есть *отображение в* (*инъективная функция*, или *инъекция*), если $\forall a \in A \forall b \in A$ условие $a \neq b$ влечет $f(a) \neq f(b)$.

Инъективная функция различные элементы из области определения переводит в различные элементы из области значений.

Функция $f: A \rightarrow B$ есть *отображение на* (*сюръективная функция*, или *сюръекция*), если область значений B совпадает с образом $f(A)$, то есть если $f(A) = B$.

Функция $f: A \rightarrow B$ есть *взаимно-однозначная функция* (или *биекция*), если f является отображением в и отображением на, то есть является одновременно инъективной и сюръективной функцией: 1) $a \neq b \rightarrow f(a) \neq f(b)$, 2) $Im(f) = B$.

Определение. Композиция $g \circ f$ функций $f: A \rightarrow B$ и $g: B \rightarrow C$ есть функция $g \circ f: A \rightarrow C$, для которой $\forall x \in A ((g \circ f)(x) = g(f(x)))$.

Замечание. Символ композиции $\circ$ иногда опускается.

Утверждение. $(h \circ g) \circ f = h \circ (g \circ f)$.

Доказательство. Пусть $f: A \rightarrow B$, $g: B \rightarrow C$, $h: C \rightarrow D$. Тогда $((h \circ g) \circ f)(x) = (hg)f(x) = (h \circ g)f(x) = h(g(f(x))) = h((g \circ f)(x)) = (h \circ (g \circ f))(x)$.

Замечание. Для тождественной функции $f \circ I_A = I_B \circ f = f$.

Определение. Функция $f^{-1}: B \rightarrow A$ называется *обратной* к функции $f: A \rightarrow B$, если $f \circ f^{-1} = I_B$ и $f^{-1} \circ f = I_A$.

Замечание. 1. g обратна к $f \leftrightarrow f$ обратна к g .

2. Функция $f: A \rightarrow B$ имеет обратную функцию, $\leftrightarrow$ функция f есть взаимно-однозначное отображение.

Утверждение. Если обратная функция для функции f существует, то она единственна.

Доказательство. Пусть функции f^{-1} и g обратны к функции $f: A \rightarrow B$. Тогда $f^{-1} \circ I_B = f^{-1} \circ (f \circ g) = (f^{-1} \circ f) \circ g = I_A \circ g = g$.

Следствие. Пусть для функций f и g существуют обратные функции f^{-1} и g^{-1} . Тогда справедливы утверждения:

1. $(f^{-1})^{-1} = f$.
2. $(f \circ g)^{-1} = g^{-1} \circ f^{-1}$.

Доказательство. 1. Так как f^{-1} обратна к f , то f обратна к f^{-1} , то есть $f = (f^{-1})^{-1}$.

2. $(f \circ g) \circ (g^{-1} \circ f^{-1}) = f \circ (g \circ g^{-1}) \circ f^{-1} = f \circ I_B \circ f^{-1} = f \circ f^{-1} = I_A$.

Аналогично $(g^{-1} \circ f^{-1}) \circ (f \circ g) = I_B$. Тогда функция $g^{-1} \circ f^{-1}$ обратна к $f \circ g$, то есть $(f \circ g)^{-1} = g^{-1} \circ f^{-1}$.

Теорема. Функция $f: A \rightarrow B$ имеет обратную функцию тогда и только тогда, когда отображение f взаимно-однозначно.

Доказательство. Пусть функция f имеет обратную функцию f^{-1} . Покажем, что отображение f взаимно-однозначно, то есть что $a \neq b \rightarrow f(a) \neq f(b)$ и $B = \text{Im}(f)$. В самом деле, пусть $f(a) = f(b)$. Тогда $a = I_A(a) = f^{-1}(f(a)) = f^{-1}(f(b)) = I_A(b) = b$, то есть $f(a) = f(b) \rightarrow a = b$, откуда $a \neq b \rightarrow f(a) \neq f(b)$.

Пусть $b \in B$. Тогда $b = I_B(b) = (f \circ f^{-1})(b) = f(f^{-1}(b))$, то есть всякий b есть образ некоторого $a = f^{-1}(b) \in A$. Поэтому $B = \text{Im}(f)$.

Пусть теперь f есть взаимно-однозначное отображение. Покажем, что функция f имеет обратную функцию. В самом деле, так как $B = \text{Im}(f)$, то каждый элемент b из B есть образ в точности одного элемента a из A : $f(a) = b$. Пусть $g(b) = a$. Для соответствия $g: B \rightarrow A$ имеем:

$$\begin{aligned} (g \circ f)(a) &= g(f(a)) = g(b) = a = I_A, \\ (f \circ g)(b) &= f(g(b)) = f(a) = b = I_B. \end{aligned}$$

Следовательно, g есть обратная функция для f . Теорема доказана.

3. Унарная функция

Напомним, что функция $f: A \rightarrow B$ есть правило (отображение), которое каждому элементу из множества A сопоставляет единственный элемент из множества B . Если $f(a) = b$, то элемент b есть образ элемента a , элемент a есть прообраз элемента b . Множество A есть область определения $D(f)$ функции f . Множество B есть область значений $V(f)$ функции f .

Образ $\text{Im } f = \{f(x) : x \in A\}$ отображения $f: A \rightarrow B$ есть множество $f(A)$ всех значений функции f . Полный прообраз элемента $y \in B$ есть множество $f^{-1}(y) = \{x \in A : f(x) = y\}$. Полный прообраз множества $C \subseteq B$ есть множество $f^{-1}(C) = \{x \in A : f(x) \in C\}$.

На конечном множестве функцию удобно задавать таблицей. Например, пусть множества $A = \{1, 2, 3, 4\}$, $B = \{1, 2, 3, 4, 5\}$, функция $f = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 2 & 1 & 2 \end{pmatrix}$. Здесь

$f(1) = 3, f(2) = 2, f(3) = 1, f(4) = 2$. Порядок столбцов несуществен. Область определения $D(f) = A = \{1, 2, 3, 4\}$, область значений $V(f) = B = \{1, 2, 3, 4, 5\}$, $\text{Im } f = f(A) = \{1, 2, 3\}$.

Функции $f: A \rightarrow B$ и $g: C \rightarrow D$ равны, если $A = C, B = D, \forall x \in A f(x) = g(x)$.

Функция $I_A: A \rightarrow A$, для которой $\forall x \in A I_A(x) = x$, называется *тождественной*.

Функция $f: A \rightarrow B$ есть *вложение* (*инъективная функция*, или *инъекция*), если $\forall a \in A \forall a' \in A$ условие $a \neq a'$ влечет $f(a) \neq f(a')$.

Инъективная функция различные элементы из области определения переводит в различные элементы из области значений.

Функция $f: A \rightarrow B$ есть *отображение на* (*сюръективная функция*, или *сюръекция*), если область значений B совпадает с образом $f(A)$, то есть если $f(A) = B$.

Функция $f: A \rightarrow B$ есть *взаимно-однозначная функция* (или *биекция*), если f является вложением и отображением на, то есть если 1) $a \neq a' \rightarrow f(a) \neq f(a')$, 2) $\text{Im } f = B$.

Если $f: A \rightarrow B$ и $C \subset A$, то функция $f: C \rightarrow B$ называется *сужением* функции f на множество C и обозначается $f|C$. Функция f называется *расширением* функции $f|C$.

Композиция $g \circ f$ функций $f: A \rightarrow B$ и $g: B \rightarrow C$ есть функция $g \circ f: A \rightarrow C$, для которой $\forall x \in A (g \circ f)(x) = g(f(x))$.

Символ композиции $\circ$ иногда опускается.

Утверждение. $(h \circ g) \circ f = h \circ (g \circ f)$.

Доказательство. Пусть $f: A \rightarrow B, g: B \rightarrow C, h: C \rightarrow D$. Тогда $((h \circ g) \circ f)(x) = (hg)f(x) = (h \circ g)f(x) = h(g(f(x))) = h((g \circ f)(x)) = (h \circ (g \circ f))(x)$.

Замечание. Для тождественной функции $f \circ I_A = I_B \circ f = f$.

Определение. Функция $f: B \rightarrow A$ называется *обратной* к функции $f: A \rightarrow B$, если $f \circ f^{-1} = I_B$ и $f^{-1} \circ f = I_A$.

Замечание. g обратна к $f \leftrightarrow f$ обратна к g .

Утверждение. Если обратная функция для функции f существует, то она единственна.

Доказательство. Пусть функции f^{-1} и g обратны к функции $f: A \rightarrow B$. Тогда $f^{-1} \circ I_B = f^{-1} \circ (f \circ g) = (f^{-1} \circ f) \circ g = I_A \circ g = g$.

Следствие. Пусть для функций f и g существуют обратные функции f^{-1} и g^{-1} . Тогда справедливы утверждения:

1. $(f^{-1})^{-1} = f$.
2. $(f \circ g)^{-1} = g^{-1} \circ f^{-1}$.

Доказательство.

1. Так как f^{-1} обратна к f , то f обратна к f^{-1} , то есть $f = (f^{-1})^{-1}$.

2. $(f \circ g) \circ (g^{-1} \circ f^{-1}) = f \circ (g \circ g^{-1}) \circ f^{-1} = f \circ I_B \circ f^{-1} = f \circ f^{-1} = I_B$.

Аналогично $(g^{-1} \circ f^{-1}) \circ (f \circ g) = I_A$. Тогда функция $g^{-1} \circ f^{-1}$ обратна к $f \circ g$, то есть $(f \circ g)^{-1} = g^{-1} \circ f^{-1}$.

Теорема. Функция $f: A \rightarrow B$ имеет обратную функцию тогда и только тогда, когда отображение f взаимно-однозначно.

Доказательство. Пусть функция f имеет обратную функцию f^{-1} . Покажем, что отображение f взаимно-однозначно, то есть что $a \neq a' \rightarrow f(a) \neq f(a')$ и $B = \text{Im } f$.

В самом деле, пусть $f(a) = f(a')$. Тогда $a = I_A(a) = f^{-1}(f(a)) = f^{-1}(f(a')) = I_A(a') = a'$, то есть $f(a) = f(a') \rightarrow a = a'$, откуда $a \neq a' \rightarrow f(a) \neq f(a')$.

Пусть $b \in B$. Тогда $b = I_B(b) = (f \circ f^{-1})(b) = f(f^{-1}(b))$, то есть всякий b есть образ некоторого $a = f^{-1}(b) \in A$. Поэтому $B = \text{Im } f$.

Пусть теперь f есть взаимно-однозначное отображение. Покажем, что функция f имеет обратную функцию. В самом деле, так как $B = \text{Im } f$, то каждый элемент b из B есть образ в точности одного элемента a из A : $f(a) = b$. Пусть $g(b) = a$. Для соответствия $g : B \rightarrow A$ имеем:

$$\begin{aligned}(g \circ f)(a) &= g(f(a)) = g(b) = a = I_A, \\ (f \circ g)(b) &= f(g(b)) = f(a) = b = I_B.\end{aligned}$$

Следовательно, g есть обратная функция для f . Теорема доказана.



4. Отношение

Пусть $A_1, A_2, \dots, A_n$ есть произвольные множества, вообще говоря, разнородные.

Определение. n -арное отношение r^n на множествах $A_1, A_2, \dots, A_n$ есть подмножество r^n декартова произведения $A_1 \times A_2 \times \dots \times A_n$.

Замечание. n -арное отношение r^n на множестве A есть подмножество r^n натуральной степени множества A^n , $n > 0$. Индекс n арности (местности) отношения r иногда опускается.

Возможна множественная (суффиксная) $(x_1, \dots, x_n) \in r$ и предикатная (префиксная) $\rho(x_1, \dots, x_n)$ формы записи отношений. В последнем случае отношение ρ называют также предикатом. Для бинарного отношения используется infix-запись $x \rho y$. Унарное отношение $\rho \subseteq E$ есть подмножество множества E . Предикат $\rho(x)$, соответствующий унарному отношению, называется свойством.

Набор $a = (a_1, a_2, \dots, a_n) \in r$ (допустима запись $\rho(a_1, a_2, \dots, a_n)$) называется элементом отношения.

Определение. Отношение конечно, если оно состоит из конечного числа элементов.

5. Отношение эквивалентности

Пусть A есть произвольное множество.

Определение. Бинарное отношение $\sigma \subseteq A \times A$ есть отношение эквивалентности (обозначение $a \sim b$), если оно удовлетворяет следующим аксиомам:

- 1) $a \sim a$, рефлексивность;
- 2) $a \sim b \rightarrow b \sim a$, симметричность;
- 3) $a \sim b \ \& \ b \sim c \rightarrow a \sim c$, транзитивность.

Обозначение. $a \sim b$, $\sigma(a, b)$, $(a, b) \in \sigma$, $a \sigma b$.

Определение. Разбиение I множества A есть семейство попарно непересекающихся непустых подмножеств множества A , таких, что: $A = \bigcup_{i \in I} A_i$, $\forall i \neq j$ $(A_i \cap A_j = \emptyset)$. Подмножества A_i называются смежными классами разбиения I .

Пример. $A = \{0, 1, 2, 3, 4, 5\} = \{0, 1, 5\} \cup \{2\} \cup \{3, 4\}$.

Теорема. 1. Каждому отношению эквивалентности, определенному на множестве A , соответствует некоторое разбиение множества A .

2. Каждому разбиению множества A соответствует некоторое отношение эквивалентности, определенное на множестве A .

Коротко: между классом всех определенных на множестве A эквивалентностей и классом всех разбиений множества A существует взаимно-однозначное соответствие.

Доказательство. 1. Пусть σ есть отношение эквивалентности, определенное на множестве A и $a \in A$. Построим множество $K_a = \{x \in A : x \sim a\}$ всех элементов x , эквивалентных a . Оно обозначается также через $[a]_\sigma$. Множества K_a называются *смежными классами A по σ* , или *классами эквивалентности*.

Заметим, что если $b \in K_a$, то $b \sim a$. Покажем, что $a \sim b \leftrightarrow K_a = K_b$. В самом деле, пусть $a \sim b$. Пусть произвольный элемент $c \in K_a$. Тогда $c \sim a$, $a \sim b$, $c \sim b$, $c \in K_b$, и потому $K_a \subseteq K_b$. Аналогично показываем, что $K_b \subseteq K_a$. Тогда $K_a = K_b$. Пусть теперь $K_a = K_b$. Тогда $a \in K_b$, и $a \sim b$. Утверждение доказано.

Если два класса K_a и K_b имеют общий элемент c , то они совпадают. В самом деле, если $c \in K_a$, $c \in K_b$, то $b \sim c$, $c \sim a$ и $b \sim a$, откуда $K_a = K_b$. Поэтому всякие два класса эквивалентности либо не пересекаются, либо (в случае непустого пересечения) совпадают. Всякий элемент c попадает в класс эквивалентности K_c . Поэтому система смежных классов есть разбиение множества A .

2. Пусть задано некоторое разбиение множества A . Определим на A отношение $\sim$, положив $a \sim b \leftrightarrow$ элементы a и b принадлежат одному и тому же классу разбиения. Отношение $\sim$ удовлетворяет аксиомам 1) $a \sim a$, 2) $a \sim b \rightarrow b \sim a$, 3) $a \sim b$ & $b \sim c$, и потому оно есть отношение эквивалентности.

Замечание. 1. Разбиение множества A на одноэлементные подмножества $A = \bigcup_{a \in A} \{a\}$ и разбиение A , состоящее из одного только множества A , называются *тривиальными (несобственными) разбиениями*.

2. Разбиение A на одноэлементные подмножества соответствует отношению эквивалентности, которое есть равенство.

3. Разбиение множества A , состоящее из одного только множества A , соответствует отношению эквивалентности, содержащему все множество $A \times A$.

4. $a \sigma b \leftrightarrow [a]_\sigma = [b]_\sigma$.

Определение. Совокупность классов эквивалентности множества A называется *фактор-множеством A/σ множества A по эквивалентности σ* .

Определение. Отображение $p : A \rightarrow A/\sigma$, при котором $p(a) = [a]_\sigma$, называется *каноническим (естественным)*.

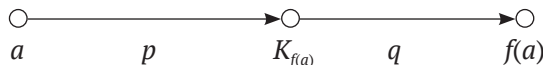
6. Каноническое разложение функции

Пусть $f : A \rightarrow B$ есть некоторая функция. Определим на A отношение $\sigma \in A \times A$, положив $\forall a \in A \forall b \in A (a \sim b \leftrightarrow f(a) = f(b))$. Отношение σ есть отношение эквивалентности, так как выполняются следующие свойства:

- 1) $a \sim a$, ибо $f(a) = f(a)$;
- 2) $a \sim b \rightarrow b \sim a$, ибо $f(a) = f(b) \rightarrow f(b) = f(a)$;
- 3) $a \sim b \ \& \ b \sim c \rightarrow a \sim c$, ибо $f(a) = f(b) \ \& \ f(b) = f(c) \rightarrow f(a) = f(c)$.

Введенное отношение σ называется ядерной эквивалентностью для отображения f . Классы эквивалентности A/σ есть полные прообразы элементов множества B при отображении f , то есть $A_b = f^{-1}(b)$.

Отображение f можно разложить в композицию двух отображений согласно следующему рисунку:



Имеет место равенство $f = q \circ p$, то есть $f(a) = q(p(a))$.

Представление $f = q \circ p$ называется каноническим разложением (представлением) функции f .

Пример. Получить каноническое разложение функции

$$f: E_{10} \rightarrow E_{10}, f = 0112105533 = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 0 & 1 & 1 & 2 & 1 & 0 & 5 & 5 & 3 & 3 \end{pmatrix}.$$

Область определения $D(f) = E_{10}$. Область значений $Im(f) = \{0, 1, 2, 3, 5\}$. Классы эквивалентности:

$$\begin{aligned} K_0 &= [0]_\sigma = f^{-1}(0) = \{0, 5\}, q(K_0) = 0, \\ K_1 &= [1]_\sigma = f^{-1}(1) = \{1, 2, 4\}, q(K_1) = 1, \\ K_2 &= [2]_\sigma = f^{-1}(2) = \{3\}, q(K_2) = 2, \\ K_3 &= [3]_\sigma = f^{-1}(3) = \{8, 9\}, q(K_3) = 3, \\ K_5 &= [5]_\sigma = f^{-1}(5) = \{6, 7\}, q(K_5) = 5. \end{aligned}$$

Функции p и q задаются следующим образом:

$$p(a) = K_{f(a)} = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ K_0 & K_1 & K_2 & K_3 & K_4 & K_5 & K_6 & K_7 & K_8 & K_9 \end{pmatrix},$$

$$D(p) = E_{10}, Im(p) = \{K_0, K_1, K_2, K_3, K_5\}; q(K_a) = a = \begin{pmatrix} K_0 & K_1 & K_2 & K_3 & K_5 \\ 0 & 1 & 2 & 3 & 5 \end{pmatrix},$$

$$D(q) = \{K_0, K_1, K_2, K_3, K_5\}, Im(q) = \{0, 1, 2, 3, 5\}; f(a) = q(p(a)).$$

7. Мощность множества. Счетные и несчетные множества

Определение. Множества A и B эквивалентны ($A \sim B$), если между их элементами можно установить взаимно-однозначное соответствие. $\square$

Отношение эквивалентности множеств обладает следующими свойствами.

1. $A \sim A$, рефлексивность.
2. $A \sim B \rightarrow B \sim A$, симметричность.
3. $A \sim B \ \& \ B \sim C \rightarrow A \sim C$, транзитивность.

Определение. *Мощность множества A (обозначение $|A|$) есть класс эквивалентных ему множеств. Мощность конечного множества есть число его элементов.*

Замечание. Эквивалентные множества A и B равномощны, то есть $A \sim B \leftrightarrow |A| = |B|$.

Определение. Множество A *сечно*, если A эквивалентно множеству $\mathbb{N}$ натуральных чисел. В противном случае множество A *несечно*.

Утверждение. Из всякого бесконечного множества можно выделить счетное подмножество.

Доказательство. Пусть A есть бесконечное множество. Выделим в A произвольный элемент a_0 . Множество $A - \{a_0\}$ бесконечно. Выделим в нем элемент a_1 . Множество $A - \{a_0, a_1\}$ бесконечно. Выделим в нем элемент a_2 . И так далее. В бесконечном множестве A выделено счетное подмножество $B = \{a_0, a_1, a_2, \dots\}$.

Утверждение. Множество $\mathbb{Q}_+$ положительных рациональных чисел сечно.

Доказательство. Расположим элементы из $\mathbb{Q}_+$ в следующей таблице.

1, 1/2, 1/3, 1/4, 1/5, ...
 2, 2/2, 2/3, 2/4, 2/5, ...
 3, 3/2, 3/3, 3/4, 3/5, ...
 4, 4/2, 4/3, 4/4, 4/5, ...
 ...

Выписываем элементы из $\mathbb{Q}_+$ по диагонали, сверху вниз, выпуская ранее встречавшиеся числа: 1, 1/2, 2, 1/3, 3, 2/3, ... Следовательно, множество $\mathbb{Q}_+$ сечно.

Утверждение. Объединение конечного или счетного множества счетных множеств сечно.

Доказательство. Расположим элементы множеств $A_1, A_2, A_3, \dots$ (их число может быть и конечным) в следующей таблице.

$A_1 : a_{11}, a_{12}, a_{13}, a_{14}, \dots$
 $A_2 : a_{21}, a_{22}, a_{23}, a_{24}, \dots$
 $A_3 : a_{31}, a_{32}, a_{33}, a_{34}, \dots$
 $A_4 : a_{41}, a_{42}, a_{43}, a_{44}, \dots$
 ...



Выписываем элементы из $A_1 \cup A_2 \cup A_3 \cup \dots$ по диагонали, сверху вниз, выпуская ранее встречавшиеся элементы: $a_{11}, a_{12}, a_{21}, a_{13}, a_{22}, a_{31}, \dots$ Следовательно, множество $A_1 \cup A_2 \cup \dots$ сечно.

Замечание. Объединение конечного множества и счетного множества сечно. Множество рациональных чисел сечно, ибо $\mathbb{Q} = \mathbb{Q}_- \cup \mathbb{Q}_+ \cup \{0\}$, где $\mathbb{Q}_-$ есть множество отрицательных рациональных чисел.

8. Мощность континуума

Утверждение. Множество C всех бесконечных последовательностей из 0 и 1 несчетно.

Доказательство. Допустим противное: существует пересчет всех бесконечных последовательностей $A_1, A_2, A_3, \dots$ из 0 и 1:

$$A_1 : a_{11}, a_{12}, a_{13}, a_{14}, \dots$$

$$A_2 : a_{21}, a_{22}, a_{23}, a_{24}, \dots$$

$$A_3 : a_{31}, a_{32}, a_{33}, a_{34}, \dots$$

$$A_4 : a_{41}, a_{42}, a_{43}, a_{44}, \dots$$

...

Построим последовательность $B : b_1, b_2, b_3, \dots$, где

$$b_i = \begin{cases} 1, & \text{если } a_{ii} = 0, \\ 0, & \text{если } a_{ii} = 1, \end{cases} \quad i = 1, 2, 3, \dots$$

Последовательность B лежит вне указанного пересчета. B отличается от A_1 элементом $b_1 \neq a_{11}$, от A_2 – элементом $b_2 \neq a_{22}$, от A_3 – элементом $b_3 \neq a_{33}$ и т. д. Следовательно, исходное множество C несчетно.

Определение. Множество A имеет *мощность континуума* c , если A эквивалентно множеству всех бесконечных последовательностей из 0 и 1.

Следствие. Множество C всех бесконечных последовательностей из 0 и 1 имеет мощность континуума: $|C| = c$ (в силу рефлексивности).

Утверждение. Множество $P(\mathbb{N})$ всех подмножеств множества натуральных чисел имеет мощность континуума.

Доказательство. Всякую бесконечную последовательность из 0 и 1 можно рассматривать как характеристическую функцию некоторого подмножества множества натуральных чисел. Следовательно, множество $P(\mathbb{N})$ имеет мощность континуума: $|P(\mathbb{N})| = c$.

Следствие. Множество всех подмножеств множества натуральных чисел несчетно.

Утверждение. Если к бесконечному множеству добавить конечное или счетное множество элементов, то его мощность не изменится.

Доказательство. Пусть A есть бесконечное множество, а B есть конечное или счетное множество, причем $A \cap B = \emptyset$. Покажем, что $A \sim A \cup B$. Выделим из множества A счетное подмножество A_1 . Тогда $A = C \cup A_1$, где $C = A - A_1$, и $A \cup B = (C \cup A_1) \cup B = C \cup (A_1 \cup B)$. Так как $A_1 \cup B \sim A_1$, то $A \cup B = C \cup (A_1 \cup B) \sim C \cup A_1$ в силу транзитивности, и с учетом того, что $A = C \cup A_1$, получаем $A \cup B \sim A$.

Утверждение. Если A есть несчетное множество, а B есть конечное или счетное его подмножество, то $A - B \sim A$.

Доказательство. Пусть $C = A - B$. Тогда $A = C \cup B$. Множество C несчетно, ибо в противном случае C конечно или счетно, и тогда $A = C \cup B$ конечно или счетно. Множество $C \cup B \sim C$, или $A \sim C$, то есть $A \sim A - B$.

Теорема. Множество $U = [0, 1]$ имеет мощность континуума c .

Доказательство. Множество U эквивалентно множеству всех последовательностей из 0 и 1.

Замечание. 1. $|[0, 1]| = |(0, 1)| = |[0, 1)| = |[0, 1]| = c$.

2. Если $a < b$, то $|[a, b]| = c$, ибо функция $y = a + x(b - a)$ отображает $[0, 1]$ на $[a, b]$ взаимно-однозначно.

3. $|[a, b]| = |(a, b)| = |(a, b]| = |[a, b)| = c$.

4. $|(-\infty, \infty)| = |\mathbb{R}| = c$, ибо функция $y = \operatorname{tg}(x)$ отображает интервал $(a, b) = (-\pi/2, \pi/2)$ на всю числовую ось $\mathbb{R}$ взаимно-однозначно.

9. Кардинальные числа. Сравнение мощностей

Определение. *Мощность множества* есть класс эквивалентных между собой множеств. *Кардинальное число*, или кардинал, есть знак (символ), приписываемый мощности как классу эквивалентных между собой множеств. Мощности конечных множеств называются *финитными кардиналами*. Мощности бесконечных множеств называются *трансфинитными кардиналами*.

Пример. Счетной мощности (мощность множества натуральных чисел) присваивается кардинальное число $\aleph_0$ (алеф-нуль). Мощности множества вещественных чисел присваивается кардинальное число c .

Замечание. Мощность множества A обозначается через $|A|$, а также через $\operatorname{card}(A)$ или $c(A)$. Мощность конечного множества есть число его элементов.

Пусть A, B есть произвольные множества и $|A|, |B|$ есть их мощности.

Априори возможны четыре случая.

1. Множество A эквивалентно некоторому подмножеству множества B , а множество B эквивалентно некоторому подмножеству множества A .

2. Множество A эквивалентно некоторому подмножеству множества B , а множество B не эквивалентно никакому подмножеству множества A .

3. Множество B эквивалентно некоторому подмножеству множества A , а множество A не эквивалентно никакому подмножеству множества B .

4. Множество A не эквивалентно никакому подмножеству множества B , а множество B не эквивалентно никакому подмножеству множества A .

Определение.

$|A| = |B|$, если $A \sim B$.

$|A| \leq |B|$, если A эквивалентно некоторому подмножеству в B .

$|A| < |B|$, если A эквивалентно некоторому подмножеству в B , а множество B не эквивалентно никакому подмножеству множества A .

$|A| \geq |B|$, если $|B| \leq |A|$.

$|A| > |B|$, если $|B| < |A|$.

Замечание. Случай, когда множество A не эквивалентно никакому подмножеству множества B , а множество B не эквивалентно никакому подмножеству множества A , невозможен.

Теорема (Кантор–Бернштейн). Если множество A эквивалентно некоторому подмножеству B_1 множества B , а множество B эквивалентно некоторому под-

множеству A_1 множества A , то множества A и B эквивалентны (то есть имеют равную мощность). Коротко: $|A| \leq |B|$ & $|B| \leq |A| \rightarrow |A| = |B|$.

Доказательство. Случаи $B_1 = B$ и $A_1 = A$ можно исключить, ибо если $B_1 = B$, то условие теоремы утверждает, что $A \sim B$, из чего, естественно, следует $A \sim B$. Случай $A_1 = A$ аналогичен.

Итак, пусть $A_1 \subset A$, $B_1 \subset B$. Пусть функции $f: A \rightarrow B_1$, $g: B \rightarrow A_1$ устанавливают взаимно-однозначное соответствие между A и B_1 и между B и A_1 , то есть $A \leftrightarrow_f B_1$, $B \leftrightarrow_g A_1$. С помощью функций f и g расслоим множества A и B на «кольца» следующим образом. Имеем:

$$\begin{aligned} A &\leftrightarrow_f B_1 \subset B, B \leftrightarrow_g A_1 \subset A, \\ A_1 &\leftrightarrow_f B_2 \subset B_1, B_1 \leftrightarrow_g A_2 \subset A_1, \\ A_2 &\leftrightarrow_f B_3 \subset B_2, B_2 \leftrightarrow_g A_3 \subset A_2, \\ &\dots \end{aligned}$$



Сформируем множества («кольца»):

$$\begin{aligned} K_0^A &= A - A_1, K_0^B = B - B_1, \\ K_1^A &= A_1 - A_2, K_1^B = B_1 - B_2, \\ K_2^A &= A_2 - A_3, K_2^B = B_2 - B_3, \\ &\dots \end{aligned}$$

Функции f и g устанавливают следующие взаимно-однозначные соответствия:

$$\begin{aligned} K_0^A &\leftrightarrow_f K_1^B, K_0^B \leftrightarrow_g K_1^A, \\ K_2^A &\leftrightarrow_f K_3^B, K_2^B \leftrightarrow_g K_3^A, \\ &\dots \end{aligned}$$

Пусть множества $C = \bigcap_{i=1}^{\infty} A_i$, $D = \bigcap_{i=1}^{\infty} B_i$. Функции f и g устанавливают взаимно-однозначные соответствия между C и D . Если бы это было не так, то возникли бы аналогичные кольца в C и D , что по построению C и D невозможно.

Пусть множества

$$\begin{aligned} A_{\text{чет}} &= \bigcap_{i=1}^{\infty} K_{2i}^A = K_0^A \cup K_2^A \cup K_4^A \cup \dots \\ A_{\text{неч}} &= \bigcap_{i=1}^{\infty} K_{2i+1}^A = K_1^A \cup K_3^A \cup K_5^A \cup \dots \end{aligned}$$

Аналогично построим множества $B_{\text{чет}}$, $B_{\text{неч}}$. Тогда

$$A = A_{\text{чет}} \cup A_{\text{неч}} \cup C, B = B_{\text{чет}} \cup B_{\text{неч}} \cup D.$$

Функция f устанавливает взаимно-однозначные соответствия:

$$A_{\text{чет}} \leftrightarrow_f B_{\text{неч}}, A_{\text{неч}} \leftrightarrow_g B_{\text{чет}}, C \leftrightarrow_{f \text{ или } g} D.$$

Тогда функция $h : A \rightarrow B$, определенная как

$$h(x) = \begin{cases} f(x), & \text{если } x \in A_{\text{чет}} \cup C \\ g(x), & \text{если } x \in A_{\text{неч}} \end{cases},$$

устанавливает взаимно-однозначное соответствие между A и B . Следовательно, $|A| = |B|$. Теорема доказана.

Следствие. Если $A \subseteq B$, то $|A| \leq |B|$.

Кардинальные числа можно сравнивать по величине.

Пусть A есть некоторое множество и $P(A)$ есть множество всех подмножеств множества A . Очевидно, что $|A| \leq |P(A)|$, ибо взаимно-однозначное соответствие между A и частью $P(A)$ устанавливается, если каждому элементу a из A сопоставить одноэлементное множество $\{a\}$ из $P(A)$.

Теорема (Кантор). $|A| < |P(A)|$.

Доказательство. Покажем, что $|A| \neq |P(A)|$. Допустим противное: $|A| = |P(A)|$ для некоторого множества A . Тогда существует взаимно-однозначное соответствие $f : A \rightarrow P(A)$ между множествами A и $P(A)$. Пусть

$$A_1 = \{a \in A : a \in f(a)\}, A_2 = \{a \in A : a \notin f(a)\}.$$

Тогда $A_2 = A - A_1$. Множество $A_2 \in P(A)$. Пусть в нашем соответствии $f(b) = A_2$ для некоторого b из A . Каждый элемент из A попадает либо в A_1 , либо в A_2 . Если $b \in A_1$, то по построению A_1 будет $b \in f(b)$ и $f(b) = A_2$. Противоречие, ибо $b \in A_1$ и $b \in A_2$, что одновременно невозможно. Если $b \in A_2$, то по построению A_2 будет $b \notin f(b)$ и $f(b) = A_2$. Противоречие, ибо $b \in A_2$ и $b \notin A_2$, что одновременно невозможно. Следовательно, наше предположение о равенстве $|A|$ и $|P(A)|$ не верно. Остается взять $|A| \neq |P(A)|$, а так как $|A| \leq |P(A)|$, то $|A| < |P(A)|$. Теорема доказана.

Иногда множество $P(A)$ всех подмножеств множества A обозначается через 2^A , а мощность $P(A)$ через $2^{|A|}$. Тогда по теореме $|A| < 2^{|A|}$.

Отправляясь от произвольного множества A , по теореме Кантора можно построить возрастающую последовательность кардинальных чисел:

$$|A| < 2^{|A|} < 2^{2^{|A|}} < \dots$$

Отправляясь от счетного множества $\mathbb{N}$ натуральных чисел, можно построить возрастающую последовательность кардиналов:

$$\aleph_0 < 2^{\aleph_0} = c = \aleph_1 < 2^{\aleph_1} = \aleph_2 < 2^{\aleph_2} = \aleph_3 < \dots$$

Мощности $\aleph_0, \aleph_1 = c, \aleph_2 = 2^c, \aleph_3 = 2^{2^c}, \dots$ – это счетная мощность, континуум, гиперконтинуум, гипергиперконтинуум и т. д.

Кантор поставил проблему о существовании промежуточной мощности между $\aleph_0$ и $\aleph_1$ (континуум-гипотеза) и промежуточных мощностей между всякими $\aleph_i$ и $\aleph_{i+1}$ (обобщенная континуум-гипотеза). В работах К. Геделя и П. Козна было установлено, что обе гипотезы не противоречат аксиоматической теории множеств (существует модель, в которой истинны аксиомы теории множеств, континуум-гипотеза, причем правила вывода сохраняют истинность выводи-

мых формул) и не могут быть в ней доказаны (существует модель, в которой истинны аксиомы теории множеств и ложна континуум-гипотеза, причем правила вывода сохраняют истинность выводимых формул, а потому континуум-гипотеза не может быть доказана в теории множеств). Отсюда следует, что обе гипотезы независимы в аксиоматической теории множеств.





АЛГЕБРА ЛОГИКИ И ПРЕДИКАТЫ

В результате освоения учебного материала данной части студент должен:

- знать основные понятия и методы математической логики; функции алгебры логики; логику предикатов; формально-аксиоматические системы для логики высказываний и логики предикатов; теоремы Геделя о формальном выводе; принципы построения логических языков программирования на основе формализма метода резолюций и аксиоматических языков программирования; методику работы с математической литературой и методику самостоятельного изучения новых логических аксиоматических систем;
- уметь анализировать и представлять функции алгебры логики в нормальных дизъюнктивных и конъюнктивных нормальных формах; анализировать формулы логики предикатов на предмет их общезначимости, выполнимости, опровержимости, невыполнимости; анализировать правила вывода формул логики высказываний и логики предикатов и проверять их правильность; программировать на одной из версий языка программирования Пролог и на одной из версий аксиоматического языка программирования; применять свои знания к решению практических задач;
- иметь навыки минимизации функции алгебры логики в классе нормальных форм; навыки анализа данной системы функций алгебры логики на функциональную полноту; навыки упрощения формул логики предикатов; навыки математического описания явлений и процессов, используя элементы математической логики; навыки расширения своих знаний в применении математической логики при разработке алгоритмов решения прикладных задач программной инженерии.

Глава 1

Алгебра логики

1.1. Функции алгебры логики

Пусть $E_2 = \{0, 1\}$ есть двухэлементное множество. Набор длины n из 0 и 1 есть последовательность длины n , составленная из 0 и 1.

Пример. (0); (1) есть наборы длины 1;

(0,0); (0,1); (1,0); (1,1) есть наборы длины 2;

(0,0,0); (0,0,1); (0,1,0); (0,1,1); (1,0,0); (1,0,1); (1,1,0); (1,1,1) есть наборы длины 3;

(0,0,...,0,0); (0,0,...,0,1); ...; ($a_1, a_2, \dots, a_n$); ...; (1,1,...,1,1) есть наборы длины n .

Пусть E_2^n есть множество всех наборов длины n из 0 и 1.

Теорема. Число $h(n)$ всех наборов длины n из 0 и 1 равно 2^n .

Доказательство. Индукция по n .

Базис. $n = 1$. $h(1) = 2$.

Предположение индукции. Пусть $h(n) = 2$.

Шаг индукции. Покажем, что $h(n+1) = 2^{n+1}$. Разобьем все наборы длины $n+1$ на два класса: класс наборов, начинающихся с 0, и класс наборов, начинающихся с 1.

0,0,...,0,0	1,0,...,0,0
0,0,...,0,1	1,0,...,0,1
0,0,...,1,0	1,0,...,1,0
...	...
0,1,...,1,1	1,1,...,1,1



Число всех наборов длины $n+1$, начинающихся с 0 (так же как и число всех наборов, начинающихся с 1), равно числу всех наборов длины n и по предположению индукции равно 2^n . Тогда $h(n+1) = 2^n + 2^n = 2 \cdot 2^n = 2^{n+1}$. Теорема доказана.

Определение. Функция алгебры логики есть функция, аргументы и значения которой принимают лишь два значения: 0 и 1.

Класс всех функций алгебры логики обозначим через P_2 . Вместо слов «функция алгебры логики» будем иногда говорить просто «функция».

Теорема. Число всех n -местных функций алгебры логики равно 2^{2^n} .

Доказательство. В табл. 1.1 перечислены все n -местные функции.

Таблица 1.1

x_1	x_2	...	x_{n-1}	x_n	f_0	f_1	f_2	...	f_r
0	0	...	0	0	0	0	0	...	1
0	0	...	0	1	0	0	0	...	1
		...						...	
1	1	...	1	0	0	0	1	...	1
1	1	...	1	1	0	1	0	...	1

Число всех строк равно 2^n , т. е. равно числу всех наборов длины n из 0 и 1. Число всех функций алгебры логики от n переменных равно 2^{2^n} , т. е. равно числу всех наборов длины 2^n из 0 и 1.

Теорема доказана.

В табл. 1.2 приведены некоторые часто используемые в практике функции:

- 0, константа нуль;
- 1, константа единица;
- $\underline{x} \equiv 1$, тождественная функция;
- x , отрицание;
- $x \vee y$, дизъюнкция;
- $x \& y$, конъюнкция (обозначается также через $x \cdot y$ или xy);
- $x \rightarrow y$, импликация;
- $x + y$, сложение (по модулю два);
- $x \equiv y$, эквивалентность (равносильность);
- $x | y$, штрих Шеффера;
- $x \uparrow y$, стрелка Пирса.

Таблица 1.2

x	y	0	1	$x \vee y$	$x \& y$	$x \rightarrow y$	$x + y$	$x \equiv y$	$x y$	$x \uparrow y$	x	$\neg x$	$f(x)$ $= x$	0	1
0	0	0	1	0	0	1	0	1	1	1	0	1	0	0	1
0	1	0	1	1	0	1	1	0	1	0	1	0	1	0	1
1	0	0	1	1	0	0	1	0	1	9					
1	1	0	1	1	1	1	0	1	0	0					

1.2. Формулы. Реализация функций формулами

Пусть $F = \{f_1^{n_1}, f_2^{n_2}, \dots\}$ есть множество функциональных символов указанной сверху местности и $\{x_1, x_2, \dots\}$ есть множество символов переменных. Верхние индексы $n_1, n_2, \dots$ могут опускаться, если их значение предполагается известным.

Пусть F есть некоторое подмножество функций из P_2 .

Определение. Формула (алгебры логики) над F определяется индуктивно следующим образом.

1. Символ переменной есть формула над F .

2. Если f есть функциональный символ арности m из F и $x_1, \dots, x_m$ есть символы различных переменных, то выражение $f(x_1, \dots, x_m)$ есть формула над F и $\{x_1, \dots, x_m\}$ есть множество ее переменных.

3. Если $A(x_1, \dots, x_m)$ есть формула над F , где $\{x_1, \dots, x_m\}$ есть множество ее переменных, и если каждое из выражений $A_1, \dots, A_m$ есть либо формула над F , либо символ переменной, то выражение $A(A_1, \dots, A_m)$ есть формула над F , причем множество ее переменных есть объединение множеств переменных формул $A_1, \dots, A_m$.

Примеры. $F = \{f_1(x, y), f_2(x, y, z), f_3(x)\}$. Следующие выражения являются формулами над F .

$$f_1(x, y); f_2(x, y, z); f_3(x); f_1(t, z); f_2(t, t, t); f_3(f_2(x, y, x)); f_1(f_2(y, t, z), f_3(f_1(x, x))).$$

Сопоставим каждому функциональному символу f^n из F некоторую функцию $f: E_2^n \rightarrow E_2$. Если множество функциональных символов $F = \{f_1^{n_1}, f_2^{n_2}, \dots\}$, то обозначим множество соответствующих функций $F = \{f_1^{n_1}, f_2^{n_2}, \dots\}$. Между F и F существует взаимно-однозначное соответствие, при котором $f_i^{n_i}$ соответствует $f_i^{n_i}$, $i = 1, 2, \dots$

Каждой формуле над F сопоставим функцию следующим образом.

1. Переменной x сопоставим тождественную функцию x .
2. Формуле $f(x_1, \dots, x_m)$ над F сопоставим функцию $f(x_1, \dots, x_m)$ из F .
3. Если формуле $A(x_1, \dots, x_m)$ над F сопоставлена функция $f(x_1, \dots, x_m)$, а формулам $A_1, \dots, A_m$ над F сопоставлены функции $f_1, \dots, f_m$, то формуле $A(A_1, \dots, A_m)$ над F сопоставим функцию $f(f_1, \dots, f_m)$ из P_2 .

Таким образом, каждая формула над F реализует некоторую функцию из P_2 . Пусть формула $A(x_1, \dots, x_n)$ реализует некоторую функцию $f(x_1, \dots, x_n)$, и пусть $a = (a_1, \dots, a_n)$ есть набор длины n из 0 и 1. Тогда значение формулы A на наборе a есть $f(a)$, т. е. $A(a_1, \dots, a_n) = f(a_1, \dots, a_n)$.

В дальнейшем формулу $A(x_1, \dots, x_m)$ будем отождествлять с функцией $f(x_1, \dots, x_m)$, которую формула A реализует, и обозначать эту функцию через $A(x_1, \dots, x_m)$. Говоря о формуле A над F , будем говорить просто о формуле A , не упоминая об F , если из контекста ясно, о каком множестве F идет речь.

Имея в виду взаимно-однозначное соответствие между F и F , вместо слов «формула $A(x_1, \dots, x_m)$ над множеством функциональных символов F » будем говорить «формула $A(x_1, \dots, x_m)$ над множеством функций F ».

Пример. Формула $xy \vee \bar{x}$ реализует функцию $f(x, y)$, приведенную в табл. 1.3.

Таблица 1.3

x	y	$f(x, y)$
0	0	0
0	1	0
1	0	1
1	1	0

Определение. Функция f есть суперпозиция над F , если f реализуется некоторой формулой над F .

Определение. Пусть A есть некоторая формула над множеством F функций из P_2 . Если A есть $f(x_1, \dots, x_m)$ из F , то единственной подформулой формулы A является она сама. Если A есть формула $f(A_1, \dots, A_m)$, где $f \in F$, а $A_1, \dots, A_m$ есть некоторые формулы над F , то подформулами формулы A являются она сама и все подформулы формул $A_1, \dots, A_m$.

Замечание. Пусть $A(B)$ означает, что B есть подформула формулы A .

Везде далее слова множество, система, класс будем считать синонимами.

Определение. Класс функций F называется *функционально замкнутым*, если вместе с любыми своими функциями он содержит и любую их суперпозицию.

Определение. Множество функций $[F]$ называется *замыканием класса функций* F , если оно содержит все суперпозиции функций над множеством F и не содержит никаких других функций.

Замечание. 1. $F \subseteq [F]$.

2. $[[F]] = [F]$.

3. $F_1 \subseteq F_2$ влечет $[F_1] \subseteq [F_2]$.

4. Множество функций F замкнуто, если $[F] = F$.

Определение. Система G функций из замкнутого класса F *полна* в F (является порождающей системой для F), если $[G] = F$. Система функций H *полна* (в P_2), если $[H] = P_2$. Полная в F система функций G называется *базисом* в F , если никакая собственная подсистема в G не является полной в F .

1.3. Равносильные преобразования формул

Пусть A_1 и A_2 есть формулы, а $x_1, \dots, x_n$ есть полный список их переменных. Формулы A_1 и A_2 называются *равносильными* (*равными*, *эквивалентными*), если для любого набора значений аргументов $x_1, \dots, x_n$ они принимают одинаковые значения.

Пример. $A_1(x, y) = \overline{xy \vee \bar{x}}$; $A_2(x, y) = \overline{x \rightarrow y}$; $A_3(x) = x$; $A_4(x, y) = x \vee y$. В табл. 1.4 приведены значения формул $A_1 - A_4$, из чего видно, что $A_1 = A_2$, $A_2 \neq A_3$; $A_2 \neq A_4$; $A_3 \neq A_4$.

Таблица 1.4

$x \ y$	A_1	A_2	A_3	A_4
0 0	0	0	0	0
0 1	0	0	0	1
1 0	1	1	1	1
1 1	0	0	1	1

В инженерной практике наиболее распространены представления функций формулами, построенными с помощью конъюнкции, дизъюнкции, отрицания, констант 0 и 1, т. е. формулами над $F = \{x \ \& \ y, x \ \vee \ y, \neg, 0, 1\}$. Такие формулы называются *булевыми*. Иногда в F включают импликацию.

Примем соглашение об опускании скобок в соответствии со следующим приоритетом операций: $\neg$, $\&$, $\vee$, $\rightarrow$. Укажем некоторые свойства операций $\neg$, $\&$, $\vee$. Эти операции (как и их свойства) называются булевыми.

Пусть A, B, C есть произвольные формулы над F . Тогда справедливы следующие свойства булевых операций.

1. Идемпоентность

$$A \& A = A \quad A \vee A = A$$

2. Коммутативность

$$A \& B = B \& A \quad A \vee B = B \vee A$$

3. Ассоциативность

$$A \& (B \& C) = (A \& B) \& C \quad A \vee (B \vee C) = (A \vee B) \vee C$$

4. Правила поглощения

$$A \& (A \vee B) = A \quad A \vee (A \& B) = A$$

5. Дистрибутивность

$$A \& (B \vee C) = (A \& B) \vee (A \& C) \quad A \vee (B \& C) = (A \vee B) \& (A \vee C)$$

6. Инволюция

$$\overline{\overline{A}} = A$$

7. Свойства констант

нейтральный элемент	поглощающий элемент
$A \& 1 = A; A \vee 0 = A$	$A \& 0 = 0 \quad A \vee 1 = 1$

8. Правила Аристотеля

закон исключенного третьего	закон противоречия
$A \vee \overline{A} = 1$	$A \& \overline{A} = 0$

9. Правила де Моргана

$$\overline{A \& B} = \overline{A} \vee \overline{B} \quad \overline{A \vee B} = \overline{A} \& \overline{B}$$

10. Связь импликации, дизъюнкции и конъюнкции

$$A \rightarrow B = \overline{A} \vee B \quad A \rightarrow B = \overline{A \& B}$$

11. Правила Порецкого

$$A \& (A \vee B) = A \& B \quad A \vee (\overline{A} \& B) = A \vee B$$

12. Правила склеивания

$$(A \vee B) \& (A \vee \overline{B}) = A \quad (A \& B) \vee (A \& \overline{B}) = A$$

Все эти равенства устанавливаются непосредственной проверкой.

Правило подстановки (замены равным). Если

- 1) A, C есть формулы;
- 2) B есть подформула формулы A , т. е. A есть $A(B)$;
- 3) $B = C$, то $A(B) = A(C)$. Коротко правило подстановки записывают так:

$$\frac{A(B), B = C}{A(B) = A(C)}.$$

Примем без доказательства следующее утверждение.

Теорема. Если A и B есть булевы формулы и $A = B$, то с помощью булевых равенств 1–12 и правила подстановки от формулы A можно перейти к формуле B за конечное число шагов.



Эта теорема широко используется при упрощении формул.

Пример. $\overline{x \& y \vee xy} = \overline{x \vee \overline{y} \vee xy} = x \vee \overline{y} \vee xy = x \vee xy \vee \overline{y} = x \vee \overline{y}.$

Замечание. Пусть M есть некоторое множество и $P(M)$ есть множество всех подмножеств множества M . Если A, B, C есть произвольные подмножества из M и $\neg A$ интерпретируется как $M - A$ (т. е. A есть дополнение A до M), $A \& B$ как $A \cap B$, $A \vee B$ как $A \cup B$, 0 как пустое множество $\emptyset$, а 1 есть все множество M , то при таком теоретико-множественном понимании операций $\neg, \&, \vee$ булевы свойства 1–12 останутся справедливыми.

Множество M , в котором определены операции $\neg, \&, \vee$ и константы 0 и 1 , удовлетворяющие аксиомам 1–12, называется *булевой алгеброй*. Обозначим булеву алгебру через $(M, \&, \vee, \neg, 0, 1)$. Тогда системы $(\{0, 1\}, \&, \vee, \neg, 0, 1)$ и $(P(M), \cap, \cup, \neg, \emptyset, M)$ являются булевыми алгебрами.

Множество M , в котором определены две операции $\&$ и $\vee$, удовлетворяющие аксиомам 1–4, называется *решеткой*. Решетка *дистрибутивна*, если дополнительно выполняется аксиома 5 дистрибутивности.

Пусть множество $M' = \{0, 1, 2, 01, 02, 12, 012\}$. Элемент 012 понимаем как $0 \& 1 \& 2$. Аналогично другие элементы из M' . Множество M состоит из множества M' и всех дизъюнкций попарно различных элементов множества M' . При этом ни одно дизъюнктивное слагаемое, рассматриваемое как множество своих сомножителей, не содержится в другом его дизъюнктивном слагаемом. Например, элементом множества M является дизъюнкция $01 \vee 12 \vee 02$. Множество M с операциями $\&$ и $\vee$, удовлетворяющими аксиомам 1–5, образует (свободную) дистрибутивную решетку с образующими $0, 1, 2$. Приведем пример преобразований в такой решетке. Решеточное выражение

$$(0 \vee 2)(01 \vee 12 \vee 012) = 001 \vee 012 \vee 0012 \vee 201 \vee 212 \vee 2012 = \\ 01 \vee 012 \vee 012 \vee 012 \vee 12 \vee 012 = 01 \vee 12.$$

1.4. Нормальные формы. Совершенные нормальные формы

Элементарная конъюнкция есть конъюнкция, составленная из попарно различных переменных или отрицаний переменных.

Пример. $x, y, xy, x_1x_2x_3.$

Дизъюнктивная нормальная форма (ДНФ) есть дизъюнкция, составленная из попарно различных элементарных конъюнкций.

Пример. $xy, xy \vee x, x_1x_2x_3 \vee x_1x_2x_3.$

Элементарная дизъюнкция есть дизъюнкция, составленная из попарно различных переменных или отрицаний переменных.

Пример. $x, x \vee \overline{y}, \overline{x} \vee y \vee \overline{z}.$

Конъюнктивная нормальная форма (КНФ) есть конъюнкция, составленная из попарно различных элементарных дизъюнкций.

Пример. $x, x \vee \bar{y}, (x \vee y)(x \vee \bar{y} \vee z)(x \vee y \vee \bar{z})$.

Введем следующие обозначения:

$$x^c = \begin{cases} x, & \text{если } c = 1; \\ \bar{x}, & \text{если } c = 0; \end{cases} \quad \bigwedge_{i=1}^n A_i = A_1 \& A_2 \& \dots \& A_n; \quad \bigvee_{i=1}^n A_i = A_1 \vee A_2 \vee \dots \vee A_n.$$

Знак $\leftrightarrow$ означает «тогда и только тогда».

Заметим, что $x^c = 1 \leftrightarrow x = c$;

$$x_1^{c_1} \& x_2^{c_2} \& \dots \& x_n^{c_n} = 1 \leftrightarrow x_1 = c_1, x_2 = c_2, \dots, x_n = c_n.$$

Лемма Шеннона (о разложении функции по компонентам). Всякая функция алгебры логики допускает представление:

$$f(x_1, \dots, x_k, x_{k+1}, \dots, x_n) = \bigvee_{(c_1, \dots, c_k) \in E_2^k} f(c_1, \dots, c_k, x_{k+1}, \dots, x_n) x_1^{c_1} \dots x_k^{c_k}, \quad (1.1)$$

где дизъюнкция берется по всем наборам $(c_1, \dots, c_k)$ длины k из 0 и 1.

Доказательство. Пусть левая часть (1.1) равна 1. Тогда последовательно получаем следующее:

$$\begin{aligned} f(x_1, \dots, x_k, x_{k+1}, \dots, x_n) &= 1 \leftrightarrow \\ f(c_1, \dots, c_k, x_{k+1}, \dots, x_n) &= 1, c_1 = x_1, \dots, c_k = x_k \leftrightarrow \\ f(c_1, \dots, c_k, x_{k+1}, \dots, x_n) &= 1, x_1^{c_1} = 1, \dots, x_k^{c_k} = 1 \leftrightarrow \\ f(c_1, \dots, c_k, x_{k+1}, \dots, x_n) &= 1, x_1^{c_1} \& \dots \& x_k^{c_k} = 1 \leftrightarrow \\ f(c_1, \dots, c_k, x_{k+1}, \dots, x_n) &\& x_1^{c_1} \& \dots \& x_k^{c_k} = 1 \leftrightarrow \\ \bigvee_{(c_1, \dots, c_k) \in E_2^k} f(c_1, \dots, c_k, x_{k+1}, \dots, x_n) x_1^{c_1} &\dots x_k^{c_k} = 1. \end{aligned}$$

Равенство (1.1) доказано. Лемма установлена.

Замечание. Равенство (1.1) называют также дизъюнктивным представлением Шеннона функции f . Функция $f(c_1, \dots, c_k, x_{k+1}, \dots, x_n)$ называется *компонентой* функции $f(x_1, \dots, x_n)$. Иногда $f(c_1, \dots, c_k, x_{k+1}, \dots, x_n)$ называют остаточной функцией. Константы $c_1, \dots, c_k$ могут стоять на любых аргументных местах функции f .

Совершенные нормальные формы

Теорема (О СДНФ). Всякая не равная тождественному нулю функция $f(x_1, \dots, x_n)$ допускает представление

$$f(x_1, \dots, x_n) = \bigvee_{f(c_1, \dots, c_n)=1} x_1^{c_1} \dots x_n^{c_n}, \quad (1.2)$$

где дизъюнкция берется по всем наборам $\mathbf{c} = (c_1, \dots, c_n)$ из 0 и 1, для которых $f(\mathbf{c}) = 1$.

Доказательство. Пусть $f(x_1, \dots, x_n) \neq 0$. Согласно лемме о разложении функции по компонентам при $k = n$ получаем

$$f(x_1, \dots, x_n) = \bigvee_{(c_1, \dots, c_n) \in E_2^n} f(c_1, \dots, c_n) x_1^{c_1} \dots x_n^{c_n}.$$

Из правой части равенства удалим все нулевые дизъюнктивные члены, в которых $f(c_1, \dots, c_n) = 0$. Тогда получим

$$f(x_1, \dots, x_n) = \bigvee_{f(c_1, \dots, c_n)=1} f(c_1, \dots, c_n) x_1^{c_1} \dots x_n^{c_n}.$$

Так как в этой формуле в любом дизъюнктивном члене элемент $f(c_1, \dots, c_n) = 1$, то она принимает вид

$$f(x_1, \dots, x_n) = \bigvee_{f(c_1, \dots, c_n)=1} x_1^{c_1} \dots x_n^{c_n}.$$

Теорема доказана.

Определение. Правая часть представления (1.2) называется *совершенной дизъюнктивной нормальной формой* (СДНФ) функции f .

Каждое слагаемое в СДНФ называется *конъюнктом*. Конъюнкт $x_1^{c_1} \dots x_n^{c_n} = 1$ на единственном наборе $x_1 = c_1, \dots, x_n = c_n$.

Пример. СДНФ для функции f , приведенной в табл. 1.5, имеет вид:

$$f(x, y, z) = \overline{x} \overline{y} \overline{z} \vee \overline{x} y \overline{z} \vee x \overline{y} \overline{z} \vee x y \overline{z}.$$

Дизъюнктивные слагаемые $\overline{x} \overline{y} \overline{z}$, $\overline{x} y \overline{z}$, $x \overline{y} \overline{z}$, $x y \overline{z}$ являются конъюнктами.

Таблица 1.5

x	0	0	0	0	1	1	1	1
y	0	0	1	1	0	0	1	1
z	0	1	0	1	0	1	0	1
f	1	0	0	1	0	1	0	1



Замечание. Всякую функцию алгебры логики можно реализовать формулой, построенной с помощью конъюнкции, дизъюнкции и отрицания. Поэтому множество функций $F = \{x \& y, x \vee y, x\}$ составляет полную систему. Так как $x \vee y = \overline{x \& y}$, то система функций $F = \{x \& y, \overline{x}\}$ полна. Так как $x \& y = \overline{x \vee y}$, то система $F = \{x \vee y, x\}$ полна. Система $F = \{x | y\}$, состоящая из единственной функции – штриха Шеффера, полна, ибо $x = x | x$, а $x \& y = (x | y) | (x | y)$. Система $F = \{x \uparrow y\}$, состоящая только из стрелки Пирса, полна, так как $x = x \uparrow x$, а $x \vee y = (x \uparrow y) \uparrow (x \uparrow y)$. Очевидно, что $x \uparrow y = \overline{x \vee y}$.

Теорема (о единственности СДНФ). Для всякой функции, не равной тождественному нулю, существует единственная СДНФ.

Доказательство. Существование СДНФ для функции $f \neq 0$ вытекает из предыдущей теоремы. Покажем, что эта СДНФ единственная. В самом деле, имеется $2^n - 1$ n -местных функций, не равных нулю тождественно. Подсчитаем число различных СДНФ от n переменных. Пусть C_n^k означает число сочетаний из n элементов по k . Тогда число одночленных СДНФ $x_1^{c_1} x_2^{c_2} \dots x_n^{c_n}$ равно C_n^1 . Число

k -членных СДНФ равно $C_{2^n}^k$. Число n -членных СДНФ равно $C_{2^n}^n$. Число всех различных СДНФ:

$$C_{2^n}^1 + C_{2^n}^2 + \dots + C_{2^n}^k + \dots + C_{2^n}^n = 2^{2^n} - 1.$$

Итак, $2^{2^n} - 1$ функций реализуются посредством $2^{2^n} - 1$ СДНФ, т. е. каждая функция реализуется единственной СДНФ.

Теорема (о СКНФ). Всякая не равная тождественной единице функция $f(x_1, \dots, x_n)$ допускает представление

$$f(x_1, \dots, x_n) = \bigwedge_{f(c_1, \dots, c_n)=0} (x_1^{c_1} \vee \dots \vee x_n^{c_n}), \quad (1.3)$$

где конъюнкция берется по всем наборам $c = (c_1, \dots, c_n)$ из 0 и 1, для которых $f(c) = 0$.

Доказательство. Заметим, что $\overline{x^c} = x^{\bar{c}}$. Пусть функция $f(x_1, \dots, x_n) \not\equiv 1$, тогда $\bar{f} \not\equiv 0$, и потому функция $\bar{f}$ допускает представление в виде СДНФ

$$\bar{f}(x_1, \dots, x_n) = \bigvee_{\bar{f}(c)=1} x_1^{c_1} \dots x_n^{c_n}.$$

Отсюда

$$f(x_1, \dots, x_n) = \overline{\bigvee_{f(c)=0} x_1^{c_1} \dots x_n^{c_n}} = \bigwedge_{f(c)=0} \overline{(x_1^{c_1} \vee \dots \vee x_n^{c_n})} = \bigwedge_{f(c)=0} (\overline{x_1^{c_1}} \wedge \dots \wedge \overline{x_n^{c_n}}) = \bigwedge_{f(c)=0} (x_1^{\bar{c}_1} \wedge \dots \wedge x_n^{\bar{c}_n}).$$

Теорема доказана.

Правая часть в представлении (1.3) называется *совершенной конъюнктивной нормальной формой* (СКНФ) функции f .

Пример. СКНФ для функции, приведенной в табл. 1.6, имеет вид:

$$f(x, y, z) = (x \vee y \vee z)(x \vee \bar{y} \vee \bar{z})(\bar{x} \vee y \vee \bar{z})(\bar{x} \vee \bar{y} \vee \bar{z}).$$

Таблица 1.6

x	0	0	0	1	1	1	1
y	0	0	1	1	0	0	1
z	0	1	0	1	0	1	1
f	0	1	1	0	1	0	0

1.5. Минимизация нормальных форм

Минимальной ДНФ (МДНФ) функции $f(x_1, \dots, x_n)$ называется ДНФ, реализующая функцию f и содержащая минимальное число символов переменных по сравнению со всеми другими ДНФ, реализующими функцию f .

Если для всякого набора $a = (a_1, \dots, a_n)$ значений переменных условие $g(a) = 1$ влечет $f(a) = 1$, то функция g называется частью функции f (или функция f на-

крывает функцию g). Если при этом для некоторого набора $c = (c_1, \dots, c_n)$ функция $g(c) = 1$, то говорят, что функция g **накрывает** единицу функции f на наборе c (или что g накрывает конъюнкт $x_1^{c_1} \dots x_n^{c_n}$ функции f). Заметим, что конъюнкт функции f есть часть функции f , накрывающая только одну единицу функции f .

Элементарная конъюнкция K называется **импликантом** функции f , если для всякого набора $a = (a_1, \dots, a_n)$ из 0 и 1 условие $K(a) = 1$ влечет $f(a) = 1$.

Импликант K функции f называется **простым**, если выражение, получающееся из него удалением любых множителей, уже не есть импликант функции f .

Всякий импликант функции f есть часть функции f .

Теорема. Всякая функция реализуется дизъюнкцией всех своих простых импликант (ПИ).

Доказательство. Пусть $f(x_1, \dots, x_n)$ есть функция, а $A = K_1 \vee \dots \vee K_m$ есть дизъюнкция всех ее простых импликант. Пусть $a_1, \dots, a_n$ есть произвольный набор длины n из 0 и 1.

Если $A(a) = 1$, то найдется дизъюнктивное слагаемое $K_i(a) = 1$, что влечет $f(a) = 1$, ибо K_i есть импликант функции f .

Если $f(a) = 1$, то в СДНФ для функции f найдется элементарная конъюнкция K , равная на этом наборе единице. Один из простых импликантов K_j функции f получается удалением некоторых множителей из K , и потому $K_j(a) = 1$, а тогда $A(a) = 1$.

Следовательно, $f = A$. Теорема доказана.

Определение. Сокращенная ДНФ функции f есть дизъюнкция всех простых импликант функции f .

Замечание. Всякая функция f реализуется своей сокращенной ДНФ. Для всякой функции, не равной тождественно нулю, существует единственная сокращенная ДНФ. □

Пусть A и B есть произвольные формулы. Из свойств булевых операций вытекают следующие обратимые правила преобразования ДНФ:

- 1) $\frac{A \cdot B \vee A \cdot \bar{B}}{A}$, полное склеивание (развертывание);
- 2) $\frac{A \cdot B \vee A \cdot \bar{B}}{A \vee A \cdot B \vee A \cdot \bar{B}}$, неполное склеивание;
- 3) $\frac{A \vee A \cdot B}{A}$, поглощение;
- 4) $\frac{A \vee A}{A}$; $\frac{A \& A}{A}$, идемпотентность (удаление дублирующих членов).

Замечание. Предыдущие обратимые правила можно написать в виде следующих равенств:

- 1) $A \cdot B \vee A \cdot B = A$;
- 2) $A \cdot B \vee A \cdot B = A \vee A \cdot B \vee A \cdot B$;
- 3) $A \vee A \cdot B = A$;
- 4) $A \vee A = A$, $A \& A = A$.

Теорема (Куайна). Если в СДНФ функции f провести все операции неполного склеивания, а затем все операции поглощения и удаления дублирующих членов, то в результате получится сокращенная ДНФ функции f .

Доказательство. Пусть имеем сокращенную ДНФ функции f . Проведем все операции развертывания к каждому простому импликанту для получения недостающих переменных в каждом дизъюнктивном слагаемом сокращенной ДНФ. В полученном выражении из нескольких одинаковых дизъюнктивных слагаемых оставим только по одному экземпляру. В результате получим СДНФ функции f . Теперь, исходя из полученной СДНФ, в обратном порядке проведем операции добавления одинаковых дизъюнктивных слагаемых (с помощью правил идемпотентности), неполного склеивания и поглощения. В итоге получим исходную сокращенную ДНФ.

1.5.1. Алгоритм Куайна построения сокращенной ДНФ

1. Получить СДНФ функции f .
2. Провести все операции неполного склеивания.
3. Провести все операции поглощения.

Пример. Построим сокращенную ДНФ для функции, приведенной в табл. 1.7.

Таблица 1.7

x	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
y	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
z	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
t	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
f	1	1	1	1	0	1	0	0	1	0	1	0	1	1	1	1

1. Строим СДНФ функции f . $f(x,y,z,t) = \overline{x} \overline{y} z t \vee \overline{x} y z t \vee \overline{x} y \overline{z} t \vee \overline{x} y z \overline{t} \vee \overline{x} y z t \vee \overline{x} y z t \vee \overline{x} y z t \vee \overline{x} y z t \vee \overline{x} y z t \vee \overline{x} y z t \vee \overline{x} y z t$. Занумеруем дизъюнктивные члены в полученной СДНФ в порядке от 1 до 11.

2. Проводим все операции неполного склеивания.

Первый этап склеиваний:

Слагаемые	Склеивание по	Результат
1,2	t	$\overline{x} y z$
1,3	z	$\overline{x} y t$
1,6	x	$y z t$
2,4	z	$\overline{x} y t$
2,5	y	$\overline{x} z t$
3,4	t	$\overline{x} y z$
3,7	x	$y z t$
5,9	x	$y z t$
6,7	z	$\overline{x} y t$

Слагаемые	Склеивание по	Результат
6,8	y	$xz\bar{t}$
7,10	y	xzt
8,9	t	$xy\bar{z}$
8,10	z	xyt
9,11	z	xyt
10,11	t	xyz

После первого этапа склеиваний (и возможных поглощений) получаем, что

$$f(x,y,z,t) = \bar{x}\bar{y}\bar{z} \vee \bar{x}y\bar{t} \vee \bar{y}\bar{z}t \vee \bar{x}y\bar{t} \vee \bar{x}\bar{z}t \vee \bar{x}y\bar{z} \vee \bar{y}\bar{z}t \vee \bar{y}zt \vee \bar{x}y\bar{t} \vee \bar{x}\bar{z}t \vee \bar{x}zt \vee \bar{x}yz \vee x\bar{y}\bar{z} \vee xyt \vee xyt \vee xyz.$$

Пронумеруем дизъюнктивные члены в полученной ДНФ в порядке их следования от 1 до 15.

Второй этап склеиваний.

Слагаемые	Склеивание по	Результат
1,6	z	$\bar{x}y$
2,4	t	$\bar{x}y$
2,8	x	$y\bar{t}$
3,7	z	$\bar{x}y$
8,13	y	$x\bar{t}$
10,11	z	$x\bar{t}$
12,15	z	xy
13,14	t	xy

После второго этапа склеиваний и последующих поглощений получаем, что $f(x,y,z,t) = xy \vee \bar{x}y \vee y\bar{t} \vee x\bar{t} \vee \bar{x}zt \vee yzt$. Это и будет сокращенной ДНФ для функции f , ибо дальнейшие склеивания невозможны.

1.5.2. Алгоритм построения сокращенной ДНФ с помощью КНФ

Пусть $f(x_1, \dots, x_n)$ есть некоторая функция алгебры логики. Построим для f некоторую КНФ. Осуществим далее следующие преобразования.

1. В КНФ раскроем скобки и удалим дублирующие члены согласно равенствам $K \cdot K = K$, $K \vee K = K$; удалим дизъюнктивные слагаемые, содержащие одновременно переменную и ее отрицание. В результате получим дизъюнкцию конъюнкций, каждая из которых содержит только по одному элементу из каждой скобки КНФ.

2. В полученном выражении удалим нулевые дизъюнктивные слагаемые.

3. В полученном выражении проведем все поглощения (согласно равенству $A \vee AB = A$), а затем удалим дублирующие члены.

В результате проведенных операций получим сокращенную ДНФ функции f . Покажем это.

Для каждой элементарной дизъюнкции D в КНФ и каждой элементарной конъюнкции K в сокращенной ДНФ (сокр. ДНФ) существует некоторый множитель вида x^a из K , содержащийся в D , т. е.

$$\forall D \in \text{ДНФ} \forall K \in \text{сокр. ДНФ} \exists x^a \in K (x^a \in D).$$

Допустим противное: в КНФ существует элементарная конъюнкция D , в сокращенной ДНФ существует элементарная конъюнкция K , для которой всякий множитель вида x^a из K не входит в D . Не уменьшая общности, возьмем для простоты $K = x_1^{a_1} x_2^{a_2} \dots x_k^{a_k}$, $D = x_{k+1}^{a_{k+1}} \vee \dots \vee x_r^{a_r}$.

Положим $x_1 = a_1, \dots, x_k = a_k, x_{k+1} = c_{k+1} \neq a_{k+1}, \dots, x_r = c_r \neq a_r$. Тогда $K(a_1, \dots, a_k) = 1$, и потому $f(a_1, \dots, a_k, c_{k+1}, \dots, c_r) = 1$. С другой стороны, $D(c_{k+1}, \dots, c_r) = 0$, и потому $f(a_1, \dots, a_k, c_{k+1}, \dots, c_r) = 0$. Противоречие.

Пусть по-прежнему для простоты произвольный простой импликант K из сокращенной ДНФ равен $x_1^{a_1} \& x_2^{a_2} \& \dots \& x_k^{a_k}$. Тогда элементы $x_1^{a_1}, x_2^{a_2}, \dots, x_k^{a_k}$ попадут в не менее чем k скобок в КНФ. Если допустим, что этого нет, то при перемножении скобок из КНФ не получим дизъюнктивного слагаемого, который содержал бы множители $x_1^{a_1}, x_2^{a_2}, \dots, x_k^{a_k}$, а потому, строя из результата перемножения сокращенную ДНФ вычеркиванием лишних сомножителей, не получим простого импликанта K .

Так как $x_1^{a_1}, x_2^{a_2}, \dots, x_k^{a_k}$ содержатся в k разных скобках КНФ, а всякая другая скобка, отличная от указанных k скобок, содержит хотя бы один элемент вида x из K , то при раскрытии скобок имеем простой импликант K . После проведения всех операций поглощения и удаления дублирующих множителей останутся только простые импликанты из сокращенной ДНФ, ибо если предположить наличие в результате хотя бы одного дизъюнктивного слагаемого, отличного от всех простых импликантов сокращенной ДНФ, то можно подобрать такие значения переменных функции f , на которых все простые импликанты примут значение 0, а это дополнительное слагаемое значение 1, чего быть не может.

Пример. Построим сокращенную ДНФ этим способом для функции $f = 1111010010101111$ из предыдущего примера.

$$\begin{aligned} f(x, y, z, t) &= (x \vee \bar{y} \vee \bar{z} \vee t)(x \vee \bar{y} \vee \bar{z} \vee t)(x \vee \bar{y} \vee \bar{z} \vee t)(\bar{x} \vee y \vee z \vee \bar{t})(\bar{x} \vee y \vee \bar{z} \vee \bar{t}) = \\ &= (x \vee \bar{y} \vee t)(x \vee \bar{y} \vee \bar{z} \vee t)(\bar{x} \vee y \vee t) = (x \vee \bar{y} \vee \bar{y} \bar{z} \vee t)(\bar{x} \vee y \vee t) = \\ &= xy \vee xt \vee \bar{x}y \vee \bar{y}t \vee xt \vee xzt \vee yzt. \end{aligned}$$

Сокращенная ДНФ для функции

$$f(x, y, z, t) = xy \vee xt \vee \bar{x}y \vee \bar{y}t \vee xt \vee xzt \vee yzt,$$

что, естественно, совпадает с результатом предыдущего примера.

Тупиковой ДНФ (ТДНФ) функции f называется такая ДНФ ее простых импликант, из которой нельзя удалить ни одного импликанта, не изменив функции f .

Теорема. Всякая минимальная ДНФ некоторой функции является ее тупиковой ДНФ.

Доказательство. В МДНФ входят только простые импликанты, иначе некоторые множители в непростом импликанте можно удалить в противоречие с минимальностью исходной ДНФ. В МДНФ нет лишних импликант, иначе исходная ДНФ не является минимальной.

Вывод. Для получения МДНФ функции f нужно построить все ТДНФ функции f и выбрать те из них, которые содержат минимальное число букв.

1.5.3. Построение всех тупиковых ДНФ

Пусть $f(x_1, \dots, x_n)$ есть функция алгебры логики.

1. Построим СДНФ функции f , и пусть $P_1, P_2, \dots, P_s$ есть ее конъюнкты.

2. Построим сокращенную ДНФ функции f , и пусть $K_1, K_2, \dots, K_m$ есть все ее простые импликанты.

3. Построим матрицу покрытий простых импликант функции f ее конъюнктами (табл. 1.8), полагая, что

$$a_{ij} = \begin{cases} 1, & \text{если каждый множитель в } K_i \text{ есть} \\ & \text{множитель в } P_j; (P_j \text{ есть часть в } K_i); \\ 0 & \text{в противном случае.} \end{cases}$$

Таблица 1.8

N	P_1	P_2	...	P_j	...	P_s
K_1	a_{11}	a_{12}	...	a_{1j}	...	a_{1s}
K_2	a_{21}	a_{22}	...	a_{2j}	...	a_{2s}
			...		...	
K_i	a_{i1}	a_{i2}	...	a_{ij}	...	a_{is}
			...		...	
K_m	a_{m1}	a_{m2}	...	a_{mj}	...	a_{ms}

4. Для каждого столбца j ($1 \vee j \vee s$) найдем множество E_j всех тех номеров i строк, для которых $a_{ij} = 1$. Пусть $E_j = (e_{j_1}, e_{j_2}, \dots, e_{j_r})$. Составим выражение

$A = \bigwedge_{j=1}^s (e_{j_1} \vee e_{j_2} \vee \dots \vee e_{j_r})$. Назовем его *решеточным выражением*. Это выражение

можно рассматривать как формулу, построенную в свободной дистрибутивной решетке с образующими $1, 2, \dots, m$ и с операциями $\&$ и $\vee$.

5. В выражении A раскроем скобки, приведя выражение A к равносильному выражению $B = \bigvee_i (e_{i_1} \& e_{i_2} \& \dots \& e_{i_s})$, где перечислены все конъюнкции $e_{i_1} \& e_{i_2} \& \dots \& e_{i_s}$, элементы $e_{i_1}, e_{i_2}, \dots, e_{i_s}$ которой взяты из скобок $1, 2, \dots, s$ соответственно в выражении A .

6. В выражении B проведем все операции удаления дублирующих членов и все операции поглощения. В результате получим равносильное выражение C , представляющее собой дизъюнкцию элементарных конъюнкций.

Утверждение. Каждая элементарная конъюнкция $i_1 \& i_2 \& \dots \& i_r$ в S дает ТДНФ $K_{i_1} \vee K_{i_2} \vee \dots \vee K_{i_r}$ для f . Все ТДНФ для функции f исчерпываются элементарными конъюнкциями в выражении S .

Пример. Сокращенная ДНФ для функции $f = 1111010010101111$ имеет вид $f = xy \vee \overline{xy} \vee \overline{yt} \vee \overline{xt} \vee \overline{xzt} \vee \overline{yzt}$.

Для функции f построим все минимальные ДНФ.

1. Строим матрицу покрытий (табл. 1.9).

Таблица 1.9

№	ПИ	$\overline{x}$ $\overline{y}$ $\overline{z}$ $\overline{t}$	$\overline{x}$ $\overline{y}$ $\overline{z}$ $\overline{t}$	$\overline{x}$ $\overline{y}$ $\overline{z}$ $\overline{t}$	$\overline{x}$ $\overline{y}$ $\overline{z}$ $\overline{t}$	$\overline{x}$ $\overline{y}$ $\overline{z}$ $\overline{t}$	x $\overline{y}$ $\overline{z}$ $\overline{t}$	x $\overline{y}$ $\overline{z}$ $\overline{t}$	x $\overline{y}$ $\overline{z}$ $\overline{t}$	x $\overline{y}$ $\overline{z}$ $\overline{t}$	x $\overline{y}$ $\overline{z}$ $\overline{t}$	x $\overline{y}$ $\overline{z}$ $\overline{t}$
1	xy								+	+	+	+
2	$\overline{\overline{xy}}$	+	+	+	+							
3	$\overline{yt}$	+		+			+	+				
4	$\overline{xt}$						+	+	+		+	
5	$\overline{\overline{xzt}}$		+			+						
6	$\overline{yzt}$					+				+		

2. Строим решеточное выражение (по столбцам табл. 1.9).

$$E = (2 \vee 3)(2 \vee 5)(2 \vee 3)2(5 \vee 6)(3 \vee 4)(3 \vee 4)(1 \vee 6)(1 \vee 4)(1) = \\ = (2 \vee 3)(2 \vee 5)(5 \vee 6)(3 \vee 4)(1 \vee 4)(1 \vee 6)12 = \\ = (5 \vee 6)(3 \vee 4)(1)(2) = 1235 \vee 1245 \vee 1236 \vee 1246.$$

3. Строим все тупиковые ДНФ функции f :

- $f = xy \vee \overline{xy} \vee \overline{yt} \vee \overline{xzt}$, простые импликанты 1,2,3,5;
- $f = xy \vee \overline{xy} \vee \overline{xt} \vee \overline{xzt}$, простые импликанты 1,2,4,5;
- $f = xy \vee \overline{xy} \vee \overline{yt} \vee \overline{yzt}$, простые импликанты 1,2,3,6;
- $f = xy \vee \overline{xy} \vee \overline{yt} \vee \overline{xt} \vee \overline{yzt}$, простые импликанты 1,2,4,6.

4. Все найденные ТДНФ являются минимальными ДНФ.

1.5.4. Алгоритм минимизации функций в классе ДНФ

1. Строим СДНФ функции f .
 2. Строим сокращенную ДНФ функции f .
 3. С помощью матрицы покрытий и решеточного выражения строим все ТДНФ функции f .

4. Среди построенных ТДНФ выбираем все минимальные дизъюнктивные нормальные формы функции f .

1.5.5. Алгоритм минимизации функций в классе КНФ

Чтобы построить все минимальные КНФ (МКНФ) функции f , следует построить все МДНФ функции $\bar{f}$ и взять от каждой из них отрицание, для чего заменить знаки $\&$ на $\vee$, а $\vee$ – на $\&$ (сохранив первоначальное распределение скобок) и над каждой буквой поставить знак отрицания. Полученные КНФ для функции f будут минимальными. В самом деле, если бы для f существовала КНФ с меньшим числом букв, то ее отрицание дало бы для $\bar{f}$ ДНФ с меньшим числом букв, чем в любой из минимальных ДНФ для $\bar{f}$. Противоречие с их минимальностью.

1.5.6. Алгоритм минимизации функций в классе нормальных форм

Пусть f есть функция алгебры логики.

1. Строим все МДНФ функции f .
2. Строим все МКНФ функции f .
3. Из построенных минимальных форм выбираем простейшие (по числу букв).

Пример. В классе нормальных форм минимизировать функцию

$$f = 01011110.$$

1. Строим СДНФ для функции f .

$$f(x,y,z) = \bar{x}\bar{y}z \vee \bar{x}yz \vee x\bar{y}\bar{z} \vee x\bar{y}z \vee xy\bar{z}.$$

2. Строим сокращенную ДНФ функции f .

$$\begin{aligned} f(x,y,z) &= (x \vee y \vee z)(x \vee \bar{y} \vee z)(\bar{x} \vee \bar{y} \vee \bar{z}) = \\ &= (x \vee x\bar{y} \vee xz \vee xy \vee y\bar{y} \vee yz \vee xz \vee yz \vee zz)(\bar{x} \vee \bar{y} \vee \bar{z}) = (x \vee z)(\bar{x} \vee \bar{y} \vee \bar{z}) = \\ &= x\bar{x} \vee x\bar{y} \vee x\bar{z} \vee \bar{x}\bar{z} \vee \bar{y}\bar{z} \vee \bar{z}\bar{z} = \bar{x}\bar{z} \vee \bar{y}\bar{z} \vee x\bar{y}. \end{aligned}$$

3. Строим матрицу покрытий (табл. 1.10).

Таблица 1.10

№	ПИ	$\bar{x}\bar{y}z$	$\bar{x}yz$	$x\bar{y}\bar{z}$	$x\bar{y}z$	$xy\bar{z}$
1	$\bar{x}z$	+	+			
2	$\bar{y}z$	+			+	
3	$x\bar{y}$			+	+	
4	$x\bar{z}$			+		+

Решеточное выражение $E = (1 \vee 2)1(3 \vee 4)(2 \vee 3)4 = 134 \vee 124$.

4. Строим все тупиковые ДНФ функции f .

$$f(x,y,z) = \bar{x}\bar{z} \vee x\bar{y} \vee x\bar{z}; \quad f(x,y,z) = \bar{x}\bar{z} \vee \bar{y}z \vee x\bar{z}.$$

5. Обе построенные ТДНФ являются минимальными.

6. Повторяем эти этапы для функции $\bar{f}$.

$$\text{СДНФ. } \bar{f}(x,y,z) = \bar{x}\bar{y}\bar{z} \vee \bar{x}y\bar{z} \vee x\bar{y}z.$$

Сокращенная ДНФ.

$$\begin{aligned}\bar{f}(x,y,z) &= (x \vee y \vee \bar{z})(x \vee \bar{y} \vee \bar{z})(\bar{x} \vee y \vee z)(\bar{x} \vee y \vee \bar{z})(\bar{x} \vee \bar{y} \vee z) = \\ &= (x \vee z)(\bar{x} \vee y)(\bar{x} \vee y \vee z) = (x \vee z)(y \vee yz) = xyz \vee xz.\end{aligned}$$

Строим матрицу покрытий (табл. 1.11).

Таблица 1.11

№	ПИ	$\bar{x}\bar{y}\bar{z}$	$\bar{x}y\bar{z}$	$x\bar{y}z$
1	$\bar{x}z$	+	+	
2	xyz			+



Решеточный многочлен $E = 112 = 12$. Единственная тупиковая ДНФ (она же минимальная) для функции $\bar{f}(x,y,z) = \bar{x}z \vee xyz$.

Минимальная КНФ функции $f(x,y,z) = (x \vee z)(\bar{x} \vee \bar{y} \vee \bar{z})$. Из построенных МДНФ и МКНФ выбираем простейшую: $f(x,y,z) = (x \vee z)(\bar{x} \vee \bar{y} \vee \bar{z})$.

Пример. В классе нормальных форм минимизировать функцию $f = 11011011$.

1. СДНФ. $f(x,y,z) = \bar{x}yz \vee x\bar{y}z \vee x\bar{y}\bar{z} \vee x\bar{y}z \vee x\bar{y}z \vee x\bar{y}z$.

2. Сокращенная ДНФ. $f(x,y,z) = (x \vee \bar{y} \vee z)(\bar{x} \vee y \vee \bar{z}) = xy \vee x\bar{z} \vee \bar{y}\bar{z} \vee \bar{x}z \vee yz \vee \bar{x}y$.

3. Строим матрицу покрытий (табл. 1.12).

Таблица 1.12

№	ПИ	$\bar{x}yz$	$x\bar{y}z$	$x\bar{y}\bar{z}$	$x\bar{y}z \vee x\bar{y}\bar{z}$	xyz
1	xy				+	+
2	$\bar{x}z$				+	+
3	$\bar{y}z$	+			+	
4	$\bar{x}z$		+	+		
5	yz			+		+
6	$\bar{x}y$	+	+			

Решеточный многочлен. $E = (3 \vee 6)(4 \vee 6)(4 \vee 5)(2 \vee 3)(1 \vee 2)(1 \vee 5) = 1246 \vee 1356 \vee 134 \vee 256 \vee 2345$.

4. Все тупиковые ДНФ функции f .

$$f(x,y,z) = xy \vee x\bar{z} \vee \bar{x}z \vee \bar{x}y; \quad f(x,y,z) = xy \vee \bar{y}z \vee yz \vee \bar{x}y;$$

$$f(x,y,z) = xy \vee \bar{y}z \vee xz; \quad f(x,y,z) = xz \vee yz \vee \bar{x}y; \quad f(x,y,z) = xz \vee \bar{y}z \vee \bar{x}z \vee yz.$$

5. Все минимальные ДНФ функции f .

$$f(x,y,z) = xy \vee \bar{y}\bar{z} \vee \bar{x}z; f(x,y,z) = x\bar{z} \vee yz \vee \bar{x}\bar{y}.$$

6. Повторяем указанные выше этапы для функции $\bar{f}$.

$$\text{СДНФ. } \bar{f}(x,y,z) = \bar{x}yz \vee x\bar{y}z.$$

Сокращенная ДНФ.

$$\begin{aligned} \bar{f}(x,y,z) &= (x \vee y \vee z)(x \vee y \vee \bar{z})(x \vee \bar{y} \vee \bar{z})(\bar{x} \vee y \vee \bar{z})(\bar{x} \vee \bar{y} \vee z)(\bar{x} \vee \bar{y} \vee \bar{z}) = \\ &= (x \vee y)(x \vee \bar{y} \vee \bar{z})(\bar{x} \vee y \vee z)(\bar{x} \vee \bar{y}) = (x \vee yz)(\bar{x} \vee \bar{x}z) = x\bar{y}z \vee x\bar{y}\bar{z}. \end{aligned}$$

Построенная сокращенная ДНФ функции $\bar{f}$ является для нее тупиковой и минимальной.

Минимальная КНФ функции $f(x,y,z) = (x \vee \bar{y} \vee z)(\bar{x} \vee y \vee \bar{z})$.

Построенные МДНФ и МКНФ имеют одно и то же число букв; все они составляют минимальные формы для f .

$$f(x,y,z) = xy \vee \bar{y}\bar{z} \vee \bar{x}z; f(x,y,z) = x\bar{z} \vee yz \vee \bar{x}\bar{y};$$

$$f(x,y,z) = (x \vee \bar{y} \vee z)(\bar{x} \vee y \vee \bar{z}).$$

1.6. Минимизация частично определенных функций

Пусть функция $f(x_1, \dots, x_n)$ частично (не всюду) определена. Если f не определена на p наборах из 0 и 1, то существует 2^p возможностей для доопределения функции f . Полностью определенная функция $g(x_1, \dots, x_n)$ есть доопределение функции f , если g совпадает с f на тех наборах из 0 и 1, на которых f определена.

Задача минимизации частично определенной функции f сводится к отысканию такого доопределения g функции f , которое имеет простейшую (по числу букв) минимальную форму.

Обозначим через $f_0(x_1, \dots, x_n)$ и $f_1(x_1, \dots, x_n)$ доопределения нулями и единицами соответственно частично определенной функции $f(x_1, \dots, x_n)$.

Теорема. Минимальная ДНФ частично определенной функции $f(x_1, \dots, x_n)$ есть дизъюнкция самых коротких импликант в сокращенной ДНФ доопределения $f_1(x_1, \dots, x_n)$, которые в совокупности покрывают все конъюнкты доопределения $f_0(x_1, \dots, x_n)$.

Доказательство. Рассмотрим СДНФ некоторого доопределения $g(x_1, \dots, x_n)$ функции $f(x_1, \dots, x_n)$. Конъюнкты, входящие в эту форму, войдут и в СДНФ доопределения f_1 . Поэтому любой простой импликант функции g будет совпадать с некоторым импликантом функции f_1 или покрываться им. Самые короткие импликанты, покрывающие единицы функции f , есть импликанты функции f_1 . Доопределение f_0 имеет минимальное количество конъюнктов в своей СДНФ, следовательно, и количество простых импликант функции f_1 , необходимых для

накрытия этих конъюнктов, будет наименьшим. Оптимальная ДНФ, составленная из самых коротких простых импликант в сокращенной ДНФ функции f_1 , покрывающих все конъюнкты функции f_0 , будет самой короткой ДНФ, доопределяющей функцию f .

Так как единицы функции f_1 составлены из единиц функции f и единиц на наборах, на которых f не определена, то построенная ДНФ, покрывая все единицы функции f_0 (а следовательно, и все единицы функции f), совпадает с минимальной ДНФ некоторого доопределения g функции f .

1.6.1. Алгоритм минимизации частично определенных функций в классе ДНФ

1. Строим СДНФ функции f_1 .
2. Строим сокращенную ДНФ функции f_1 .
3. С помощью матрицы покрытий конъюнктов функции f_0 простыми импликантами функции f_1 и решеточного выражения строим все тупиковые ДНФ (для некоторых доопределений функции f).
4. Среди полученных ТДНФ выбираем простейшие; они являются минимальными ДНФ (для некоторых доопределений функции f).

1.6.2. Алгоритм минимизации частично определенных функций в классе КНФ

Построение минимальных КНФ для частично определенной функции аналогично построению минимальных КНФ для всюду определенной функции.

Алгоритм минимизации частично определенных функций в классе нормальных форм аналогичен алгоритму минимизации в классе нормальных форм для всюду определенных функций.

Пример. В классе нормальных форм минимизировать частично определенную функцию $f(x,y,z,t) = 1---010010-01--1$.

Решение. Минимизируем функцию f в классе ДНФ.

1. Строим СДНФ для доопределения нулями f_0 функции f .

$$f_0(x,y,z,t) = \overline{xyzt} \vee \overline{xyzt} \vee \overline{xyzt} \vee \overline{xyzt} \vee xyzt.$$

2. Строим сокращенную ДНФ для доопределения единицами f_1 функции f (табл. 1.13).

Таблица 1.13

№	$xyzt$	f	f_0	f_1	$\bar{f}$	h_0	h_1
0	0000	1	1	1	0	0	0
1	0001	–	0	1	–	0	1
2	0010	–	0	1	–	0	1
3	0011	–	0	1	–	0	1
4	0100	0	0	0	1	1	1

Окончание табл. 1.13

№	xyzt	f	f ₀	f ₁	f̄	h ₀	h ₁
5	0101	1	1	1	0	0	0
6	0110	0	0	0	1	1	1
7	0111	0	0	0	1	1	1
8	1000	1	1	1	0	0	0
9	1001	0	0	0	1	1	1
10	1010	–	0	1	–	0	1
11	1011	0	0	0	1	1	1
12	1100	1	1	1	0	0	0
13	1101	–	0	1	–	0	1
14	1110	–	0	1	–	0	1
15	1111	1	1	1	0	0	0

$$\begin{aligned}
 f_1(x,y,z,t) &= (x \vee \bar{y} \vee z \vee t)(x \vee \bar{y} \vee \bar{z} \vee t)(x \vee \bar{y} \vee \bar{z} \vee \bar{t})(\bar{x} \vee y \vee z \vee \bar{t})(\bar{x} \vee \bar{y} \vee \bar{z} \vee \bar{t}) = \\
 &= (x \vee \bar{y} \vee t)(xy \vee \bar{x}\bar{y} \vee \bar{z} \vee \bar{t})(\bar{x} \vee y \vee z \vee \bar{t}) = \\
 &= (x \vee \bar{y} \vee t)(xy \vee \bar{x}\bar{y} \vee \bar{x}\bar{z} \vee y\bar{z} \vee \bar{z}\bar{t} \vee \bar{x}t \vee y\bar{t} \vee \bar{z}t \vee \bar{t}) = \\
 &= (x \vee \bar{y} \vee t)(xy \vee \bar{x}\bar{y} \vee \bar{x}\bar{z} \vee y\bar{z} \vee \bar{t}) = \\
 &= xy \vee \bar{x}\bar{y} \vee \bar{x}\bar{z} \vee y\bar{z} \vee \bar{t} \vee xy\bar{t} \vee \bar{x}\bar{y}t \vee \bar{x}\bar{z}t \vee y\bar{z}t = \\
 &= xy \vee \bar{x}\bar{y} \vee \bar{x}\bar{z} \vee y\bar{z} \vee \bar{t}.
 \end{aligned}$$

3. Строим матрицу покрытий конъюнктов в СДНФ для доопределения нуля-ми f_0 функции f с помощью построенной сокращенной ДНФ для f_1 (табл. 1.14).

4. По табл. 13.14 строим решеточный многочлен

$$E = (2 \vee 4)(5 \vee 6)(3 \vee 4)(1 \vee 3)1 = 145 \vee 1235 \vee 146 \vee 1236.$$

Таблица 1.14

№	ПИ	$\bar{x}\bar{y}\bar{z}\bar{t}$	$\bar{x}\bar{y}z\bar{t}$	$\bar{x}y\bar{z}\bar{t}$	$xy\bar{z}\bar{t}$	xyzt
1	xy				+	+
2	$\bar{x}\bar{y}$	+				
3	$\bar{x}t$			+	+	
4	$\bar{y}t$	+		+		
5	$\bar{x}\bar{z}t$		+			
6	$\bar{y}zt$		+			

5. Строим все тупиковые ДНФ.

$$\begin{aligned}
 g_1 &= xy \vee \bar{y}\bar{t} \vee \bar{x}\bar{z}t; & g_2 &= xy \vee \bar{x}\bar{y} \vee \bar{x}t \vee \bar{x}\bar{z}t; \\
 g_3 &= xy \vee \bar{y}t \vee y\bar{z}t; & g_4 &= xy \vee \bar{x}\bar{y} \vee \bar{x}t \vee y\bar{z}t.
 \end{aligned}$$

6. Из построенных тупиковых ДНФ выбираем минимальные.

$$g_1 = xy \vee \bar{y}\bar{t} \vee \bar{x}\bar{z}t; \quad g_3 = xy \vee \bar{y}\bar{t} \vee yzt.$$

Функции g_1 и g_3 есть минимальные доопределения функции f в классе ДНФ. Минимизируем теперь функцию f в классе КНФ. Для этого проведем минимизацию функции $\bar{f}$ в классе ДНФ. Пусть h_0 и h_1 есть доопределения нулями и единицами соответственно функции $\bar{f}$.

1. Строим СДНФ для доопределения нулями h_0 функции $\bar{f}$.

$$h_0(x, y, z, t) = \bar{y}y\bar{z}t \vee \bar{x}y\bar{z}t \vee \bar{x}yzt \vee \bar{x}\bar{y}z\bar{t} \vee \bar{x}yzt.$$

2. Сокращенная ДНФ для

$$\begin{aligned} h_1 &= (x \vee y \vee z \vee t)(x \vee \bar{y} \vee z \vee \bar{t})(\bar{x} \vee y \vee z \vee t)(\bar{x} \vee \bar{y} \vee z \vee \bar{t})(\bar{x} \vee y \vee \bar{z} \vee \bar{t}) = \\ &= (x \vee z \vee \bar{y}\bar{t} \vee \bar{y}t)(\bar{x} \vee z \vee t)(\bar{x} \vee \bar{y} \vee \bar{z} \vee \bar{t}) = \\ &= (x \vee z \vee \bar{y}\bar{t} \vee \bar{y}t)(\bar{x} \vee \bar{y}z \vee z\bar{t} \vee \bar{y}t \vee \bar{z}t) = \\ &= \bar{y}t \vee \bar{x}yz \vee xz\bar{t} \vee xz\bar{t} \vee \bar{x}z \vee \bar{y}z \vee \bar{z}t \vee \bar{x}y\bar{t} \vee yz\bar{t} = \\ &= \bar{y}t \vee \bar{x}z \vee \bar{z}t \vee xz\bar{t} \vee \bar{x}y\bar{t} \vee yz. \end{aligned}$$

3. Матрица покрытий конъюнктов в СДНФ для h_0 с помощью простых импликант в сокращенной ДНФ для h_1 приведена в табл. 1.15.

Таблица 1.15

№	ПИ	$\bar{x}y\bar{z}t$	$\bar{x}yzt$	$\bar{x}y\bar{z}\bar{t}$	$\bar{x}yzt$	$\bar{x}y\bar{z}t$
1	$\bar{y}t$				+	+
2	$\bar{x}z$		+	+		
3	$\bar{z}t$		+			
4	$xz\bar{t}$				+	
5	$\bar{x}y\bar{t}$	+	+			
6	$\bar{y}z$					+

4. Решеточное выражение $E = 5(2 \vee 3 \vee 5)2(1 \vee 4)(1 \vee 6) = 25(1 \vee 46) = 125 \vee 2456$.

5. Строим две тупиковые ДНФ.

$$g_5 = \bar{y}t \vee \bar{x}z \vee \bar{x}y\bar{t}, \quad g_6 = \bar{x}z \vee xz\bar{t} \vee \bar{x}y\bar{t} \vee \bar{y}z.$$

Минимальная ДНФ $g_5 = \bar{y}t \vee \bar{x}z \vee \bar{x}y\bar{t}$.

6. Функция $\bar{g}_5 = (y \vee \bar{t})(x \vee \bar{z})(x \vee \bar{y} \vee t)$ есть минимальное доопределение функции f в классе КНФ.

Найденные МДНФ g_1, g_3 и МКНФ $\bar{g}_5$ являются минимальными доопределениями функции f в классе нормальных форм.

Техническая реализация минимальных форм для функции часто проще, а потому дешевле реализации ее СДНФ (СКНФ). Следовательно, этап минимизации при конструировании логических схем является одним из важнейших.



1.7. Двойственные функции. Принцип двойственности

Определение. Двойственной для функции $f(x_1, \dots, x_n)$ называется функция

$$f^*(x_1, \dots, x_n) = \overline{f(\bar{x}_1, \dots, \bar{x}_n)}.$$

Примеры.

1. $f = x \& y$; $f^* = \overline{\overline{x} \& \overline{y}} = x \vee y$.
2. $f = x \vee y$; $f^* = \overline{\overline{x} \vee \overline{y}} = x \& y$.
3. $f = x$; $f^* = \overline{\overline{x}} = x$.
4. $f = \bar{x}$; $f^* = \overline{\overline{\bar{x}}} = \bar{x}$.
5. $f(x_1, \dots, x_n) \equiv 0$; $f^* = \overline{f(\bar{x}_1, \dots, \bar{x}_n)} = \overline{0} = 1$.
6. $f(x_1, \dots, x_n) \equiv 1$; $f^* = \overline{f(\bar{x}_1, \dots, \bar{x}_n)} = \overline{1} = 0$.
7. $f = x \rightarrow y$; $f^* = \overline{\overline{x} \rightarrow \overline{y}} = x \vee y = y \rightarrow x$.

Заметим, что $(f^*((x_1, \dots, x_n))^* = (\overline{f(\bar{x}_1, \dots, \bar{x}_n)})^* = \overline{\overline{f(\bar{x}_1, \dots, \bar{x}_n)}} = f(x_1, \dots, x_n)$, т. е. $(f^*)^* = f$.

Теорема (о суперпозиции двойственных функций). Функция, двойственная суперпозиции функций, равна суперпозиции функций, двойственных к функциям, составляющим эту суперпозицию.

Доказательство.



$$\begin{aligned} (f(g_1, \dots, g_m))^* &= \overline{f(\overline{g_1}, \dots, \overline{g_m})} = \overline{f(\overline{g_1(x_{11}, \dots, x_{1,n_1})}, \dots, \overline{g_m(x_{m1}, \dots, x_{m,n_m})})} = \\ &= \overline{f(g_1(x_{11}, \dots, x_{1,n_1}), \dots, g_m(x_{m1}, \dots, x_{m,n_m}))} = \\ &= \overline{f((\overline{g_1(x_{11}, \dots, x_{1,n_1})})^*, \dots, (\overline{g_m(x_{m1}, \dots, x_{m,n_m})})^*)} = \\ &= \overline{f(\overline{g_1^*}, \dots, \overline{g_m^*})} = \overline{f(g_1^*, \dots, g_m^*)} = f^*(g_1^*, \dots, g_m^*). \end{aligned}$$

Теорема доказана.

Принцип двойственности

Если функция f задана формулой, построенной с помощью $\&$, $\vee$, $\neg$, 0 , 1 и переменных, то по теореме о суперпозиции двойственных функций и ввиду того, что для функций $x \& y$, $x \vee y$, x , $x \vee 0$, 1 двойственными являются функции $x \vee y$,



$x \& y, x, \bar{x}, 1, 0$ соответственно, f^* получается из f заменой $\&$ на $\vee$, $\vee$ на $\&$, 0 на 1 , 1 на 0 (при сохранении исходной расстановки скобок).

Пример. $\overline{(x \& y \vee z \& 1 \& (x \vee 0))}^* = \overline{x \vee y \& z \vee 0 \vee (x \& 1)}$.

Определение. Функция, совпадающая со своей двойственной, называется *самодвойственной*.

Если функция $f(x_1, \dots, x_n)$ самодвойственна, то функция $\bar{f}$ тоже самодвойственна, так как $(\bar{f}(x_1, \dots, x_n))^* = \bar{f}(\bar{x}_1, \dots, \bar{x}_n) = \bar{f}^*(x_1, \dots, x_n) = \bar{f}(x_1, \dots, x_n)$.

Теорема. Класс самодвойственных функций замкнут относительно суперпозиции.

Доказательство. Пусть функции $f, g_1, \dots, g_m$ самодвойственны. Тогда $f^* = f, g_1^* = g_1, \dots, g_m^* = g_m$. Суперпозиция $h = f(g_1, \dots, g_m)$ этих функций самодвойственна, ибо функция $h^* = (f(g_1, \dots, g_m))^* = f^*(g_1^*, \dots, g_m^*) = f(g_1, \dots, g_m) = h$ по теореме о суперпозиции двойственных функций.

Определение. Наборы $a = (a_1, \dots, a_n)$ и $a' = (\bar{a}_1, \dots, \bar{a}_n)$ из 0 и 1 называются *противоположными*.

Следствие. Функция самодвойственна, если и только если на всяких двух противоположных наборах она принимала разные значения.

Доказательство. Функция $f(x_1, \dots, x_n)$ самодвойственна $\leftrightarrow f(x_1, \dots, x_n) = f^*(x_1, \dots, x_n) \leftrightarrow f(x_1, \dots, x_n) = \bar{f}(\bar{x}_1, \dots, \bar{x}_n) \leftrightarrow f(x_1, \dots, x_n) \neq \bar{f}(\bar{x}_1, \dots, \bar{x}_n) \leftrightarrow$ на всяких двух противоположных наборах функция f принимает различные значения.

Пример. $f = 01001101, g = 01001111$. Функция f самодвойственна, а функция g несамоодвойственна, ибо $g(0,0,1) = g(1,1,0)$.

Лемма (о несамоодвойственной функции). Подстановкой функций x и $\bar{x}$ в несамоодвойственную функцию можно получить одну из констант.

Доказательство. Пусть $f(x_1, \dots, x_n)$ есть несамоодвойственная функция. Тогда существует набор $(a_1, \dots, a_n)$, для которого $f(a_1, \dots, a_n) = \bar{f}(\bar{a}_1, \dots, \bar{a}_n)$. Построим функцию $h(x)$, заменив единицы в $f(a_1, \dots, a_n)$ на x , а нули на $\bar{x}$. Так как $x = x^0, x = x^1$, то $h(x) = f(x^{a_1}, \dots, x^{a_n})$. Заметим, что $0^{a_i} = \bar{a}_i, 1^{a_i} = a_i$. Тогда $h(1) = f(1^{a_1}, \dots, 1^{a_n}) = f(a_1, \dots, a_n) = \bar{f}(\bar{a}_1, \dots, \bar{a}_n) = \bar{f}(0^{a_1}, \dots, 0^{a_n}) = h(0)$, т. е. $h(1) = h(0)$. Следовательно, функция $h(x)$ есть одна из констант.

1.8. Линейные функции



Арифметические функции в алгебре логики – это сложение и умножение по модулю два. Вот эти функции:

xu	$x + y$	$x \cdot y$
00	0	0
01	1	0
10	1	0
11	0	0

Вычитание (по модулю два) определяется как операция, обратная сложению, т. е. $x - y$ равно такому элементу z , для которого $x = y + z$. В частности, $0 - x$ равно такому y , что $0 = x + y$. Правая часть этого равенства равна нулю только при $y = x$. Поэтому $0 - x = x$. Отсюда следует, что $-x = x$ (по модулю два).

Деление (по модулю два) определяется как операция, обратная умножению, т. е. x/y равно такому z , для которого $x = y \cdot z$. В частности, $1/x$ равно такому y , для которого $1 = x \cdot y$. Правая часть равенства равна 1 только при $y = x = 1$. Так что обратный элемент $x^{-1} = 1/x$ возможен лишь при $x = 1$ и равен 1.

Следующие свойства арифметических операций проверяются непосредственно.

1. $x + (y + z) = (x + y) + z$.
2. $x + y = y + x$.
3. $x + 0 = x$.
4. $x + (-x) = 0$.
5. $x \cdot (y \cdot z) = (x \cdot y) \cdot z$.
6. $x \cdot y = y \cdot x$.
7. $x \cdot 1 = x$.
8. $x \cdot (x^{-1}) = 1$.
9. $x(y + z) = xy + xz$.

Двухэлементное множество $\{0, 1\}$ с операциями сложения и умножения по модулю два образует коммутативное поле F .

Справедливы также следующие свойства.

10. $x + x = 0$.
11. $x \cdot x = x$.

Так что поле F имеет характеристику два, а операция умножения идемпотентна.

Следующие четыре равенства устанавливают связь между арифметическими и логическими операциями.

1. $\bar{x} = x + 1$.
2. $x \& y = x \cdot y$.
3. $x \vee y = \overline{\bar{x} \& \bar{y}} = (x + 1)(y + 1) + 1 = xy + x + y$.
4. $x + y = \overline{xy \vee \bar{x} \bar{y}}$.



Многочлен Жегалкина в поле F есть выражение

$$\sum_{(i_1, \dots, i_n) \in E_2^n} a_{i_1, i_2, \dots, i_n} x_1^{i_1} x_2^{i_2} \dots x_n^{i_n},$$

где $x^i = \begin{cases} x, & \text{если } i = 1, \\ 1, & \text{если } i = 0, \end{cases}$ а каждый коэффициент $a_{i_1, i_2, \dots, i_n}$ равен 0 или 1.

Теорема (Жегалкин). Всякую функцию алгебры логики можно представить единственным полиномом Жегалкина.

Доказательство. Каждый многочлен Жегалкина определяет некоторую функцию алгебры логики. Два различных многочлена определяют различные

функции. Аналогично тому, как это делали при доказательстве единственности СДНФ, можно показать, что существует 2^{2^n} различных многочленов Жегалкина от n переменных.

Пример. Многочлен Жегалкина для функции $f(x, y, z) = (\bar{x} \vee \bar{y} \vee \bar{z})$.

$$\begin{aligned} f(x, y, z) &= \bar{x}\bar{y}z \vee \bar{x}y\bar{z} \vee x\bar{y}\bar{z} \vee x\bar{y}z \vee x\bar{y}z \vee xyz = \\ &= (x+1)(y+1)z + (x+1)y\bar{z} + x(y+1)\bar{z} + xy(z+1) + xyz = \\ &= xyz + xz + yz + z + xyz + yz + xyz + xz + xyz + xy + xyz = xyz + xy + z. \end{aligned}$$

Многочлен Жегалкина можно получить также с помощью треугольника Паскаля по единицам его левой стороны (табл. 1.16) следующим образом. Построим многочлен Жегалкина для функции $g = 10011110$. Верхняя сторона треугольника есть функция g . Любой другой элемент треугольника есть сумма по модулю два двух соседних элементов предыдущей строки. Левая сторона треугольника для функции g содержит шесть единиц. Многочлен Жегалкина будет содержать шесть слагаемых. Первая единица треугольника соответствует набору 000. Первое слагаемое многочлена есть 1. Третья снизу единица в левой стороне треугольника соответствует набору 101. В качестве слагаемого многочлена берем xz . Аналогично для других единиц треугольника. Слева от наборов показаны слагаемые многочлена Жегалкина.

Тогда $g(x, y, z) = 1 + z + y + xz + xy + xyz$.

Таблица 1.16

N	xyz	f	g	Треугольник Паскаля
1	000	0	1	$g = 1 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1 \ 0$
z	001	1	0	1 0 1 0 0 0 1
y	010	0	0	1 1 1 0 0 1
yz	011	1	1	0 0 1 0 1
x	100	0	1	0 1 1 1
xz	101	1	1	1 0 0
xy	110	1	1	1 0
xyz	111	1	0	1

Функция $f(x_1, \dots, x_n)$ называется линейной, если многочлен Жегалкина для нее имеет линейный относительно переменных вид:

$$f(x_1, \dots, x_n) = a_1x_1 + \dots + a_nx_n + a_{n+1},$$

где каждое a_i равно 0 или 1.

Теорема. Класс линейных функций замкнут относительно суперпозиции.

Доказательство. Пусть линейные функции

$$\begin{aligned} f(x_1, \dots, x_n) &= a_1x_1 + \dots + a_nx_n + a_{n+1}, \\ g_i(x_1, \dots, x_m) &= b_{i1}x_1 + \dots + b_{im}x_m + b_{i,m+1}, \\ i &= 1, 2, \dots, n \end{aligned}$$

для простоты зависят от одних и тех же переменных. Тогда их суперпозиция

$$\begin{aligned} f(g_1, \dots, g_n) = & a_1(b_{11}x_1 + \dots + b_{1m}x_m + b_{1,m+1}) + \\ & a_2(b_{21}x_1 + \dots + b_{2m}x_m + b_{2,m+1}) + \\ & \dots \\ & a_n(b_{n1}x_1 + \dots + b_{nm}x_m + b_{n,m+1}) + a_{n+1} = \\ & (a_1b_{11} + \dots + a_nb_{n1})x_1 + \dots + (a_1b_{1m} + \dots + a_nb_{nm})x_m + (a_1b_{1,m+1} + \dots + a_nb_{n,m+1}) + a_{n+1} \end{aligned}$$

есть линейная функция.

Лемма (о нелинейной функции). Суперпозицией нелинейной функции, отрицания и константы 1 можно получить конъюнкцию.

Доказательство. Пусть $f(x_1, \dots, x_n)$ есть нелинейная функция. Тогда полином Жегалкина для нее содержит слагаемое, в котором присутствует произведение x_1x_2 . Будем считать для простоты, что x_1x_2 в многочлене Жегалкина является этим произведением. Произведя группировку слагаемых, функцию f представим в виде

$$f(x_1, \dots, x_n) = x_1x_2 \cdot h_0(x_3, \dots, x_n) + x_1 \cdot h_1(x_3, \dots, x_n) + x_2 \cdot h_2(x_3, \dots, x_n) + h_3(x_3, \dots, x_n).$$

Функция h_0 не есть тождественный нуль, иначе в полиноме Жегалкина отсутствует слагаемое с произведением x_1x_2 . Тогда существует набор $(a_3, \dots, a_n)$ из 0 и 1, для которого $h_0(a_3, \dots, a_n) = 1$. Пусть $h_1(a_3, \dots, a_n) = a$, $h_2(a_3, \dots, a_n) = b$, $h_3(a_3, \dots, a_n) = c$. Тогда функция $g(x_1, x_2) = f(x_1, x_2, a_3, \dots, a_n) = x_1x_2 + ax_1 + bx_2 + c$.

Построим функцию

$$\begin{aligned} h(x_1, x_2) &= g(x_1 + b, x_2 + a) = \\ &= (x_1 + b)(x_2 + a) + a(x_1 + b) + b(x_2 + a) + c = \\ &= x_1x_2 + ax_1 + bx_2 + ab + ax_1 + ab + bx_2 + ab + c = x_1x_2 + d, \end{aligned}$$

где $d = ab + c$. Если $d = 0$, то $h(x_1, x_2) = x_1x_2$. Если $d = 1$, то $h(x_1, x_2) = x_1x_2 + 1$, и тогда $x_1x_2 = h(x_1, x_2)$. Лемма доказана.

Определение. Функция $f(x_1, \dots, x_n)$ сохраняет константу $a \in \{0, 1\}$, если $f(a, \dots, a) = a$.

Пример. Функция x_1x_2 сохраняет 0, сохраняет 1. Функция $x \rightarrow y$ сохраняет 1 и не сохраняет 0.

Теорема. Класс функций, сохраняющих константу, замкнут относительно суперпозиции.

Доказательство. Пусть функции $f(x_1, \dots, x_n)$ и $g_i(x_1, \dots, x_m)$, $i = 1, 2, \dots, n$, сохраняют константу a . Тогда их суперпозиция $h = f(g_1, \dots, g_n)$ сохраняет константу a , ибо $h(a, \dots, a) = f(g_1(a, \dots, a), \dots, g_n(a, \dots, a)) = f(a, \dots, a) = a$.

1.9. Монотонные функции

Определение. Если $a = (a_1, \dots, a_n)$ и $b = (b_1, \dots, b_n)$ есть наборы длины n из 0 и 1, то $a \leq b$, если $a_1 \leq b_1, \dots, a_n \leq b_n$.

Пример. $(0,1,0) \leq (1,1,0)$. Наборы $(0,1)$ и $(1,0)$ несравнимы. Также несравнимы наборы $(0,1)$ и $(0,1,0)$.

Определение. Функция $f(x_1, \dots, x_n)$ называется *монотонной*, если для всяких наборов $a = (a_1, \dots, a_n)$, $b = (b_1, \dots, b_n)$ условие $a \leq b$ влечет $f(a) \leq f(b)$.

Теорема. Класс монотонных функций замкнут относительно суперпозиции.

Доказательство. Пусть $f(x_1, \dots, x_m)$ и $g_i(x_1, \dots, x_n)$, $i = 1, 2, \dots, n$, есть монотонные функции (для простоты от одних и тех же переменных). Покажем, что их суперпозиция $h(x_1, \dots, x_n) = f(g_1(x_1, \dots, x_n), \dots, g_n(x_1, \dots, x_n))$ монотонна. Пусть $c = (c_1, \dots, c_n)$, $d = (d_1, \dots, d_n)$ есть произвольные наборы из 0 и 1, для которых $c \leq d$, т. е. $c_i \leq d_i$, $i = 1, 2, \dots, n$. Пусть $a_i = g_i(c)$, $b_i = g_i(d)$, $i = 1, 2, \dots, m$. Пусть $a = (a_1, \dots, a_m)$, $b = (b_1, \dots, b_m)$. Так как функции g_i монотонны и $c \leq d$, то $g_i(c) \leq g_i(d)$, т. е. $a_i \leq b_i$, $i = 1, 2, \dots, m$. Следовательно, $a \leq b$. Отсюда в силу монотонности функции f получаем, что $f(a) \leq f(b)$. Тогда $h(c) = f(g_1(c), \dots, g_n(c)) = f(a_1, \dots, a_m) = f(a) \leq f(b) = f(b_1, \dots, b_m) = f(g_1(d), \dots, g_n(d)) = h(d)$.

Итак, для любых наборов c и d из 0 и 1 неравенство $c \leq d$ влечет $h(c) \leq h(d)$, т. е. функция h монотонна. Теорема доказана.

Лемма (о немонотонной функции). Суперпозицией констант 0 и 1 и переменной x в немонотонную функцию можно получить отрицание.

Доказательство. Пусть $f(x_1, \dots, x_n)$ есть немонотонная функция. Тогда существуют наборы $a = (a_1, \dots, a_n)$ и $b = (b_1, \dots, b_n)$, для которых $a \leq b$, но $f(a) > f(b)$. Пусть $i_1, \dots, i_k$ есть все те номера аргументов, для которых $a_i < b_i$, $p = 1, \dots, k$. На всех остальных аргументных местах j имеем $a_j = b_j$. В выражении $f(a_1, \dots, a_n)$ заменим нули на местах $i_1, \dots, i_k$ на x . В результате получим функцию $g(x)$, для которой $g(0) = f(a) = 1$ и $g(1) = f(b) = 0$. Функция $g(x)$ является отрицанием.

Теорема (критерий монотонности функции). Функция монотонна, если и только если ее сокращенная ДНФ не содержит отрицаний.

Доказательство. Пусть простой импликант

$$I = x_{i_1}^{c_1} \dots x_{i_{j-1}}^{c_{j-1}} \cdot \overline{x_{i_j}^{c_j}} \cdot x_{i_{j+1}}^{c_{j+1}} \dots x_{i_t}^{c_t}$$

монотонной функции $f(x_1, \dots, x_n)$ содержит отрицание переменной. Покажем, что выражение

$$J = x_{i_1}^{c_1} \dots x_{i_{j-1}}^{c_{j-1}} \cdot x_{i_{j+1}}^{c_{j+1}} \dots x_{i_t}^{c_t}$$

есть импликант функции f . Очевидно, что $I = J \cdot \overline{x_{i_j}}$.

Пусть $a = (a_1, \dots, a_n)$ есть произвольный набор из 0 и 1 длины n , для которого $J(a) = 1$. Покажем, что $f(a) = 1$. Возможны два случая. Пусть набор a есть $b = (a_1, \dots, a_{i_{j-1}}, 0, a_{i_{j+1}}, \dots, a_n)$.

Тогда $I(b) = J(b) \cdot \overline{0} = 1$, откуда $f(b) = 1$, ибо I есть импликант для f . Так как a есть b , то $f(a) = 1$.

Пусть теперь набор a есть $c = (a_1, \dots, a_{i_{j-1}}, 1, a_{i_{j+1}}, \dots, a_n)$.

Так как $b \leq c$, то $1 = f(b) \leq f(c)$, откуда $f(c) = 1$. Поэтому $f(a) = 1$. В обоих случаях $f(a) = 1$ влечет $f(a) = 1$, т. е. f есть импликант функции f . Противоречие с простотой импликанта I .

В обратную сторону доказательство очевидно.

Следствие. Функция монотонна, если и только если ее минимальная ДНФ не содержит отрицаний.

1.10. Теорема Поста о функциональной полноте

Теорема (Е. Пост). Чтобы система функций из P_2 была функционально полной (в P_2), необходимо и достаточно, чтобы эта система содержала:

- 1) функцию, не сохраняющую 0;
- 2) функцию, не сохраняющую 1;
- 3) несамодвойственную функцию;
- 4) немонотонную функцию;
- 5) нелинейную функцию.

Доказательство. Пусть система функций F из P_2 полна в P_2 . Допустим, что в F нет одной из указанных функций, например немонотонной. Тогда все функции в F монотонны, а так как класс монотонных функций замкнут относительно суперпозиции, то никакая суперпозиция над F не даст немонотонной функции.

Пусть система F содержит все указанные функции:

- $f_1(x_1, \dots, x_n)$ не сохраняет 0;
- $f_2(x_1, \dots, x_n)$ не сохраняет 1;
- $f_3(x_1, \dots, x_n)$ несамодвойственная функция;
- $f_4(x_1, \dots, x_n)$ немонотонная функция;
- $f_5(x_1, \dots, x_n)$ нелинейная функция.

Покажем, что суперпозицией функций системы F можно получить полную систему $G = \{x \& y, x\}$.

1. Пусть $g(x) = f_1(x, \dots, x)$. Тогда $g(0) = f_1(0, \dots, 0) = 1$. Далее возможны два случая:
 - $g(1) = 1$. Тогда $g(x) \equiv 1$. Функция $h(x) = f_2(g(x), \dots, g(x)) = f_2(1, \dots, 1) = 0$, т. е. $h(x) \equiv 0$. Получили константы 0 и 1;
 - $g(1) = 0$. Тогда $g(x) = x$. По лемме о несамодвойственной функции суперпозицией над $\{f_3, x\}$ можно получить одну из констант, например 0. Тогда $f_1(0, \dots, 0) = 1$ есть другая константа.

В обоих случаях получили обе константы.

2. По лемме о немонотонной функции суперпозицией над $\{f_4, 0, 1\}$ можно получить отрицание.

3. По лемме о нелинейной функции суперпозицией над $\{f_5, 1, \bar{x}\}$ можно получить конъюнкцию. Теорема доказана.

Примеры.

1. $F = \{1, x*y\}$, где $x*y = 0010$.

Проверяем условия полноты.

- 1) константа 1 не сохраняет 0;



- 2) $x * y$ не сохраняет 1;
- 3) $x * y$ несамоудовлетворительна, ибо $0 * 0 = 1 * 1$;
- 4) $x * y$ немонотонна, ибо $(1, 0) \leq (1, 1)$, но $1 * 0 > 1 * 1$;
- 5) $x * y$ нелинейна, ибо $x * y = \overline{xy} = xy + y$.

Следовательно, система F по теореме Поста полна. Отрицание есть $1 * x$.

Конъюнкция $x \& y = \overline{x * y}$.

2. $F = \{0, 1, \overline{x}, m(x, y, z)\}$, где функция $m(x, y, z) = 00010111$ равна единице на тех и только тех наборах, в которых число единиц больше числа нулей.

Проверяем условия полноты:

- 1) 0 не сохраняет 1;
- 2) 1 не сохраняет 0;
- 3) константа 1 несамоудовлетворительна;
- 4) отрицание немонотонно;
- 5) функция $m(x, y, z) = xyz + xy + xz + yz$ не линейна.

По теореме Поста система F полна. Конъюнкция $x \& y = m(x, y, 0)$.

Заметим, что система F избыточно полна. Ее подсистемы

$\{1, \overline{x}, m(x, y, z)\}$ и $\{0, \overline{x}, m(x, y, z)\}$

являются функционально полными системами.

Замечание. Если система функций F из P_2 функционально полна, то двойственная система F^* тоже функционально полна.

Приведем примеры функционально полных систем, являющихся базисами в P_2 , и примеры представления функций формулами в этих базисах.

1. $\{x \uparrow y\}$, базис Вебба (базис Пирса).

$$\begin{aligned} f(x, y, z) &= \overline{xyz} \vee \overline{xz} = (x \vee z)(\overline{xy} \vee \overline{z}) = (x \vee z)(\overline{xy} \vee \overline{z}) = \overline{x \vee z \vee xy \vee z} = \\ &= \overline{(x \uparrow z) \uparrow ((xy) \uparrow (z \uparrow z))} = (x \uparrow z) \uparrow ((x \uparrow y) \uparrow (z \uparrow z)). \end{aligned}$$

2. $\{x|y\}$, базис Шеффера.

$$\begin{aligned} f(x, y, z) &= \overline{xyz} \vee \overline{xz} = \overline{(xyz \vee xz)} = (\overline{x} \vee \overline{z})(\overline{yz} \vee \overline{x}) = \\ &= (x|z) \cdot ((y|z)|(x|x)) = (x|z)|((y|z)|(x|x)) = \\ &= [(x|z)|((y|z)|(x|x))]|[(x|z)|((y|z)|(x|x))]. \end{aligned}$$

3. $\{0, x \rightarrow y\}$, импликативный базис.

$$f(x, y, z) = \overline{xyz} \vee \overline{xz} = \overline{xyz \vee xz} = \overline{x \vee z \vee x \vee y \vee z \vee 0} = (x \rightarrow z) \rightarrow ((z \rightarrow (y \rightarrow x)) \rightarrow 0).$$

4. $\{1, x \leftrightarrow y\}$, коимпликативный базис, где $x \leftrightarrow y = \overline{x \rightarrow y}$.

$$\begin{aligned} f(x, y, z) &= \overline{xyz} \vee \overline{xz} = (x \vee z)(\overline{xy} \vee \overline{z}) = ((1 \rightarrow x) \vee z) \cdot (z \rightarrow (y \leftrightarrow x)) = \\ &= ((1 \rightarrow x) \rightarrow z) \leftrightarrow (z \leftrightarrow (y \leftrightarrow x)) = ((1 \rightarrow x) \leftrightarrow z) \leftrightarrow (z \leftrightarrow (y \leftrightarrow x)). \end{aligned}$$

5. $\{x \rightarrow y, x \nrightarrow y\}$.

$$f(x, y, z) = \overline{x}yz \vee x\overline{z} = x \nrightarrow z \vee y \cdot \overline{x \vee z} = x \nrightarrow z \vee y \cdot \overline{x \vee z} = x \nrightarrow z \vee y \nrightarrow (z \rightarrow x) = (x \rightarrow z) \rightarrow (y \nrightarrow (z \rightarrow x)).$$

6. $\{x+y, x \rightarrow y\}$.

$$f(x, y, z) = \overline{x}yz \vee x\overline{z} = (x \vee \overline{y}z)\overline{z} \vee \overline{x}(y \vee \overline{z})z = (x \vee \overline{y}z)\overline{z} \vee x \vee \overline{y}zz = (x \vee \overline{y}z) + z = (\overline{z} \vee y \vee x) + z = ((z \rightarrow y) \rightarrow x) + z.$$

7. $\{x \equiv y, x \nrightarrow y\}$.

$$f(x, y, z) = \overline{x}yz \vee x\overline{z} = xyz + xz + yz + x = (xyz + xz + yz + z) + x + z = (xy + x + y + 1)z + x + z = \overline{x}yz + x + z = ((z \nrightarrow x) \nrightarrow y) \equiv x \equiv z.$$

8. $\{\neg x, x \rightarrow y\}$, импликативный базис.

$$f(x, y, z) = \overline{x}yz \vee x\overline{z} =_5 (x \rightarrow z) \rightarrow (y \nrightarrow (z \rightarrow x)) = (x \rightarrow z) \rightarrow \overline{(y \rightarrow (z \rightarrow x))}.$$

Знак $=_5$ означает, что равенство написано на основании формулы 5.

9. $\{\neg x, x \nrightarrow y\}$, коимпликативный базис.

$$f(x, y, z) = \overline{x}yz \vee x\overline{z} = (*69* \vee z)(\overline{z} \vee \overline{xy}) = \overline{x \rightarrow z} \cdot \overline{z \rightarrow (y \nrightarrow x)} = \overline{x \nrightarrow z} \cdot \overline{y \nrightarrow x} = \overline{x \nrightarrow z} \nrightarrow (z \nrightarrow (y \nrightarrow x)).$$

10. $\{x \vee y, \neg x\}$, дизъюнктивный базис Буля.

$$f(x, y, z) = \overline{x}yz \vee x\overline{z} = \overline{x \vee z} \cdot (x \vee y \vee \overline{z}).$$

11. $\{x \& y, \neg x\}$, конъюнктивный базис Буля.

$$f(x, y, z) = \overline{x}yz \vee x\overline{z} = (\overline{x} \vee \overline{z}) \cdot \overline{x \vee yz} = \overline{x} \cdot \overline{z} \cdot \overline{x \vee y} \cdot \overline{z}.$$

12. $\{x \equiv y, x \& y, 0\}$.

$$f(x, y, z) = \overline{x}yz \vee x\overline{z} = (x + z) \cdot (z \rightarrow y) = (x + z + 0) \cdot (yz + z + 1) = (x \equiv z \equiv 0) \cdot (y \cdot z \equiv z).$$

13. $\{x+y, x \vee y, 1\}$.

$$f(x, y, z) = \overline{x}yz \vee x\overline{z} = xyz + xz + yz + x = x(yz + z) + (yz + z) + x + z = ((yz + z) \vee x) + z = (x \vee ((yz + z) + y)) + z = (x \vee ((y \vee z) + y)) + z.$$

14. $\{x \equiv y, x \vee y, 0\}$.

$$f(x, y, z) = \overline{x}yz \vee x\overline{z} = (x \equiv z) \uparrow (x \uparrow y) = 0 \equiv ((x \equiv z) \vee (0 \equiv (x \vee y))).$$

15. $\{x + y, x \& y, 1\}$, базис Жегалкина.

$$f(x,y,z) = \bar{x}yz \vee x\bar{z} = x + xz + yz + xyz = (x + xyz) + (xz + yz) = x(x + yz) + z(x + yz) = (x + y \cdot z) \cdot (x + z).$$

16. $\{x \equiv y, x + y, x \vee y\}$. Преобразование функции $f(x,y,z) = \bar{x}yz \vee x\bar{z}$ совпадает с преобразованиями функции f в пункте 13.

17. $\{x \equiv y, x + y, x \& y\}$. Преобразование функции $f(x,y,z) = \bar{x}yz \vee x\bar{z}$ совпадает с преобразованиями функции f в пункте 15.

1.11. Предполные классы

Определение. Замкнутый класс K из P_2 предполон, если K не является полным, но для всякой функции f не из K система $K \cup \{f\}$ полна.

Теорема. Следующие замкнутые классы функций предполны: класс T_0 функций, сохраняющих константу 0; класс T_1 функций, сохраняющих константу 1; класс M монотонных функций; класс L линейных функций; класс S самодвойственных функций.

Доказательство. Покажем, что класс монотонных функций предполон. Пусть функция $f \notin M$. Заметим, что функции $x \& y, x \vee y, 0, 1$ монотонны. По лемме о немонотонной функции отрицание $\bar{x} \in \{f, 0, 1\}$. Система $\{x \& y, x \vee y, x\}$ полна.

Покажем, что класс T_1 предполон. Пусть функция $f(x_1, \dots, x_n) \notin T_1$, т. е. $f(1, \dots, 1) = 0$. Пусть $g(x) = f(x, \dots, x)$. Тогда $g(1) = f(1, \dots, 1) = 0$. Заметим, что функции $x \vee y, x \& y$ лежат в T_1 . Возможны следующие случаи.

- а) $g(0) = 1$. Тогда $g(x) = \bar{x}$. Следовательно, система $x \& y, \bar{x} \subseteq [T_1 \cup \{f\}]$, и потому класс $T_1 \cup \{f\}$ полон;
- б) $g(0) = 0$. Тогда $g(x) \equiv 0$. Очевидно, что $x \rightarrow y \in T_1$, $\bar{x}$ есть $x \rightarrow 0$. Получаем $\{x \& y, x\} \subseteq [T_1 \cup \{f\}]$, и потому класс $T_1 \cup \{f\}$ полон.

Покажем, что класс T_0 предполон. Пусть функция $f(x_1, \dots, x_n) \notin T_0$. Пусть $g(x) = f(x, \dots, x)$ и $g(0) = f(0, \dots, 0) = 1$. Возможны следующие случаи.

- $g(1) = 0$. Тогда $g(x) = \bar{x}$.
- $g(1) = 1$. Тогда $g(x) \equiv 1$.

Так как $x + y \in T_0$, то $x = x + 1$. В обоих случаях получено отрицание. Так как $x \& y \in T_0$, то $\{x \& y, x\} \subseteq [T_0 \cup \{f\}]$. Следовательно, класс T_0 предполон.

Покажем, что класс L линейных функций предполон. Пусть $f(x_1, \dots, x_n) \notin L$. Заметим, что функции $x + 1, 0, 1$ лежат в L . По лемме о нелинейной функции $x \& y \in [f, 0, 1]$. Следовательно, $\{x \& y, x\} \subseteq [L \cup \{f\}]$, и потому система $L \cup \{f\}$ полна.

Покажем, что класс S самодвойственных функций предполон. Пусть функция $f \notin S$. Заметим, что $x \in S$. По лемме о несамодвойственной функции константы 0 и 1 лежат в $\{f, \bar{x}\}$. Функция $g(x,y,z) = xy \vee xz \vee yz \in S$. Тогда $g(x,y,1) = xy \vee x \vee y = x \vee y$. Система $\{x \vee y, x\} \subseteq [S \cup \{f\}]$. Следовательно, система $S \cup \{f\}$

полна, и потому класс S самодвойственных функций предполон. Теорема доказана.

Приведем следующую перефразировку теоремы Поста.

Теорема. Система функций полна, если и только если она не содержится целиком ни в одном из пяти предполных классов.

Замечание. Пост описал все замкнутые классы в P_2 . Оказалось, что множество всех замкнутых классов в P_2 счетно и образует решетку по отношению к отношению включения замкнутых классов. Эта решетка имеет единственный наибольший элемент класса P_2 и три минимальных элемента: $O_1 = [x]$, $O_2 = [1]$, $O_3 = [0]$.



Функции k -значной логики

2.1. Функции и отношения

Пусть $E_k = \{0, 1, \dots, k-1\}$, $k \geq 2$, $E_k^1 = E_k$, $E_k^2 = E_k \times E_k$, $E_k^n = E_k^{n-1} \times E_k$. Функции k -значной логики есть n -арные функции вида $f(x_1, x_2, \dots, x_n): E_k^n \rightarrow E_k$. Константы $0, 1, \dots, k-1$ есть 0-арные функции.

Пусть P_k есть множество всех функций k -значной логики.

h -арное отношение ρ на E_k есть некоторое подмножество из E_k^h .

Определение. Функция $f(x_1, x_2, \dots, x_n)$ из P_k сохраняет h -арное отношение ρ на E_k (или h -арный предикат ρ), если для всяких n наборов из ρ :

$$(a_1^0, a_1^1, \dots, a_1^{h-1}) \in \rho,$$

$$(a_2^0, a_2^1, \dots, a_2^{h-1}) \in \rho,$$

...

$$(a_n^0, a_n^1, \dots, a_n^{h-1}) \in \rho,$$

$$a^0 \quad a^1 \quad a^{h-1}$$

$$\text{набор } (f(a^0), f(a^1), \dots, f(a^{h-1})) \in \rho.$$

Определение. Функция $f(x_1, x_2, \dots, x_n)$ существенно зависит от переменной x_i , если $f(a_1, a_2, \dots, a_{i-1}, b, a_{i+1}, \dots, a_n) \neq f(a_1, a_2, \dots, a_{i-1}, c, a_{i+1}, \dots, a_n)$ для некоторых элементов $a_1, a_2, \dots, a_{i-1}, b, c, a_{i+1}, \dots, a_n$ из E_k . Функция f называется существенной, если у нее имеются, по меньшей мере, две переменные, от которых она зависит существенно.

Определение. Понятие суперпозиции функций над множеством $A \subseteq P_k$ определяется по индукции:

- 1) каждая функция f из A есть суперпозиция над A ;
- 2) если $g_0(x_1, \dots, x_n)$ есть суперпозиция над A и если для каждого $i = 1, 2, \dots, n$ функция $g_i(x_{i1}, \dots, x_{im_i})$ есть либо суперпозиция над A , либо $g_i(x_{i1}, \dots, x_{im_i}) = x_{il}$, где x_{il} ($1 \leq l \leq m_i$) есть любая переменная, то функция $g_0(g_1(x_{11}, \dots, x_{1m_1}), \dots, g_n(x_{n1}, \dots, x_{1m_n}))$ является суперпозицией над A .

Определение. Множество A функций из P_k называется (функционально) замкнутым классом относительно операции суперпозиции (или просто замкнутым классом), если вместе с любыми функциями из A множеству A принадлежит и их суперпозиция.

Определение. Замыканием множества A из P_k называется множество всех суперпозиций над A .

Замыкание множества A обозначают через $[A]$. Очевидно, что $A \subseteq [A]$. Множество A есть замкнутый класс, если $[A] = A$.

Определение. Множество A функций из P_k называется полным в замкнутом классе $M \subseteq P_k$, если $[A] = M$. Система A называется полной (в P_k), если $[A] = P_k$.

Определение. Множество A из P_k называется предполным классом, если $A \neq P_k$ и для всякой функции f , не принадлежащей A , система функций $A \cup \{f\}$ является полной.

В табл. 2.1 приведены примеры функций из P_k . Определим также следующие функции:

$$\begin{aligned} x \vee y &= \max(x, y), \quad x \& y = \min(x, y), \\ x + y &= \text{rest}(x+y, k), \quad x \cdot y = \text{rest}(x \cdot y, k), \end{aligned}$$

где $\text{rest}(x, k)$ означает остаток от деления x на k , а выражения $x + y$ и $x \cdot y$ в скобках являются обычными суммой и произведением.

Опишем некоторые функционально полные системы.

Теорема. Система функций $\{0, 1, \dots, k-1, J_0(x), J_1(x), \dots, J_{k-1}(x), \max(x, y), \min(x, y)\}$ полна (в P_k).

Доказательство. Справедливость теоремы вытекает из тождества

$$f(x_1, \dots, x_n) = \bigvee_{(i_1, \dots, i_n) \in E_k^n} J_{i_1}(x_1) \& \dots \& J_{i_n}(x_n) \& f(i_1, \dots, i_n). \quad (2.1)$$

Докажем его. Пусть $(a_1, \dots, a_n)$ есть произвольный набор из E_k .

Рассмотрим равенство (2.1) на этом наборе:

$$f(a_1, \dots, a_n) = \bigvee_{(i_1, \dots, i_n) \in E_k^n} J_{i_1}(a_1) \& \dots \& J_{i_n}(a_n) \& f(i_1, \dots, i_n). \quad (2.2)$$

Таблица 2.1

x	$\bar{x}$	Nx	$J_i(x)$	$j_i(x)$
0	1	$k-1$	0	0
1	2	$k-2$	0	0
...	...	...	...	...
i	$i+1$	$k-i-1$	$k-1$	1
...	...	...	...	...
$k-2$	$k-1$	1	0	0
$k-1$	0	0	0	0

Если $(i_1, \dots, i_n) \neq (a_1, \dots, a_n)$, то при некотором j будет $i_j \neq a_j$, и тогда $J_{i_j}(a_j) = 0$. В правой части (2.2) останется лишь одно слагаемое, то, у которого $(i_1, \dots, i_n) = (a_1, \dots, a_n)$, т. е. слагаемое $(k-1) \& (k-1) \& \dots \& (k-1) \& f(a_1, \dots, a_n)$. Оно равно $f(a_1, \dots, a_n)$. Поскольку в представлении (2.1) участвуют лишь функции, указанные в теореме, то приведенная система функций полна (в P_k).

Теорема. Система функций $\{x + 1, \max(x, y)\}$ полна в P_k .

Доказательство. Построим функции $x + l$, $0 \leq l \leq k-1$. Имеем $x + 2 = (x + 1) + 1$, $x + 3 = (x + 2) + 1$, ..., $x + (k-1) = (x + (k-2)) + 1$, $x = (x + (k-1)) + 1$. Очевидно, что $x \vee (x + 1) \vee (x + 2) \vee \dots \vee (x + (k-1)) \equiv k-1$. Теперь можно получить остальные константы: $0 = (k-1) + 1$, $1 = 0 + 1$, ..., $(k-2) = (k-3) + 1$. Далее $J_i(x) = 1 + \max\{x + l : l \neq k-1-i\}$. В самом деле, пусть $x = i$. Тогда левая и правая части этого соотношения равны соответственно $J_i(i) = k-1$ и $1 + \max\{i + l : l \neq k-1-i\} = 1 + \max\{i + l : i + l \neq k-1\} = k-1$. Пусть $x = j \neq i$. Тогда слева имеем $J_i(j) = 0$, а справа $1 + \max\{j + l : l \neq k-1-i\} = 1 + \max\{j, j+1, \dots, j+k-1-j, \dots, j+k-1\} = 1 + k-1 = 0$.

Построим функции $f_{i,s}(x) = \begin{cases} s, & \text{если } x = i, \\ 0, & \text{если } x \neq i, \end{cases} \quad i, s \in E_k$.

Имеем $f_{i,s}(x) = s + 1 + \max\{J_i(x), k-1-s\}$.

Это равенство подтверждается серией графиков на рис. 2.1.

Пусть $\varphi(x)$ есть произвольная унарная функция из P_k (рис. 2.2). Очевидно, что $f_{i,\varphi(s)}(x) = \begin{cases} \varphi(i), & \text{если } x = i, \\ 0, & \text{если } x \neq i. \end{cases}$ Тогда $\varphi(x) = \max\{f_{i,\varphi(s)}(x) : i = 0, 1, \dots, k-1\}$. Значит,

любая одноместная функция, в том числе и функция Nx , выражается через $x + 1$ и $\max(x, y)$. Остается заметить, что $\min(x, y) = N(\max(Nx, Ny))$ (см. рис. 2.2).

Следствие. Функция $v(x, y) = \max(x, y) + 1$, называемая часто функцией Вебба, образует полную в P_k систему.

Доказательство. Имеем $x + 1 = v(x, x)$ и $\max(x, y) = \max(x, y) + 1 + k-1 = v(x, y) + k-1$.

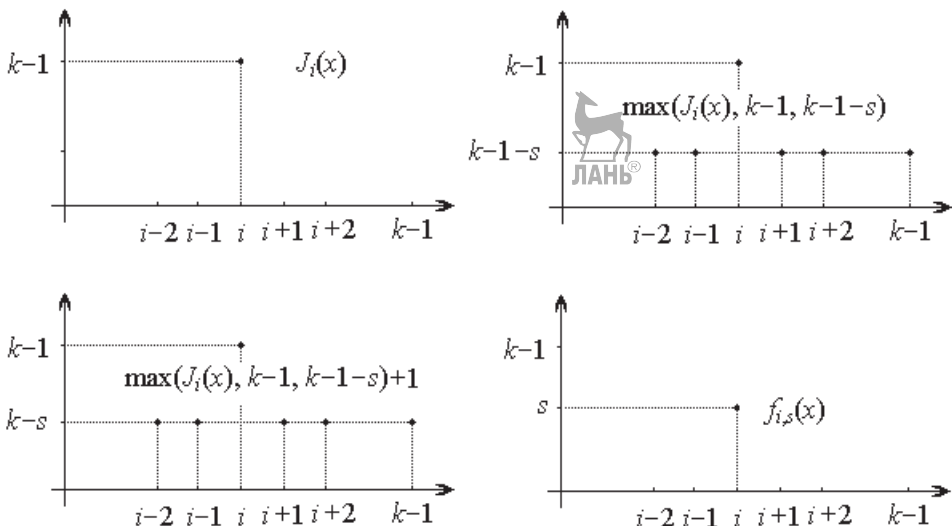


Рис. 2.1

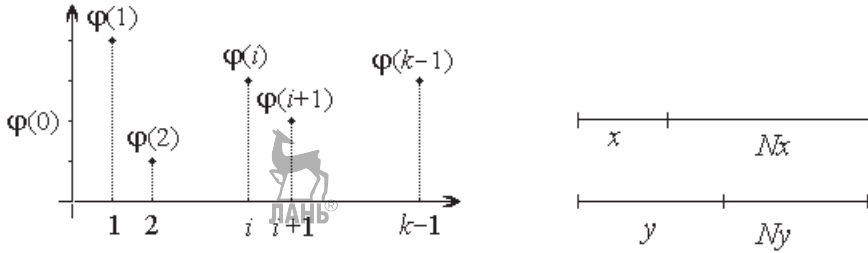


Рис. 2.2

2.2. Самодвойственные функции

Определение. Пусть $s(x)$ есть некоторая подстановка на E_k . Функция $f^s(x_1, \dots, x_n) = s^{-1}(f(s(x_1), \dots, s(x_n)))$ называется двойственной к функции $f(x_1, \dots, x_n)$ относительно $s(x)$.

Определение. Пусть $s(x) \in S_k$. Функция $f(x_1, \dots, x_n) \in P_k$ называется самодвойственной относительно $s(x)$, если $f^s = f$, т. е. если она двойственна самой себе относительно $s(x)$.

Пусть $S_k(s(x))$ есть множество всех самодвойственных относительно $s(x)$ функций из P_k .

Утверждение. Множество $S_k(s(x))$ является замкнутым классом.

Теорема. Если подстановка $s(x)$ на E_k имеет только циклы одинаковой простой длины r , то класс $S_k(s(x))$ самодвойственных относительно $s(x)$ функций предполон.

Пусть $\mathbb{S}$ есть совокупность всех предполных классов самодвойственных функций.

2.3. Монотонные функции

Определение. Пусть E есть некоторое множество. Частичным порядком на E называется бинарное отношение $\leq$, удовлетворяющее следующим аксиомам: $\forall a \in E \forall b \in E \forall c \in E$.

- 1) $a \leq a$;
- 2) $a \leq b \ \& \ b \leq a \rightarrow a = b$;
- 3) $a \leq b \ \& \ b \leq c \rightarrow a \leq c$.

Если $a \leq b$, то элементы a и b называются сравнимыми. В противном случае a и b несравнимы.

Множество E с заданным на нем частичным порядком называется частично упорядоченным.

Элемент e частично упорядоченного множества E называется наибольшим в E , если $\forall a \in E (a \leq e)$. Элемент o из E называется наименьшим в E , если $\forall a \in E (o \leq a)$. Если наибольший (равно как и наименьший) элемент в частично упорядоченном множестве существует, то он единственный.

Элемент m из E называется максимальным в E , если $\forall a \in E$ условие a сравнимо с m влечет $a \leq m$. Аналогично элемент n минимален в E , если $\forall a \in E$ условие a сравнимо с n влечет $n \leq a$.

Пусть $\mathbb{O}$ есть множество всех отношений частичного порядка с единственным максимальным и единственным минимальным элементами.

Теорема (В. В. Мартынюк). Пусть $\rho \in \mathbb{O}$. Тогда A_ρ есть предполный класс.

Пусть $\mathbb{M}$ есть совокупность всех предполных классов монотонных функций.

2.4. Линейные функции

Определение. Пусть $G = (E_k, +)$ есть абелева группа, определенная на E_k с операцией сложения «+» и нулем o . Функция $f(x_1, \dots, x_n)$ из P_k называется линейной относительно G , если $\forall \mathbf{a} = (a_1, \dots, a_n) \in E_k^n$ и $\forall \mathbf{b} = (b_1, \dots, b_n) \in E_k^n$ $f(\mathbf{a} + \mathbf{b}) = f(\mathbf{a}) + f(\mathbf{b}) - f(o)$, где $\mathbf{a} + \mathbf{b} = (a_1 + b_1, \dots, a_n + b_n)$, $o = (o, \dots, o) \in E_k^n$.

Обозначим через L_G множество всех функций, линейных относительно G .

Теорема. L_G есть предполный класс.

Пусть $\mathbb{L}$ есть совокупность всех предполных классов линейных функций.

2.5. Функции, сохраняющие разбиение

Определение. Функция $f(x_1, \dots, x_n)$ из P_k сохраняет разбиение $\mathfrak{E}$ множества E_k , если f сохраняет отношение эквивалентности ρ , порожденное этим разбиением.

Через Eq обозначим класс всех отношений эквивалентности на E_k , отличных от тривиальных отношений ι_2 и π_2 , порождающих разбиения $\langle \{0\}, \{1\}, \dots, \{k-1\} \rangle_k$ и $\langle 0, 1, \dots, k-1 \rangle_k$ соответственно.

Теорема. Если $\rho \in Eq$, то A_ρ есть предполный класс.

Пусть $\mathbb{E}$ есть совокупность всех предполных классов, сохраняющих разбиения множества E_k .

2.6. Классы типа $\mathbb{C}$

Через ι_h , $h \geq 2$, обозначим h -арное отношение на E_k , состоящее из всех наборов множества E_k^h , каждый из которых имеет хотя бы два одинаковых элемента.

Определение. h -арное отношение ρ на E_k ($h \geq 1$) называется тотально рефлексивным, если $\rho \supseteq \iota_h$.

Всякое унарное отношение является тотально рефлексивным. Через π_h обозначим h -арное отношение на E_k ($h \geq 1$), совпадающее с E_k^h . Через σ_h обозначается h -арное отношение на E_k ($h \geq 1$), состоящее из всех таких наборов множества E_k^h , в каждом из которых все элементы разные.

Подстановка на E_k есть взаимно-однозначное отображение $s(x): E_k \rightarrow E_k$. Через S_k обозначим множество всех подстановок на E_k .

Определение. Пусть $1 \leq h \leq k$, $s(x) \in S_h$ и ρ есть h -арное отношение на E_k . Определим отношение ρ_s (s -сдвиг отношения ρ) с помощью соотношения

$$(a_0, \dots, a_{h-1}) \in \rho_s \leftrightarrow (a_{s(0)}, \dots, a_{s(h-1)}) \in \rho.$$

Далее отношение:

- ρ симметрично относительно $s \leftrightarrow \rho = \rho_s$;
- ρ тотально симметрично $\leftrightarrow \rho = \rho_s \forall s \in S_h$;
- ρ асимметрично относительно $s \leftrightarrow \rho \cap \rho_s = \emptyset$.

Через $\mathbb{T}_h$ ($2 \leq h \leq k$) обозначим множество всех h -арных отношений на E_k , которые тотально симметричны, тотально рефлексивны и отличны от ι_h и π_h .

Определение. Отношение $\rho \subseteq E_k^h$ называется центральным, если $\rho \in \mathbb{T}_h$ и $\exists C \subset E_k$ ($C \neq \emptyset$ & $\forall a_0 \in E_k, \dots, \forall a_{h-2} \in E_k \forall u \in C (a_0, \dots, a_{h-2}, u) \in \rho$). Множество C называется центром отношения ρ . Каждый элемент из C называется центральным.

Через Z_h обозначим множество всех h -арных центральных отношений, $2 \leq h \leq k$. Пусть $Z = \bigcup_{h=2}^{k-1} Z_h$. Пусть A_ρ есть класс всех функций, сохраняющих отношение ρ .

Теорема. Если $\rho \in Z$, то A_ρ есть предполный класс.

Пусть $\mathbb{C} = \{A_\rho : \rho \in Z\}$.

2.7. Классы типа $\mathbb{B}$

Пусть $h \geq 2, s \geq 1, a \in E_{h^s}$ и $[a^{(0)}, a^{(1)}, \dots, a^{(s-1)}]$ есть h -адическая запись (развертка) числа a в системе счисления по основанию h (здесь $0 \leq a^{(i)} < h, i = 0, 1, \dots, s-1$, и $a^{(0)} + a^{(1)}h + \dots + a^{(s-2)}h^{s-2} + a^{(s-1)}h^{s-1} = a$). Всякий набор $(a^{(0)}, a^{(1)}, \dots, a^{(s-1)})$ из E_h^s можно рассматривать как развертку соответствующего числа a из E_{h^s} , и наоборот. Таким образом, между элементами из E_{h^s} и наборами из E_h^s существует взаимно-однозначное соответствие.

Определение. Определим h -адическое элементарное отношение ζ_s следующим образом. Набор $(a_0, a_1, \dots, a_{h-1})$ из E_{h^s} лежит в ζ_s , если в h -адических записях

$$\begin{aligned} a_0 &= [a_0^{(0)}, a_0^{(1)}, \dots, a_0^{(j)}, \dots, a_0^{(s-1)}], \\ &\dots \\ a_l &= [a_l^{(0)}, a_l^{(1)}, \dots, a_l^{(j)}, \dots, a_l^{(s-1)}], \\ &\dots \\ a_{h-1} &= [a_{h-1}^{(0)}, a_{h-1}^{(1)}, \dots, a_{h-1}^{(j)}, \dots, a_{h-1}^{(s-1)}], \end{aligned}$$

чисел a_l из $E_{h^s}, l = 0, 1, \dots, h-1$ (в системе счисления по основанию h) каждый набор-столбец $(a_0^{(j)}, a_1^{(j)}, \dots, a_{h-1}^{(j)})$ из E_h^h , где $a_l^{(j)}$ есть j -я цифра в h -адической записи числа $a_l, j = 0, 1, \dots, s-1$, лежит в E_h (т. е. неразнозначен).

Определение. h -арное отношение ρ на E_k называется сильногомоморфным прообразом h -арного отношения σ на E_l при отображении (сильном гомоморфизме)

$$\begin{aligned} p: E_k &\rightarrow E_l, \text{ если } \forall a_0 \in E_k \forall a_1 \in E_k \dots \forall a_{h-1} \in E_k \\ (p(a_0), p(a_1), \dots, p(a_{h-1})) &\in \sigma \leftrightarrow (a_0, a_1, \dots, a_{h-1}) \in \rho. \end{aligned}$$

h -арное отношение ρ на E_k называется сильногомоморфным прообразом h -арного отношения σ на E_l , если существует сильный гомоморфизм $p: E_k \rightarrow E_l$, при котором ρ есть прообраз отношения σ .

Теорема. Если 1) $3 \leq h \leq k$, 2) h -арное отношение ρ на E_k есть сильногомоморфный прообраз h -адического элементарного отношения ζ_s на E_{h^s} при отображении $p: E_k \rightarrow E_{h^s}$, то A_ρ является предполным классом.

Замечание. Итак, описаны все шесть семейств предполных классов в k -значной логике: 1) классы $\mathbb{S}$ самодвойственных функций; 2) классы $\mathbb{M}$ монотонных функций; 3) классы $\mathbb{L}$ линейных функций; 4) классы $\mathbb{E}$ функций, сохраняющих разбиения множества E_k ; 5) классы типа $\mathbb{C}$; 6) классы типа $\mathbb{B}$.

Справедлива следующая теорема.

Теорема (Розенберга о функциональной полноте). Система F функций k -значной логики полна (в P_k), если и только если она не содержится целиком ни в одном из предполных классов шести перечисленных семейств.

2.8. Сравнение функций двузначной и многозначной логик

Отметим сходство и различие двузначных и многозначных логик.

1. P_2 и P_k ($k \geq 3$) имеют конечное число предполных классов.
2. Множество замкнутых классов в P_2 счетно (взаимно-однозначно со множеством натуральных чисел), и каждый замкнутый класс имеет конечный базис. Множество замкнутых классов в P_k ($k \geq 3$) имеет мощность континуума (взаимно-однозначно со множеством всех вещественных чисел). В P_k ($k \geq 3$) существуют замкнутые классы, не имеющие конечного базиса.
3. Число предполных классов в P_2 равно пяти. Число предполных классов в P_k ($k \geq 3$) растет с ростом k , но число типов предполных классов конечно и равно шести. Число предполных классов в каждом типе предполных классов растет с ростом k .



Производные булевой функции в синтезе логических схем



3.1. Производная булевой функции

Пусть $f(x_1, x_2, \dots, x_n)$ есть функция алгебры логики.

Определение. Производная первого порядка $\frac{\partial f}{\partial x_i}$ (или f'_{x_i}) от функции f по переменной x_i есть функция

$$\frac{\partial f}{\partial x_i} = f'_{x_i} = f(x_1, x_2, \dots, x_{i-1}, x, x_{i+1}, \dots, x_n) + f(x_1, x_2, \dots, x_{i-1}, \bar{x}, x_{i+1}, \dots, x_n) = f(x_1, x_2, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n) + f(x_1, x_2, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n),$$

где $f(x_1, x_2, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n)$ есть единичная остаточная функция;

$f(x_1, x_2, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n)$ есть нулевая остаточная функция;

$+$ есть знак суммы по модулю два.

Замечание. Производная $\frac{\partial f}{\partial x_i}$ (или f'_{x_i}) от функции $f(x_1, \dots, x_n)$ определяет условие, при котором функция f изменяет значение при изменении значения переменной x_i на противоположное.

Пример. $f(x_1, x_2, x_3) = \bar{x}_2 \bar{x}_3 \vee x_1 x_2 x_3$. Производная

$$\frac{\partial f}{\partial x_1} = (\bar{x}_2 \bar{x}_3 \vee x_2 x_3) + \bar{x}_2 \bar{x}_3 = (x_2 \bar{x}_3 \vee x_2 x_3) \cdot \neg(\bar{x}_2 \bar{x}_3) \vee \neg(\bar{x}_2 \bar{x}_3 \vee x_2 x_3) \bar{x}_2 \bar{x}_3 = (\bar{x}_2 \bar{x}_3 \vee x_2 x_3)(x_2 \vee x_3) \vee (x_2 \vee x_3)(\bar{x}_2 \vee \bar{x}_3) \bar{x}_2 \bar{x}_3 = x_2 x_3.$$

Функция $f(x_1, x_2, x_3)$ изменяет свое значение s на $\neg s$ при изменении значения переменной x_1 при условии, что конъюнкция $x_2 x_3$ принимает значение 1, т. е. $x_2 = 1$ и $x_3 = 1$.

Определение. Смешанная производная $\frac{\partial^k f}{\partial x_{i_1} \partial x_{i_2} \dots \partial x_{i_k}}$ (или $f_{x_{i_1} x_{i_2} \dots x_{i_k}}^{(k)}$) порядка k от функции $f(x_1, x_2, \dots, x_n)$ определяется по индукции следующим образом.

$$1. \frac{\partial f}{\partial x_{i_1}} = f(x_1, x_2, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n) + f(x_1, x_2, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n).$$

$$2. \frac{\partial^k f}{\partial x_{i_1} \partial x_{i_2} \dots \partial x_{i_k}} = \frac{\partial}{\partial x_{i_k}} \left(\frac{\partial^{k-1} f}{\partial x_{i_1} \partial x_{i_2} \dots \partial x_{i_{k-1}}} \right).$$

Пример.

$$f(x, y, z) = \bar{x}yz \vee xy\bar{z} \vee x\bar{y}z = (x+1)yz + x(y+1)(z+1) + xy(z+1) = \\ (xyz + yz) + (xyz + xy + xz + x) + (xyz + xy) = x + xz + yz + xyz.$$

$$f'_x(y, z) = f(0, y, z) + f(1, y, z) = (0 + 0z + yz + 0yz) + (1 + 1z + yz + 1yz) = 1 + z + yz.$$

$$f'_y(x, z) = f(x, 0, z) + f(x, 1, z) = (x + xz + 0z + x0z) + (x + xz + 1z + x1z) = \\ (x + xz) + (x + xz + z + xz) = z + xz.$$

$$f'_z(x, y) = f(x, y, 0) + f(x, y, 1) = (x + x0 + y0 + xy0) + (x + x1 + y1 + xy1) = \\ (x) + (x + x + y + xy) = x + y + xy.$$

$$f''_{xy} = f'_x(f'_y(x, z)) = f'_x(z + xz) = (z + 0z) + (z + 1z) = z.$$

$$f''_{xz} = f'_x(f'_z(x, y)) = f'_x(x + y + xy) = (0 + y + 0y) + (1 + y + 1y) = (y) + (1) = 1 + y.$$

$$f''_{yz} = f'_y(f'_z(x, y)) = f'_y(x + y + xy) = (x + 0 + x0) + (x + 1 + x1) = (x) + (1) = 1 + x.$$

$$f'''_{xyz} = f'_x(f''_{yz}(x)) = f'_x(1 + x) = (1 + 0) + (1 + 1) = 1.$$

Значения производных приведены в табл. 3.1.

Таблица 3.1

n	xyz	f	f'_x	f'_y	f'_z	f''_{xy}	f''_{xz}	f''_{yz}	f'''_{xyz}
0	000	0	1	0	0	0	1	1	1
1	001	0	0	1	0	1	1	1	1
2	010	0	1	0	1	0	0	1	1
3	011	1	1	1	1	1	0	1	1
4	100	1	1	0	1	0	1	0	1
5	101	0	0	0	1	1	1	0	1
6	110	1	1	0	1	0	0	0	1
7	111	0	1	0	1	1	0	0	1

Пусть C_S^r есть все сочетания элементов множества $S = \{i_1, i_2, \dots, i_k\}$ по r элементов в каждом сочетании.

Определение. Производная (Бохмана) $\frac{\partial f^{(k)}}{\partial (x_{i_1}, x_{i_2}, \dots, x_{i_k})}$ (или $f_{(x_{i_1}, x_{i_2}, \dots, x_{i_k})}^{(k)}$) по направлению от функции $f(x_1, x_2, \dots, x_n)$ по переменным $x_{i_1}, x_{i_2}, \dots, x_{i_k}$ равна сумме по модулю два всех смешанных производных первого, второго, ..., k -го порядка при фиксации индексов переменных во множестве S :

$$\frac{\partial^k f}{\partial(x_{i_1}, x_{i_2}, \dots, x_{i_k})} = \sum_{i \in C_S^1} \frac{\partial f}{\partial x_i} + \sum_{(i,j) \in C_S^2} \frac{\partial^2 f}{\partial x_i \partial x_j} + \sum_{(i,j,l) \in C_S^3} \frac{\partial^3 f}{\partial x_i \partial x_j \partial x_l} + \dots + \frac{\partial^k f}{\partial x_{i_1} \partial x_{i_2} \dots \partial x_{i_k}}.$$

Замечание. Производная $f_{(x_{i_1}, x_{i_2}, \dots, x_{i_k})}^{(k)}$ от функции $f(x_1, x_2, \dots, x_n)$ по переменным $x_{i_1}, x_{i_2}, \dots, x_{i_k}$ определяет условие, при котором эта функция изменяет значение при одновременном изменении значений переменных $x_{i_1}, x_{i_2}, \dots, x_{i_k}$.

Определение. Вес производной $p\left(\frac{\partial^{(k)} f}{\partial(x_{i_1}, x_{i_2}, \dots, x_{i_k})}\right)$ есть число элементарных конъюнкций в СДНФ, представляющей производную.

Пример. $f(x_1, x_2, x_3) = x_1 x_2 \vee x_1 x_3$. Определим условия изменения значения булевой функции f при изменении значений каждой переменной, первой и второй переменных одновременно и всех трех переменных одновременно.

$$\frac{\partial f}{\partial x_1} = x_2 \vee \bar{x}_3;$$

$$\frac{\partial f}{\partial x_2} = x_1 + x_1 \bar{x}_3 = x_1(1 + \bar{x}_3) = x_1 x_3;$$

$$\frac{\partial f}{\partial x_3} = x_1 x_2 + x_1 = x_1(x_2 + 1) = x_1 \bar{x}_2.$$

Условие $\frac{\partial f}{\partial x_1} = 1$ является условием изменения значения функции f при изменении значения переменной x_1 : $x_2 = 1$ или $x_2 = 0$. Значение функции f изменяется при изменении значения переменной x_2 , если $x_1 = 1$ и $x_3 = 1$. Значение функции f изменяется, когда изменяется значение переменной x_3 , при $x_1 = 1$ и $x_2 = 0$.

$$\frac{\partial^2 f}{\partial x_1 \partial x_2} = \frac{\partial}{\partial x_2} \left(\frac{\partial f}{\partial x_1} \right) = 1 + \bar{x}_3 = x_3;$$

$$\frac{\partial^2 f}{\partial(x_1, x_2)} = \frac{\partial f}{\partial x_1} + \frac{\partial f}{\partial x_2} + \frac{\partial^2 f}{\partial x_1 \partial x_2} = (x_2 \vee \bar{x}_3) + x_1 x_3 + x_3 =$$

$$(x_2 \vee \bar{x}_3) + x_3(x_1 + 1) = (x_2 \vee \bar{x}_3) + \bar{x}_1 x_3 =$$

$$(x_2 \vee \bar{x}_3) \cdot \neg(x_1 x_3) \vee (\neg(x_2 \vee \bar{x}_3)) \cdot x_1 x_3 = (x_2 \vee \bar{x}_3)(\bar{x}_1 \vee \bar{x}_3) \vee (\bar{x}_2 \vee x_3)\bar{x}_1 x_3 =$$

$$x_1 x_2 \vee x_2 \bar{x}_3 \vee \bar{x}_1 \bar{x}_3 \vee \bar{x}_3 \vee \bar{x}_1 \bar{x}_2 x_3 = x_1 x_2 \vee (\bar{x}_3 \vee \bar{x}_1 \bar{x}_2 x_3) =$$

$$x_1 x_2 \vee (\bar{x}_3 \vee \bar{x}_1 \bar{x}_2)(\bar{x}_3 \vee x_3) = \bar{x}_3 \vee x_1 x_2 \vee \bar{x}_1 \bar{x}_2 = \bar{x}_3 \vee (x_1 \equiv x_2).$$

Значение функции f изменяется при любом одновременном изменении значений переменных x_1 и x_2 , при условии $x_3 = 0$ или $x_1 = x_2$.

Найдем $\frac{\partial^3 f}{\partial(x_1, x_2, x_3)}$.

$$\frac{\partial^2 f}{\partial x_1 \partial x_2} = \frac{\partial}{\partial x_2} \left(\frac{\partial f}{\partial x_1} \right) = 1 + \bar{x}_3 = x_3;$$

$$\frac{\partial^2 f}{\partial x_1 \partial x_3} = \frac{\partial}{\partial x_3} \left(\frac{\partial f}{\partial x_1} \right) = x_2 + 1 = \bar{x}_2;$$

$$\frac{\partial^2 f}{\partial x_2 \partial x_3} = \frac{\partial}{\partial x_3} \left(\frac{\partial f}{\partial x_2} \right) = x_1;$$

$$\frac{\partial^3 f}{\partial x_1 \partial x_2 \partial x_3} = \frac{\partial}{\partial x_3} \left(\frac{\partial^2 f}{\partial x_1 \partial x_2} \right) = 1;$$

$$\frac{\partial^3 f}{\partial(x_1, x_2, \dots, x_3)} = \frac{\partial f}{\partial x_1} + \frac{\partial f}{\partial x_2} + \frac{\partial f}{\partial x_3} + \frac{\partial^2 f}{\partial x_1 \partial x_2} + \frac{\partial^2 f}{\partial x_1 \partial x_3} + \frac{\partial^2 f}{\partial x_2 \partial x_3} + \frac{\partial^3 f}{\partial x_1 \partial x_2 \partial x_3} =$$

$$(x_2 \vee \bar{x}_3) + x_1 \bar{x}_3 + x_1 \bar{x}_2 + x_3 + \bar{x}_2 + x_1 + 1 =$$

$$(x_2 \vee \bar{x}_3) + \bar{x}_3(x_1 + 1) + \bar{x}_2(x_1 + 1) + \bar{x}_1 =$$

$$(x_2 \vee \bar{x}_3) + \bar{x}_1 \bar{x}_3 + \bar{x}_1(\bar{x}_2 + 1) = (x_2 \vee \bar{x}_3) + \bar{x}_1 \bar{x}_3 + \bar{x}_1 \bar{x}_2 =$$

$$(x_2 \vee \bar{x}_3) \cdot \neg(\bar{x}_1 \bar{x}_3) \vee (\neg(x_2 \vee \bar{x}_3))(\bar{x}_1 \bar{x}_3) + \bar{x}_1 \bar{x}_2 =$$

$$(x_2 \vee \bar{x}_3)(x_1 \vee \bar{x}_3) \vee (\bar{x}_2 \bar{x}_3) \bar{x}_1 \bar{x}_3 + \bar{x}_1 \bar{x}_2 = (x_2 \vee \bar{x}_3)(x_1 \vee \bar{x}_3) \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 + \bar{x}_1 \bar{x}_2 =$$

$$x_1 \bar{x}_2 \vee \bar{x}_2 \bar{x}_3 \vee \bar{x}_1 \bar{x}_3 \vee \bar{x}_3 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 + \bar{x}_1 \bar{x}_2 = x_1 \bar{x}_2 \vee (\bar{x}_3 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3) + \bar{x}_1 \bar{x}_2 =$$

$$x_1 \bar{x}_2 \vee (\bar{x}_3 \vee \bar{x}_1 \bar{x}_2)(\bar{x}_3 \vee \bar{x}_3) + \bar{x}_1 \bar{x}_2 = x_1 \bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_1 \bar{x}_2 + \bar{x}_1 \bar{x}_2 =$$

$$(\bar{x}_3 \vee x_1 \bar{x}_2 \vee \bar{x}_1 \bar{x}_2) \cdot \neg(\bar{x}_1 \bar{x}_2) \vee (\neg(\bar{x}_3 \vee \bar{x}_1 \bar{x}_2 \vee \bar{x}_1 \bar{x}_2)) \bar{x}_1 \bar{x}_2 =$$

$$(\bar{x}_3 \vee x_1 \bar{x}_2 \vee \bar{x}_1 \bar{x}_2)(x_1 \vee \bar{x}_2) \vee \bar{x}_3 \cdot (x_1 \vee \bar{x}_2) + (\bar{x}_1 \vee \bar{x}_2) \cdot \bar{x}_1 \bar{x}_2 =$$

$$x_1 \bar{x}_3 \vee x_1 \bar{x}_2 \vee \bar{x}_2 \bar{x}_3 \vee \bar{x}_1 \bar{x}_2 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 =$$

$$x_1 \bar{x}_2 \bar{x}_3 \vee x_1 \bar{x}_2 \bar{x}_3 \vee x_1 \bar{x}_2 \bar{x}_3 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 =$$

$$x_1 \bar{x}_2 \bar{x}_3 \vee x_1 \bar{x}_2 \bar{x}_3 \vee x_1 \bar{x}_2 \bar{x}_3 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3.$$

При изменении двоичных векторов (x_1, x_2, x_3) : $(0,0,0) \leftrightarrow (1,1,1)$, $(0,0,1) \leftrightarrow (1,1,0)$, $(0,1,1) \leftrightarrow (1,0,0)$ – значение функции f изменяется.

$$\text{Вес производной Бохмана } p \left(\frac{\partial^{(3)} f}{\partial(x_1, x_2, x_3)} \right) = 6.$$

3.2. Синтез логических схем методом каскадов

Рассмотрим ряд методов проектирования схем, эффективность каждого из которых увеличивается по сравнению с предыдущими.

Метод каскадов, основанный на представлении Шеннона

$$f(x_1, \dots, x_k, x_{k+1}, \dots, x_n) = \bigvee_{(c_1, \dots, c_k) \in E_2^k} \bigwedge_{i=1}^k x_i^{c_i} f(c_1, \dots, c_k, x_{k+1}, \dots, x_n),$$

позволяет при наличии блоков исключения k переменных свести реализацию булевой функции от n переменных к реализации функции от $n - k$, $k \geq 1$, переменных. Размерность остаточных функций $f(c_1, \dots, c_k, x_{k+1}, \dots, x_n)$, в свою очередь, можно понизить, исключая t переменных и т. д. до тех пор, пока остаточные функции не будут иметь простой вид, и их реализация в заданном базисе не представляет трудности.

Сложность остаточных функций зависит от порядка исключения переменных в заданной булевой функции $f(x_1, x_2, \dots, x_n)$. Число всевозможных способов исключения переменных растет комбинаторно. При использовании только блоков, исключающих одну и ту же переменную на каждом ярусе, это число равно $n!$, но на каждом ярусе можно исключать не только одну и ту же переменную, но и различные переменные. Далее, можно исключать на каждом шаге различное число переменных (одну, две, три и т. д.). Выбор оптимального исключения переменных перебором всех способов исключения – трудоемкий процесс.

Оптимальное исключение переменных ищут, используя эвристические критерии, один из которых основан на применении понятия производной булевой функции.

Критерий оптимального исключения переменных в методе каскадов заключается в исключении сначала переменных, при переключении которых булева функция переключается при максимальном числе условий. Это максимальное число определяется весом производной.

При использовании блоков, исключающих k переменных, находят производные k -го порядка от реализуемой функции и ищут максимальное

значение веса производной Бохмана $p\left(\frac{\partial^{(k)} f}{\partial(x_{i_1}, x_{i_2}, \dots, x_{i_k})}\right)$, которое определя-

ет исключаемые переменные. Для получения остаточных булевых функций снова находятся производные, определяются их веса, а производная от рассматриваемой остаточной функции, имеющая максимальный вес, определяет соответствующие переменные, которые исключаются на этом ярусе для этой остаточной функции, и т. д., пока остаточные функции не будут иметь простую реализацию.

Пример. Синтезируем логическую схему, реализующую функцию

$$f(x_1, x_2, x_3, x_4, x_5) = x_1 \bar{x}_2 x_3 \vee \bar{x}_1 \bar{x}_3 x_4 \vee x_1 x_3 \bar{x}_5 \vee x_1 x_2 x_4 \vee \bar{x}_2 x_3 x_5 \vee \bar{x}_3 \bar{x}_4 x_5,$$

используя блоки исключения одной переменной (рис. 3.1а).

Определим переменную x_i , по которой производная $\frac{\partial f}{\partial x_i}$ имеет максимальный вес, т. е. функция $f(x_1, x_2, x_3, x_4, x_5)$ зависит от нее наиболее существенно. Имеем: $\frac{\partial f}{\partial x_1} = (\bar{x}_2 x_3 \vee \bar{x}_3 \bar{x}_5 \vee x_2 x_4 \vee \bar{x}_2 x_3 x_5 \vee \bar{x}_3 \bar{x}_4 x_5) + (\bar{x}_3 x_4 \vee \bar{x}_2 x_3 x_5 \vee \bar{x}_3 \bar{x}_4 x_5)$.

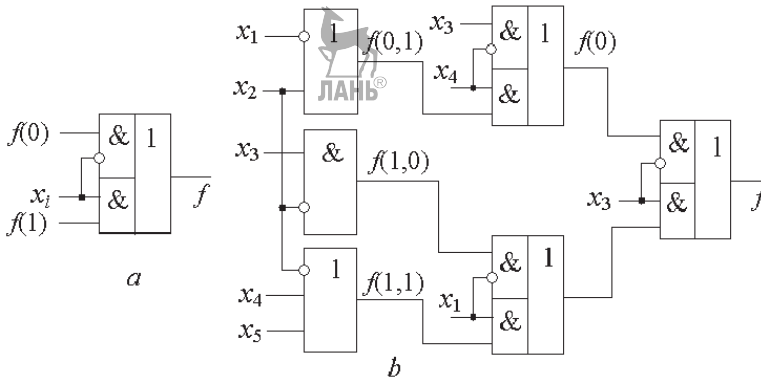


Рис. 3.1

Для вычисления веса производной $\frac{\partial f}{\partial x_1}$, зависящей от четырех переменных x_2, x_3, x_4, x_5 , представим 4-мерное пространство с образующими $\{x_2, x_3, x_4, x_5\}$ в виде декартова произведения двух 2-мерных пространств $\{x_2, x_3\} \times \{x_4, x_5\}$ с образующими $\{x_2, x_3\}$ и $\{x_4, x_5\}$ соответственно. Тогда производную $\frac{\partial f}{\partial x_1}$ можно задать в виде двумерной таблицы. Каждому значению c_2, c_3 переменных x_2, x_3 взаимно-однозначно соответствует строка таблицы, столбцу – значения c_4, c_5 переменных x_4, x_5 , и на пересечении i -й строки и j -го столбца, взаимно-однозначно соответствующем точке 4-мерного пространства с образующими $\{x_2, x_3, x_4, x_5\}$, записываем значение $\frac{\partial f}{\partial x_1}$ в этой точке. Вес производной $\frac{\partial f}{\partial x_1}$ равен числу единиц в этой таблице (табл. 3.2a): каждому значению c_2, c_3 переменных x_2, x_3 взаимно-однозначно соответствует строка таблицы, столбцу – значения c_4, c_5 переменных x_4, x_5 , и на пересечении i -й строки и j -го столбца, взаимно-однозначно соответствующем точке 4-мерного пространства с образующими $\{x_2, x_3, x_4, x_5\}$, записываем значение $\frac{\partial f}{\partial x_1}$ в этой точке. Вес производной $\frac{\partial f}{\partial x_1}$ равен числу единиц в табл. 3.2a.

Таблица 3.2a

$x_2 \ x_3$	$x_4 \ x_5$	$x_4 \ x_5$	$x_4 \ x_5$	$x_4 \ x_5$
	0	1	2	3
0	0	0	1	1
1	1	0	1	0
2	0	0	0	0
3	1	0	1	1

Вес $p\left(\frac{\partial f}{\partial x_1}\right) = 7$. Аналогично вычислим веса производных $\frac{\partial f}{\partial x_i}$, $i = 2, 3, 4, 5$ (табл. 3.2б, в, з, д).

$$\frac{\partial f}{\partial x_2} = (\bar{x}_1 \bar{x}_3 x_4 \vee x_1 x_3 \bar{x}_5 \vee x_1 x_4 \vee \bar{x}_3 \bar{x}_4 x_5) + (x_1 x_3 \vee \bar{x}_1 \bar{x}_3 x_4 \vee x_1 x_3 \bar{x}_5 \vee x_3 x_5 \vee \bar{x}_3 \bar{x}_4 x_5).$$

Таблица 3.2б

$x_1 x_3$	$x_4 x_5$	$x_4 x_5$	$x_4 x_5$	$x_4 x_5$
	0	1	2	3
0	0	0	0	0
1	0	1	0	1
2	0	0	1	1
3	0	1	0	0

Вес $p\left(\frac{\partial f}{\partial x_2}\right) = 5$.

$$\frac{\partial f}{\partial x_3} = (x_1 \bar{x}_5 \vee x_1 x_2 x_4 \vee \bar{x}_2 x_5) + (\bar{x}_1 x_4 \vee x_1 x_2 x_4 \vee \bar{x}_4 x_5).$$

Таблица 3.2в

$x_1 x_2$	$x_4 x_5$	$x_4 x_5$	$x_4 x_5$	$x_4 x_5$
	0	1	2	3
0	0	0	1	0
1	0	1	1	1
2	1	0	1	1
3	0	1	0	0

Вес $p\left(\frac{\partial f}{\partial x_3}\right) = 8$.

$$\frac{\partial f}{\partial x_4} = (x_1 \bar{x}_2 x_3 \vee \bar{x}_1 \bar{x}_3 \vee x_1 x_3 \bar{x}_5 \vee x_1 x_2 \vee \bar{x}_2 x_3 x_5) + (x_1 \bar{x}_2 x_3 \vee x_1 x_3 \bar{x}_5 \vee \bar{x}_2 x_3 x_5 \vee \bar{x}_3 x_5).$$

Таблица 3.2г

$x_1 x_2$	$x_3 x_5$	$x_3 x_5$	$x_3 x_5$	$x_3 x_5$
	0	1	2	3
0	1	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	0	0	1

$$\text{Вес } p\left(\frac{\partial f}{\partial x_4}\right) = 5.$$

$$\frac{\partial f}{\partial x_5} = (x_1 \bar{x}_2 \bar{x}_3 \vee \bar{x}_1 \bar{x}_3 x_4 \vee x_1 x_2 x_4 \vee \bar{x}_2 x_3 \vee x_3 \bar{x}_4) + (x_1 \bar{x}_2 x_3 \vee \bar{x}_1 \bar{x}_3 x_4 \vee x_1 x_3 \vee x_1 x_2 x_4).$$

Таблица 3.2б

$x_1 x_2$	$x_3 x_4$	$x_3 x_4$	$x_3 x_4$	$x_3 x_4$
	0	1	2	3
0	1	0	1	1
1	1	0	0	0
2	1	0	0	0
3	1	0	1	0

$$\text{Вес } p\left(\frac{\partial f}{\partial x_5}\right) = 7.$$

Максимальное значение $\max_i p\left(\frac{\partial f}{\partial x_i}\right)$ получено при дифференцировании функции f по переменной x_3 . Исключая эту переменную, получаем две остаточные функции: единичную $f(x_1, x_2, x_3=1, x_4, x_5) = f(1)$ и нулевую $f(x_1, x_2, x_3=0, x_4, x) = f(0)$:

$$f(1) = x_1 \bar{x}_2 \vee x_1 x_5 \vee x_1 x_2 x_4 \vee \bar{x}_2 x_5; f(0) = x_1 x_4 \vee x_1 x_2 x_4 \vee x_4 x_5.$$

Аналогично определяем оптимальное исключение переменных на следующем ярусе логической схемы (табл. 3.3а, б, в, г):

$$\frac{\partial f(1)}{\partial x_1} = (\bar{x}_2 \vee x_5 \vee x_2 x_4 \vee \bar{x}_2 x_5) + \bar{x}_2 x_5.$$

Таблица 3.3а

x_2	$x_4 x_5$	$x_4 x_5$	$x_4 x_5$	$x_4 x_5$
	0	1	2	3
0	1	0	1	0
1	1	0	1	1

$$\text{Вес } p\left(\frac{\partial f(1)}{\partial x_1}\right) = 5.$$

$$\frac{\partial f(1)}{\partial x_2} = (x_1 x_5 \vee x_1 x_4) + (x_1 \vee x_1 x_5 \vee x_5).$$

Таблица 3.3б

x_1	$x_4 x_5$	$x_4 x_5$	$x_4 x_5$	$x_4 x_5$
	0	1	2	3
0	0	1	0	1
1	0	1	0	0

$$\text{Вес } p\left(\frac{\partial f(1)}{\partial x_2}\right) = 3.$$

$$\frac{\partial f(1)}{\partial x_4} = (x_1 \bar{x}_2 \vee x_1 x_5 \vee x_1 x_2 x_4 \vee \bar{x}_2 x_5) + (x_1 \bar{x}_2 \vee x_1 x_5 \vee \bar{x}_2 x_5).$$

Таблица 3.3в

x_1	$x_2 x_5$	$x_2 x_5$	$x_2 x_5$	$x_2 x_5$
	0	1	2	3
0	0	0	0	0
1	0	0	0	1

$$\text{Вес } p\left(\frac{\partial f(1)}{\partial x_4}\right) = 1.$$

$$\frac{\partial f(1)}{\partial x_5} = (x_1 \bar{x}_2 \vee x_1 x_2 x_4 \vee \bar{x}_2) + (x_1 \bar{x}_2 \vee x_1 \vee x_1 x_2 x_4).$$

Таблица 3.4г

x_1	$x_2 x_4$	$x_2 x_4$	$x_2 x_4$	$x_2 x_4$
	0	1	2	3
0	1	1	0	0
1	0	0	1	0

$$\text{Вес } p\left(\frac{\partial f(1)}{\partial x_5}\right) = 3.$$

Исключаем переменную x_1 и получаем остаточные функции вида:

$$f(x_1 = 1, x_2, x_3 = 1, x_4, x_5) = f(1, 1) = \bar{x}_2 \vee x_5 \vee x_2 x_4 \vee \bar{x}_2 x_5 = \bar{x}_2 \vee \bar{x}_5 \vee x_4,$$

$$f(x_1 = 0, x_2, x_3 = 1, x_4, x_5) = f(1, 0) = x_2 x_5,$$

$$f(0) = x_1 x_4 \vee x_1 x_2 x_4 \vee x_4 x_5 = x_1 x_4 \vee x_2 x_4 \vee x_4 x_5,$$

$$\frac{\partial f(0)}{\partial x_1} = (x_2 x_4 \vee \bar{x}_4 x_5) + (x_4 \vee x_2 x_4 \vee \bar{x}_4 x_5).$$

Таблица 3.4а

x_2	$x_4 x_5$	$x_4 x_5$	$x_4 x_5$	$x_4 x_5$
	0	1	2	3
0	0	0	1	1
1	0	0	0	0



$$\text{Вес } p\left(\frac{\partial f(0)}{\partial x_1}\right) = 2.$$

$$\frac{\partial f(0)}{\partial x_2} = (\bar{x}_1 x_4 \vee x_4 \vee \bar{x}_4 x_5) + (\bar{x}_1 x_4 \vee \bar{x}_4 x_5).$$

Таблица 3.4б

x_1	$x_4 x_5$	$x_4 x_5$	$x_4 x_5$	$x_4 x_5$
	0	1	2	3
0	0	0	0	0
1	0	0	1	1

$$\text{Вес } p\left(\frac{\partial f(0)}{\partial x_2}\right) = 2.$$

$$\frac{\partial f(0)}{\partial x_4} = (\bar{x}_1 \vee x_2) + x_5.$$

Таблица 3.4в

x_1	$x_2 x_5$	$x_2 x_5$	$x_2 x_5$	$x_2 x_5$
	0	1	2	3
0	1	0	1	0
1	0	1	1	0

$$\text{Вес } p\left(\frac{\partial f(0)}{\partial x_4}\right) = 3.$$

$$\frac{\partial f(0)}{\partial x_5} = (\bar{x}_1 x_4 \vee x_2 x_4 \vee \bar{x}_4) + (\bar{x}_1 x_4 \vee x_2 x_4).$$



Таблица 3.4г

x_1	$x_2 x_4$	$x_2 x_4$	$x_2 x_4$	$x_2 x_4$
	0	1	2	3
0	1	0	1	0
1	1	0	1	0

$$\text{Вес } p\left(\frac{\partial f(0)}{\partial x_5}\right) = 3.$$



Исключая переменную x_4 , получаем остаточные функции вида:

$$\begin{aligned} f(x_1, x_2, x_3 = 0, x_4 = 1, x_5) &= f(0, 1) = \bar{x}_1 \vee x_2, \\ f(x_1, x_2, x_3 = 0, x_4 = 0, x_5) &= f(0, 0) = x_5. \end{aligned}$$

В результате получаем логическую схему, реализующую функцию $f(x_1, x_2, \dots, x_5)$ (рис. 3.1b).

Критерий оптимального исключения переменных имеет эвристический характер, что основано на предположении, что чем больше вес производной $p\left(\frac{\partial f}{\partial x_i}\right)$, тем больше функция f зависит от переменной x_i . Если имеются блоки исключения k -переменных, то построение схемы проводят аналогично, вычисляя вес производных k -го порядка $p\left(\frac{\partial^k f}{\partial(x_{i_1}, x_{i_2}, \dots, x_{i_k})}\right)$.

3.3. Разложение булевой функции в ряд

В дискретной математике отсутствует понятие предела, однако термин «производная» используется. Это связано с разложением булевой функции в ряд, аналогичный ряду Маклорена в точке $(0, 0, \dots, 0)$ или ряду Тейлора в произвольной точке пространства. Рассмотрим данные разложения булевой функции $f(x_1, x_2, \dots, x_n)$.

Определение. Булевы функции f_a и f_b ортогональны, если их конъюнкция $f_a \cdot f_b = 0$. $\square$



Представим функцию f в виде СДНФ $f(x_1, x_2, \dots, x_n) = \bigvee_{f(c_1, \dots, c_k)=1} \big\& x_i^{c_i}$. Всякие два разных конъюнкта функции f в СДНФ ортогональны.

Замечание. В силу принципа двойственности можно считать булевы функции f_a и f_b ортогональными, если их дизъюнкция $f_a \vee f_b = 1$.

Представим функцию f в виде СКНФ $f(x_1, x_2, \dots, x_n) = \big\&_{f(c_1, \dots, c_n)=0} \bigvee x_i^{c_i}$. Всякие два разных дизъюнкта функции f в СКНФ ортогональны.

Представим функцию f в виде полинома Жегалкина

$$f(x_1, \dots, x_n) = \sum_{(j_1, \dots, j_n) \in E_2^n} a_{j_1 j_2 \dots j_n} x_1^{j_1} x_2^{j_2} \dots x_n^{j_n}.$$

Пусть $a_{i_1 i_2 \dots i_k}$ есть коэффициент $a_{j_1 \dots j_1 \dots j_2 \dots j_2 \dots j_k \dots j_n}$ при $x_1^{j_1} x_2^{j_2} \dots x_n^{j_n}$ степени k , то есть коэффициент, в котором $j_1, j_2, \dots, j_k$ есть все k его единичных индексов. Таким образом, $a_{j_1 j_2 \dots j_n} x_1^{j_1} x_2^{j_2} \dots x_n^{j_n} = f_{i_1 i_2 \dots i_k} x_{i_1} x_{i_2} \dots x_{i_k}$.

Сгруппируем слагаемые полинома по степеням слагаемых и получим следующее представление полинома Жегалкина.

$$f(x_1, x_2, \dots, x_n) = f_0 + \sum_{i \in C_S^1} f_i x_i + \sum_{(i,j) \in C_S^2} f_{ij} x_i x_j + \dots + \sum_{(i_1, i_2, \dots, i_k) \in C_S^k} f_{i_1 i_2 \dots i_k} x_{i_1} x_{i_2} \dots x_{i_k} + \dots + f_{12 \dots n} x_1 x_2 \dots x_n, \quad (3.1)$$

где $S = \{1, 2, \dots, n\}$.

Последовательно продифференцируем полином Жегалкина функции $f(x_1, x_2, \dots, x_n)$ по переменным $x_1, x_2, \dots, x_k$ и определим значение этой производной в точке $(0, 0, \dots, 0)$. Учитывая, что

$$\frac{\partial(f+g)}{\partial x} = \frac{\partial f}{\partial x} + \frac{\partial g}{\partial x} \quad \text{и} \quad \frac{\partial}{\partial x_i} \left(\bigotimes_{j=1}^n x_j \right) = \bigotimes_{j=1, j \neq i}^n x_j,$$

после дифференцирования по переменным $x_1, x_2, \dots, x_k$ получаем, что

$$\frac{\partial^k f(0, \dots, 0)}{\partial x_1 \partial x_2 \dots \partial x_k} = f_{1,2,\dots,k}.$$

Действительно, после дифференцирования по переменным $x_1, x_2, \dots, x_k$ все члены разложения в выражении (3.1) до $f_{12 \dots k}$ обращаются в нуль, а в результате подстановки $x_{k+1} = x_{k+2} = \dots = x_n = 0$ остальные члены этого разложения, кроме $f_{1,2,\dots,k}$, также будут равны 0. Отсюда получаем теорему о разложении любой булевой функции $f(x_1, x_2, \dots, x_n)$ в точке $(0, 0, \dots, 0)$.

Теорема. Любая булева функция $f(x_1, x_2, \dots, x_n)$ представима своим значением в точке $(0, 0, \dots, 0)$ и значениями всех производных

$$\frac{\partial f}{\partial x_i}, \frac{\partial^2 f}{\partial x_i \partial x_j}, \dots, \frac{\partial^n f}{\partial x_1 \partial x_2 \dots \partial x_n}; \quad i, j (i \neq j), \dots, = 1, 2, \dots, n$$

в этой точке в виде:

$$f(x_1, x_2, \dots, x_n) = f(0, 0, \dots, 0) + \sum_{i \in C_S^1} \frac{\partial f(0, \dots, 0)}{\partial x_i} x_i + \sum_{(i,j) \in C_S^2} \frac{\partial^2 f(0, \dots, 0)}{\partial x_i \partial x_j} x_i x_j + \dots + \sum_{(i_1, i_2, \dots, i_k) \in C_S^k} \frac{\partial^k f(0, \dots, 0)}{\partial x_{i_1} \partial x_{i_2} \dots \partial x_{i_k}} x_{i_1} x_{i_2} \dots x_{i_k} + \dots + \frac{\partial^n f(0, \dots, 0)}{\partial x_1 \partial x_2 \dots \partial x_n} x_1 x_2 \dots x_n.$$

Для получения разложения булевой функции в ряд, аналогичный ряду Тейлора в точке $(c_1, c_2, \dots, c_n)$, введем новые координаты $x'_1, x'_2, \dots, x'_n$, где $x'_i = x_i + c_i$, $i = 1, 2, \dots, n$. Тогда точка $(c_1, c_2, \dots, c_n)$ в координатах $x_1, x_2, \dots, x_n$ будет соответствовать точке $(0, 0, \dots, 0)$ в координатах $x'_1, x'_2, \dots, x'_n$. Используя разложение (1.4) булевой функции в точке $(0, 0, \dots, 0)$ в координатах $x'_1, x'_2, \dots, x'_n$ и заменяя каждую переменную x'_i на $x_i + c_i$, $i = 1, 2, \dots, n$, получаем теорему о разложении булевой функции $f(x_1, x_2, \dots, x_n)$ в точке $(c_1, c_2, \dots, c_n)$.

Теорема. Любая булева функция $f(x_1, \dots, x_n)$ определяется своим значением в точке $(c_1, \dots, c_n)$ и значениями всех своих производных

$$\frac{\partial f}{\partial x_i}, \frac{\partial^2 f}{\partial x_i \partial x_j}, \dots, \frac{\partial^n f}{\partial x_1 \partial x_2 \dots \partial x_n}; \quad i, j (i \neq j), \dots, = 1, 2, \dots, n$$

в этой точке в виде:

$$\begin{aligned} f(x_1, x_2, \dots, x_n) = & f(c_1, \dots, c_n) + \sum_{i \in C_S^1} \frac{\partial f(c_1, \dots, c_n)}{\partial x_i} (x_i + c_i) + \\ & \sum_{(i, j) \in C_S^2} \frac{\partial^2 f(c_1, \dots, c_n)}{\partial x_i \partial x_j} (x_i + c_i)(x_j + c_j) + \dots + \\ & \sum_{(i_1, i_2, \dots, i_k) \in C_S^k} \frac{\partial^k f(c_1, \dots, c_n)}{\partial x_{i_1} \partial x_{i_2} \dots \partial x_{i_k}} (x_{i_1} + c_{i_1})(x_{i_2} + c_{i_2}) \dots (x_{i_k} + c_{i_k}) + \dots + \\ & \frac{\partial^n f(c_1, \dots, c_n)}{\partial x_1 \partial x_2 \dots \partial x_n} (x_1 + c_1)(x_2 + c_2) \dots (x_n + c_n). \end{aligned}$$





Синтез схем из функциональных элементов

4.1. Схема из функциональных элементов

В современной технике среди управляющих устройств важное место занимают дискретные преобразователи (ДП, трансдюсеры) с несколькими входами и несколькими выходами (рис. 4.1).



Рис. 4.1

На входы ДП в дискретные моменты времени $0, 1, \dots, t, \dots$ поступают сигналы некоторого конечного входного алфавита X , ДП их обрабатывает и выдает на выходах сигналы некоторого конечного выходного алфавита Y . Бывают дискретные преобразователи с памятью и без памяти. ДП с (конечной) памятью реализуют конечные автоматы. ДП без памяти реализуют функции. В дальнейшем мы будем рассматривать дискретные преобразователи без памяти. Если символы входного и выходного алфавитов есть $E_k = \{0, 1, \dots, k-1\}$, то ДП реализует функции k -значной логики. В ДП с p выходами реализуются p функции k -значной логики. Если символы входного и выходного алфавитов есть $E_2 = \{0, 1\}$, то ДП реализует булевы функции. Такие ДП называются функциональными элементами. Везде далее будем говорить о булевых функциях.

Пусть имеем конечное множество $F = \{F_1, \dots, F_r\}$ из r функциональных элементов. Если входам и выходам элементов приписать некоторые перемен-

ные (рис. 4.2), то элементы реализуют r функций алгебры логики $f_i(x_1, \dots, x_{n_i})$, $i = 1, 2, \dots, r$.

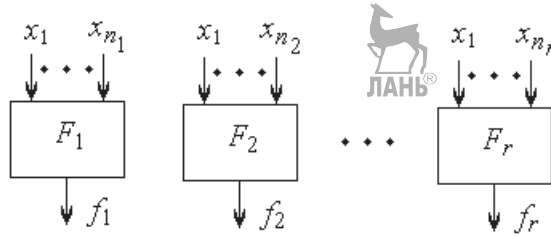


Рис. 4.2

Определение (логической сети из функциональных элементов системы F).

1. Одна изолированная вершина (одновременно вход и выход) есть сеть.
2. *Объединение сетей.* Пусть S' и S'' есть две сети без общих элементов, имеющие n и m входов и p и q выходов соответственно, тогда объединение сетей S' и S'' есть сеть S , входы которой есть $n+m$ входов сетей S' и S'' , а выходы есть $p+q$ выходов сетей S' и S'' (рис. 4.3).

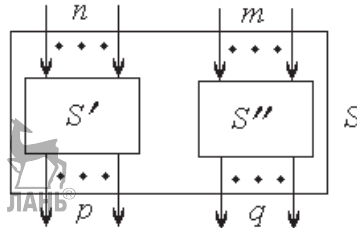


Рис. 4.3

3. *Подключение элемента к сети.* Пусть S' есть логическая сеть с n входами и p выходами, и пусть F_i есть элемент с числом входов $n_i \leq p$. Тогда результат подключения всех n_i входов элемента F_i к некоторым n_i выходам $j_1, \dots, j_{n_i}$ сети S' есть сеть S , входы которой есть входы сети S' . Выходы сети S есть выходы сети S' , кроме выходов $j_1, \dots, j_{n_i}$, а также выход элемента F_i (рис. 4.4).

4. *Расщепление выходов сети.* Пусть S' есть сеть с выходами $1, 2, \dots, j, \dots, p$. Тогда результат расщепления выхода j в сети S' есть сеть S с выходами $1, 2, \dots, j-1, j+1, \dots, p$ сети S' и еще два выхода. Входы сети S есть входы сети S' (рис. 4.5).

Определение. *Схема из функциональных элементов* есть логическая сеть, входам и выходам которой приписаны различные буквы (переменные).

Обозначение. $S(x_1, \dots, x_n; y_1, \dots, y_p)$ есть схема с входами $x_1, \dots, x_n$ и выходами $y_1, \dots, y_p$. Схема S реализует p функций $y_k = y_k(x_1, \dots, x_n)$, $k = 1, \dots, p$.

Определение. Множество $F = \{F_1, \dots, F_r\}$ из r элементов функционально полно, если для любой функции f можно построить схему, которая реализует f .

Замечание. Множество F функционально полно, если таковой является система реализуемых ими функций.

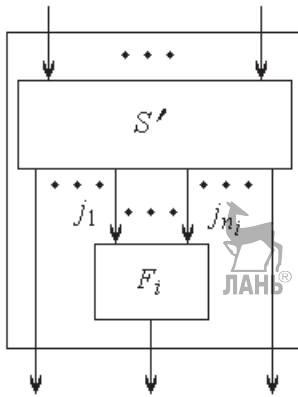


Рис. 4.4

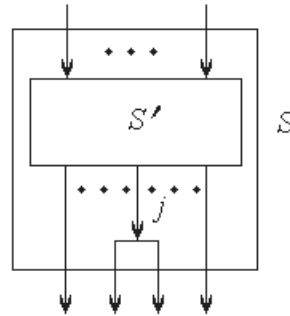


Рис. 4.5

4.2. Функции Шеннона

Пусть $F = \{F_1, \dots, F_r\}$ есть функционально полная система элементов.

Сложность $L(S)$ схемы S из функциональных элементов системы F есть число входящих в схему элементов.

Пусть $L(f)$ есть минимальная сложность схемы, реализующей функцию f , то есть $L(f) = \min L(S)$ по всем схемам S , реализующим функцию f .

Пусть $L(n)$ есть максимальная сложность минимальных реализаций всех функций n переменных, то есть $L(n) = \max L(f)$ по всем функциям из $P_2^{(n)}$.

Пусть A есть алгоритм, с помощью которого можно построить любую функцию из функционально полного набора элементов F .

Пусть $L_A(f)$ есть сложность схемы, реализующей функцию f и построенной для f с помощью алгоритма A .

Пусть $L_A(n) = \max L_A(f)$ по всем функциям из $P_2^{(n)}$.

Функции $L(n)$ и $L_A(n)$ называются функциями Шеннона.

Очевидно, что $L_A(n) \geq L(n)$.

4.3. Элементарные методы синтеза схем

Наиболее полно исследованы вопросы синтеза и сложности схем, построенных из функционально полного набора элементов, реализующих функции x & y , $x \vee y$, $\neg x$. Рассмотрим метод (алгоритм) синтеза, основанный на представлении функции с помощью СДНФ. Именно пусть

$$f(x_1, \dots, x_n) = \bigvee_{f(c_1, \dots, c_n)=1} x_1^{c_1} \dots x_n^{c_n}.$$

Пусть $K = x_1^{c_1} \dots x_n^{c_n}$. Элементарную конъюнкцию K можно реализовать схемой S_K , изображенной на рис. 4.6.

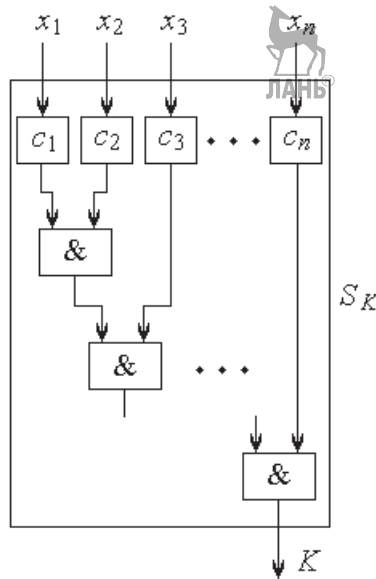


Рис. 4.6

На рис. 4.6 c_i есть $\neg$, если $c_i = 0$; c_i есть отсутствие элемента, если $c_i = 1$.

Схема S_K имеет $n - 1$ конъюнкторов $\&$ и не более n отрицаний (инверторов). Всего не более $(n - 1) + n$ элементов. Схема для элементарной конъюнкции K имеет сложность реализации $L(S_K) \leq 2n - 1$.

Все элементарные конъюнкции $K_1, \dots, K_u$ из представления функции f можно реализовать схемой G_K , изображенной на рис. 4.7.

Схема G_K имеет сложность $L(G_K) \leq u(2n - 1)$.

Далее устраиваем дизъюнкцию выходов схемы G_K , как это сделано на схеме S на рис. 4.8.

Схема S реализует функцию f . Схема S имеет $u - 1$ дизъюнкторов. Схема S имеет сложность $L(f) \leq u(2n - 1) + (u - 1) = u \cdot 2n - 1$. В худшем случае $u = 2^n$. Тогда сложность $L(n) \leq 2^n \cdot 2n - 1 = n2^{n+1} - 1$. Для описанного алгоритма А функция Шеннона $L_A(n) \leq n2^{n+1} - 1$.

Теорема (О. Б. Лупанов). Для схем из функциональных элементов для x & y ,

$$x \vee y, \neg x \text{ функция Шеннона } L(n) \leq \frac{2^n}{n} \left(1 + O\left(\frac{\log_2 n}{n}\right) \right), L(n) \geq \frac{2^n}{n}, \text{ где запись } a_n \gtrsim b_n$$

означает, что $\lim_{n \rightarrow \infty} \frac{a_n}{b_n} \geq 1$. То есть асимптотически $L(n) \asymp \frac{2^n}{n}$.

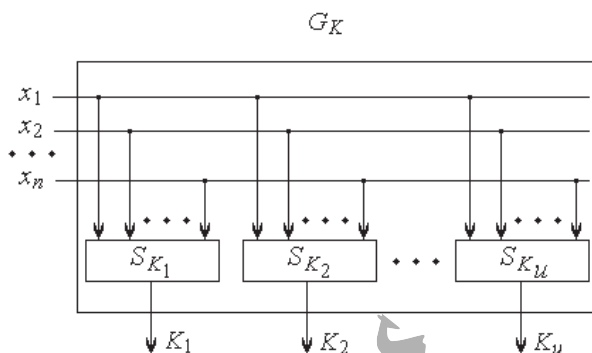


Рис. 4.7

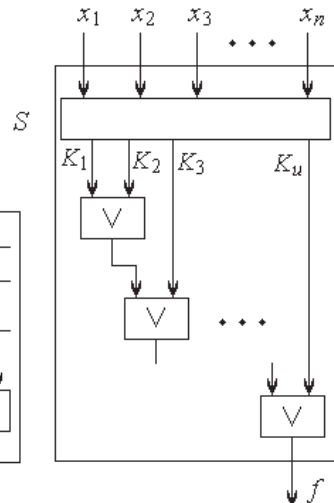


Рис. 4.8

4.4. Синтез мультиплексоров

Определение. Мультиплексор MUX есть большая интегральная схема (БИС) $M(n)$, имеющая n управляющих входов, 2^n информационных входов и один выход. Для поданного на управляющие входы набора $(c_1, \dots, c_n)$ из 0 и 1 схема делает проходным (отпирает) и пропускает сигнал единственного информационного входа, помеченного набором $(c_1, \dots, c_n)$; остальные информационные входы заперты и к выходу не проходимы (рис. 4.9).

Теорема. Мультиплексор может быть реализован схемой, содержащей $3 \cdot 2^n - 3$ конъюнкции и дизъюнкции.

Доказательство. Индукция по n .

Базис. Схема $M(1)$ изображена на рис. 4.10.

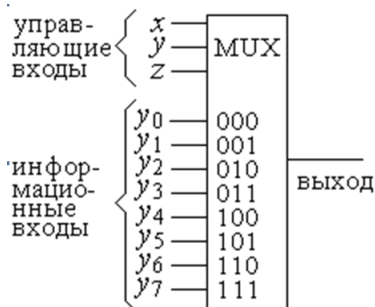


Рис. 4.9

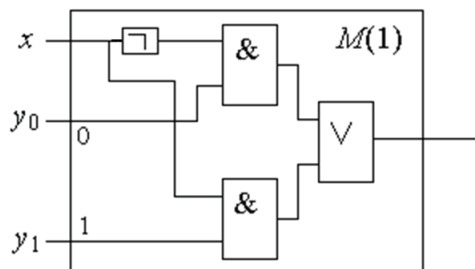


Рис. 4.10

Схема $M(1)$ имеет $3 \cdot 2^1 - 3 = 3$ конъюнкции и дизъюнкции.

Предположение индукции. Пусть мультиплексор $M(n-1)$ построен и имеет $3 \cdot 2^{n-1} - 3$ конъюнкции и дизъюнкции.

Шаг индукции. Мультиплексор $M(n)$ изображен на рис. 4.11.

Схема $M(n)$ имеет $2 \cdot (3 \cdot 2^{n-1} - 3) + 3 = 3 \cdot 2^n - 3$ конъюнкции и дизъюнкции. Шаг индукции установлен. Теорема доказана.

Утверждение. С помощью мультиплексора $M(n)$, инвертора, генератора нулей и единиц можно реализовать любую функцию алгебры логики от $n+1$ переменных.

Доказательство. По лемме Шеннона о разложении функции по компонентам $f(x_1, \dots, x_n, y) = \bigvee_{(c_1, \dots, c_n) \in E_2^n} x_1^{c_1} \dots x_n^{c_n} f(c_1, \dots, c_n, y)$.

Как функция одной переменной y функция $f(c_1, \dots, c_n, y) \in \{0, 1, y, \neg y\}$. Конъюнкция $x_1^{c_1} \dots x_n^{c_n}$ равна 1 на единственном наборе $(c_1, \dots, c_n)$. Тогда функция f может быть реализована с помощью мультиплексора так, как это изображено на рис. 4.12.

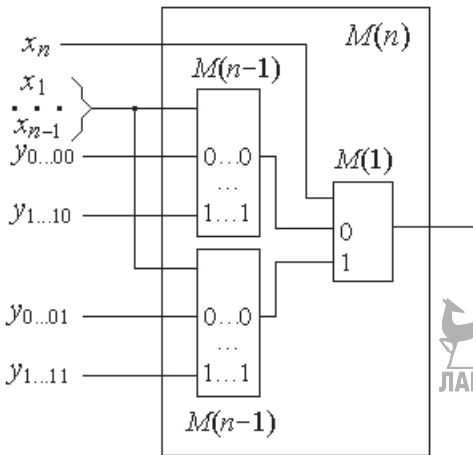


Рис. 4.11

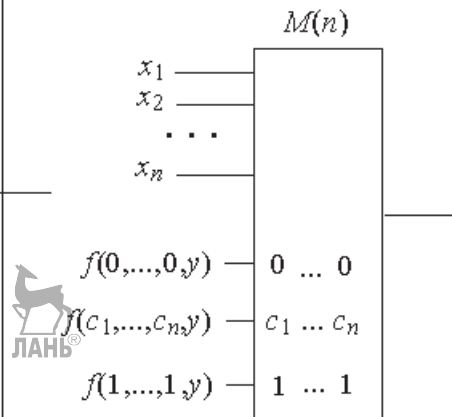


Рис. 4.12

Пример. Реализовать функцию трех переменных $f(x, y, z) = 00101101$ с помощью мультиплексора $M(2)$.

В задаче используется разложение функции по переменным x, y . Требуемая реализация приведена на рис. 4.13.

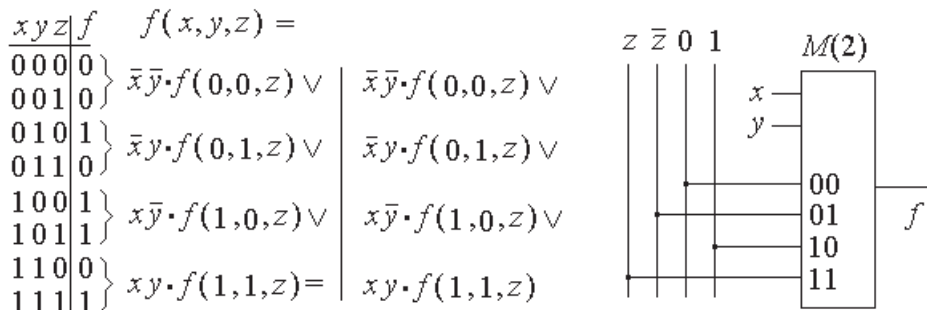


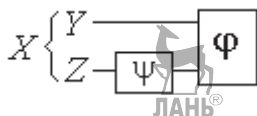
Рис. 4.13

4.5. Элементы функциональной декомпозиции

Разобьем множество $X = \{x_1, \dots, x_n\}$ из n переменных на два непересекающихся подмножества $Y = \{y_1, \dots, y_m\}$, $Z = \{z_{m+1}, \dots, z_n\}$, в сумме (в объединении) дающих все множество X .

Определение. Простая непересекающаяся декомпозиция функции $f(x_1, \dots, x_n)$ есть ее представление в виде $f(X) = \varphi(Y, \psi(Z))$ при некоторых функциях φ и ψ .

Замечание. В случае декомпозиции функция $f(X)$ может быть реализована следующей схемой, построенной из более простых функций φ и ψ .



В последующем для простоты будем считать, что $Y = \{x_1, \dots, x_m\}$, $Z = \{x_{m+1}, \dots, x_n\}$.

Определение. Y -компонента функции $f(Y,Z)$ есть совокупность функций $\{f(c_1, \dots, c_m, x_{m+1}, \dots, x_n) : (c_1, \dots, c_m) \in E_2^m\}$. Z -компонента функции $f(Y,Z)$ есть совокупность функций $\{f(x_1, \dots, x_m, c_{m+1}, \dots, c_n) : (c_{m+1}, \dots, c_n) \in E_2^{n-m}\}$.

Пример. $f(x_1, x_2, x_3, x_4)$, $Y = \{x_1, x_2\}$, $Z = \{x_3, x_4\}$ (рис. 4.14).

Столбцы таблицы на рис. 4.14 составляют Z -компоненту $\{f(0,0, x_3, x_4) = 1001, f(0,1, x_3, x_4) = 0110, f(1,1, x_3, x_4) = 1101, f(1,0, x_3, x_4) = 1011\}$ функции f .

	x_3x_4	x_3x_4	x_3x_4	x_3x_4	Функции Y -компоненты
x_1x_2	00	01	11	10	
00	1	0	1	1	$f(0,0, x_3, x_4)$
01	0	1	1	0	$f(0,1, x_3, x_4)$
11	0	1	0	1	$f(1,1, x_3, x_4)$
10	1	1	1	1	$f(1,0, x_3, x_4)$

Рис. 4.14

Теорема 1. Простая непересекающаяся декомпозиция $f(X) = \varphi(Y, \psi(Z))$ для функции $f(X)$ существует $\leftrightarrow$ всякая функция u ее Y -компоненты $f(c_1, \dots, c_m, Z) \in \{0, 1, \psi(Z), \neg\psi(Z)\}$.

Доказательство. *Достаточность.* Пусть $f(c_1, \dots, c_m, Z) \in \{0, 1, \psi(Z), \neg\psi(Z)\} \forall (c_1, \dots, c_m) \in E_2^m$. Пусть $C = (c_1, \dots, c_m)$, $Y^c = x_1^{c_1} \dots x_m^{c_m}$. Построим функцию $\varphi(Y, u)$ следующим образом.

$$f\left(\underbrace{x_1, \dots, x_m}_Y, \underbrace{x_{m+1}, \dots, x_n}_Z\right) = \bigvee_{(c_1, \dots, c_m) \in E_2^m} \underbrace{x_1^{c_1} \dots x_m^{c_m}}_{Y^C} f\left(\underbrace{c_1, \dots, c_m}_C, \underbrace{x_{m+1}, \dots, x_n}_Z\right),$$

где $f(C, Z)$ есть одна из Y -компонент в $\{0, 1, \psi(Z), \neg\psi(Z)\}$. Группируем слагаемые относительно $0, 1, \psi(Z), \neg\psi(Z)$.

$$\begin{aligned} f(Y, Z) &= \left(\bigvee_{f(C, Z) \equiv 0} Y^C \right) \cdot 0 \vee \underbrace{\left(\bigvee_{f(C, Z) \equiv 1} Y^C \right)}_{h_1(Y)} \cdot 1 \vee \underbrace{\left(\bigvee_{f(C, Z) \equiv \psi(Z)} Y^C \right)}_{h_2(Y)} \cdot \psi(Z) \vee \\ &\underbrace{\left(\bigvee_{f(C, Z) \equiv \neg\psi(Z)} Y^C \right)}_{h_3(Y)} \cdot \overline{\psi(Z)} = h_1(Y) \vee h_2(Y) \cdot \psi(Z) \vee h_3(Y) \cdot \overline{\psi(Z)}. \end{aligned}$$

Положим $\varphi(Y, u) = h_1(Y) \vee h_2(Y) \cdot u \vee h_3(Y) \cdot \bar{u}$. Тогда $f(Y, Z) = \varphi(Y, \psi(Z))$, то есть функция f допускает простую непересекающуюся декомпозицию.

Необходимость. Пусть функция $f(Y, Z)$ допускает простую непересекающуюся декомпозицию $f(Y, Z) = \varphi(Y, \psi(Z))$ при некоторых функциях φ и ψ . Разложим функцию f по переменным Y .

$$\begin{aligned} A &= f\left(\underbrace{x_1, \dots, x_m}_Y, \underbrace{x_{m+1}, \dots, x_n}_Z\right) = \varphi\left(\underbrace{x_1, \dots, x_m}_Y, \psi\left(\underbrace{x_{m+1}, \dots, x_n}_Z\right)\right) = \\ &\bigvee_{C=(c_1, \dots, c_m) \in E_2^m} \underbrace{x_1^{c_1} \dots x_m^{c_m}}_{Y^C} \underbrace{\varphi(c_1, \dots, c_m, \psi(x_{m+1}, \dots, x_n))}_{\varphi(C, \psi(Z))} = \left(\bigvee_{C \in E_2^m} Y^C \cdot \varphi(C, u) \right) \Big|_{u=\psi(Z)}. \end{aligned}$$

Как функция одной переменной u , функция $\varphi(C, u) \in \{0, 1, u, \bar{u}\}$. Группируем слагаемые относительно $0, 1, u, \bar{u}$ и получаем:

$$\begin{aligned}
A = & \left(\bigvee_{\varphi(C,u)=0} Y^C \cdot \underbrace{\varphi(C,u)}_{\equiv 0} \vee \bigvee_{\varphi(C,u)=1} Y^C \cdot \underbrace{\varphi(C,u)}_{\equiv 1} \right) \vee \\
& \bigvee_{\varphi(C,u)=u} Y^C \cdot \underbrace{\varphi(C,u)}_{\equiv u} \vee \bigvee_{\varphi(C,u)=\neg u} Y^C \cdot \underbrace{\varphi(C,u)}_{\equiv \neg u} \Bigg|_{u=\psi(Z)} = \\
& \bigvee_{\varphi(C,u)=0} Y^C \cdot \underbrace{\varphi(C,\psi(Z))}_{\equiv 0} \vee \bigvee_{\varphi(C,u)=1} Y^C \cdot \underbrace{\varphi(C,\psi(Z))}_{\equiv 1} \vee \\
& \quad f(C,Z) \in Y\text{-компл} \quad f(C,Z) \in Y\text{-компл} \\
& \bigvee_{\varphi(C,u)=u} Y^C \cdot \underbrace{\varphi(C,\psi(Z))}_{\equiv \psi(Z)} \vee \bigvee_{\varphi(C,u)=\neg u} Y^C \cdot \underbrace{\varphi(C,\psi(Z))}_{\equiv \neg \psi(Z)} . \\
& \quad f(C,Z) \in Y\text{-компл} \quad f(C,Z) \in Y\text{-компл}
\end{aligned}$$



Объединим все слагаемые одной дизъюнкцией.

$$A = \bigvee_{C=(c_1, \dots, c_m) \in E_2^m} Y^C \cdot \underbrace{f(C,Z)}_{\in \{0,1,\psi(Z),\neg\psi(Z)\}} = \bigvee_{C=(c_1, \dots, c_m) \in E_2^m} x_1^{c_1} \dots x_m^{c_m} \underbrace{f(c_1, \dots, c_m, x_{m+1}, \dots, x_n)}_{\in \{0,1,\psi(Z),\neg\psi(Z)\}},$$

то есть всякая функция $f(c_1, \dots, c_m, x_{m+1}, \dots, x_n)$ из Y -компоненты лежит в $\{0, 1, \psi(Z), \neg\psi(Z)\}$. Теорема доказана.

Пример 1. Найти простую непересекающуюся декомпозицию функции $f(x_1, x_2, x_3, x_4)$, $Y = \{x_1, x_2\}$, $Z = \{x_3, x_4\}$ (рис. 4.15).

	x_3x_4	$x_3\bar{x}_4$	$\bar{x}_3x_4$	$\bar{x}_3\bar{x}_4$	Функции Y -компоненты
x_1x_2	0 0	0 1	1 1	1 0	
0 0	1	0	1	1	$f(0, 0, x_3, x_4) = 0$
0 1	0	0	0	0	$f(0, 1, x_3, x_4) = 0$
1 1	1	0	1	1	$f(1, 1, x_3, x_4) = \psi(x_3, x_4)$
1 0	0	1	0	0	$f(1, 0, x_3, x_4) = \neg\psi(x_3, x_4)$

Рис. 4.15

Все функции Y -компоненты (по строкам) лежат в $\{0, 1, \psi(x_3, x_4), \neg\psi(x_3, x_4)\}$. Функция f допускает простую непересекающуюся декомпозицию

$$\begin{aligned}
f(x_1, x_2, x_3, x_4) = & f(0, 0, x_3, x_4) \cdot \bar{x}_1 \bar{x}_2 \vee f(0, 1, x_3, x_4) \cdot \bar{x}_1 x_2 \vee \\
& f(1, 0, x_3, x_4) \cdot x_1 \bar{x}_2 \vee f(1, 1, x_3, x_4) \cdot x_1 x_2 = \\
& \bar{x}_1 \bar{x}_2 \cdot \psi(x_3, x_4) \vee \bar{x}_1 x_2 \cdot 0 \vee x_1 \bar{x}_2 \cdot \psi(x_3, x_4) \vee x_1 x_2 \cdot \psi(x_3, x_4) = \\
& (x_1 \bar{x}_2 \cdot u \vee x_1 \bar{x}_2 \cdot \bar{u} \vee x_1 x_2 \cdot u)|_{u=\psi(x_3, x_4)} = \varphi(x_1, x_2, \psi(x_3, x_4)),
\end{aligned}$$



где $\varphi(x_1, x_2, u) = \bar{x}_1 \bar{x}_2 \cdot u \vee \bar{x}_1 x_2 \cdot \bar{u} \vee x_1 x_2 \cdot u$,

$$\psi(x_3, x_4) = 1011 = x_3 x_4 \vee x_3 \bar{x}_4 \vee x_5 x_4.$$

Реализация функции

$$f(x_1, x_2, x_3, x_4) = \bar{x}_1 \bar{x}_2 \cdot \psi(x_3, x_4) \vee \bar{x}_1 x_2 \cdot 0 \vee x_1 \bar{x}_2 \cdot \psi(x_3, x_4) \vee x_1 x_2 \cdot \psi(x_3, x_4)$$

с помощью мультиплексора приведена на рис. 4.16.

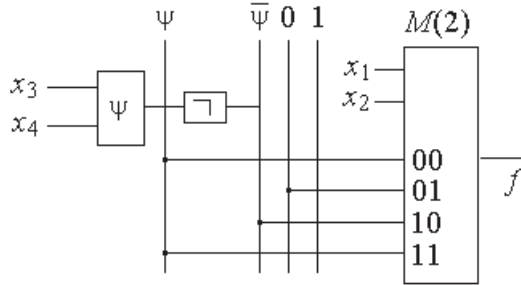


Рис. 4.16

Пример 2. Найти простую непересекающуюся декомпозицию функции $f(x_1, x_2, x_3, x_4)$ (рис. 4.17.).

	$x_3 x_4$	$x_3 x_4$	$x_3 x_4$	$x_3 x_4$	Функции Y-компоненты
$x_1 x_2$	0 0	0 1	1 1	1 0	
0 0	0	1	0	0	$f(0, 0, x_3, x_4) = \varphi(x_3, x_4)$
0 1	0	0	0	0	$f(0, 1, x_3, x_4) = 0$
1 1	0	1	1	0	$f(1, 1, x_3, x_4) = \psi(x_3, x_4)$
1 0	1	1	1	1	$f(1, 0, x_3, x_4) = 1$

Рис. 4.17

Функция f простую непересекающуюся декомпозицию не допускает, ибо функции ее Y-компонент не лежат в $\{0, 1, \psi(x_3, x_4), \neg\psi(x_3, x_4)\}$.

Замечание. Пусть непересекающиеся множества наборов $G_1 \subseteq E_2^n$, $G_2 \subseteq E_2^n$ в объединении дают E_2^n . Пусть функции

$$g_1(x_1, \dots, x_n) = \bigvee_{(c_1, \dots, c_n) \in G_1} x_1^{c_1} \dots x_n^{c_n}, \quad g_2(x_1, \dots, x_n) = \bigvee_{(c_1, \dots, c_n) \in G_2} x_1^{c_1} \dots x_n^{c_n}.$$

Тогда $g_1(x_1, \dots, x_n) = \neg g_2(x_1, \dots, x_n)$.

Пусть $X = \{x_1, \dots, x_n\}$, $Y = \{x_1, \dots, x_m\}$, $Z = \{x_{m+1}, \dots, x_n\}$.

Теорема 2. Функция $f(X)$ допускает простую непересекающуюся декомпозицию $f(Y, Z) = \varphi(Y, \psi(Z)) \leftrightarrow$ функция f имеет в Z-компоненте не более двух различных функций.

Доказательство. Необходимость. Пусть простая непересекающаяся декомпозиция $f(Y, Z) = \varphi(Y, \psi(Z))$. Разложим функцию f по переменным Z. Тогда

$$A = f\left(\underbrace{x_1, \dots, x_m}_Y, \underbrace{x_{m+1}, \dots, x_n}_Z\right) = \bigvee_{C=(c_{m+1}, \dots, c_n) \in E_2^{n-m}} x_{m+1}^{c_{m+1}} \dots x_n^{c_n} f(x_1, \dots, x_m, c_{m+1}, \dots, c_n),$$

Z -компонента $f(x_1, \dots, x_m, c_{m+1}, \dots, c_n) = \varphi(x_1, \dots, x_m, \psi(c_{m+1}, \dots, c_n)) = \varphi(Y, \psi(C))$ есть $\varphi(Y, 0)$ или $\varphi(Y, 1)$, то есть функция f имеет не более двух функций $\varphi(Y, 0)$, $\varphi(Y, 1)$ в своей Z -компоненте.

Достаточность. Пусть функция f имеет не более двух функций $h_1(Y)$, $h_2(Y)$ в своей Z -компоненте. Разложим функцию f по переменным Z :

$$A = f\left(\underbrace{x_1, \dots, x_m}_Y, \underbrace{x_{m+1}, \dots, x_n}_Z\right) = \bigvee_{C=(c_{m+1}, \dots, c_n) \in E_2^{n-m}} \underbrace{x_{m+1}^{c_{m+1}} \dots x_n^{c_n}}_{Z^C} f\left(\underbrace{x_1, \dots, x_m, c_{m+1}, \dots, c_n}_{h_1(Y) \text{ или } h_2(Y)}\right) =$$

[группируем слагаемые относительно $h_1(Y)$ и $h_2(Y)$]

$$\left(\bigvee_{\substack{f(Y, C)=h_1(Y) \\ g_1(Z)}} Z^C \right) \cdot h_1(Y) \vee \left(\bigvee_{\substack{f(Y, C)=h_2(Y) \\ g_2(Z)}} Z^C \right) \cdot h_2(Y) = g_1(Z) \cdot h_1(Z) \vee g_2(Z) \cdot h_2(Z).$$

По вышеприведенному замечанию $g_2(Z) = \overline{g_1(Z)}$. Положим $\psi(Z) = g_1(Z)$.

$$A = g_1(Z) \cdot h_1(Z) \vee \overline{g_1(Z)} \cdot h_2(Z).$$

Положим $\varphi(Y, u) = u \cdot h_1(Y) \vee \overline{u} \cdot h_2(Y)$. Тогда

$$\varphi(Y, \psi(Z)) = \psi(Z) \cdot h_1(Y) \vee \overline{\psi(Z)} \cdot h_2(Y) = g_1(Z) \cdot h_1(Y) \vee \overline{g_1(Z)} \cdot h_2(Y) = f(Y, Z),$$

то есть функция f допускает простую непересекающуюся декомпозицию. Теорема доказана.

Пример 1. Найти простую непересекающуюся декомпозицию функции $f(x_1, x_2, x_3, x_4)$, $Y = \{x_1, x_2\}$, $Z = \{x_3, x_4\}$ (рис. 4.18).

	$x_3 x_4$	$x_3 \bar{x}_4$	$\bar{x}_3 x_4$	$\bar{x}_3 \bar{x}_4$	Функции Y -компоненты
$x_1 x_2$	0 0	0 1	1 1	1 0	
0 0	0	1	0	0	$f(x_1, x_2, 0, 0) = h_1(x_1, x_2)$
0 1	1	0	1	1	$f(x_1, x_2, 0, 1) = h_2(x_1, x_2)$
1 1	0	0	0	0	$f(x_1, x_2, 1, 1) = h_1(x_1, x_2)$
1 0	0	0	0	0	$f(x_1, x_2, 1, 0) = h_2(x_1, x_2)$

Рис. 4.18

$$\begin{aligned} f(Y, Z) &= f(x_1, x_2, x_3, x_4) = \\ &= f(x_1, x_2, 0, 0) \cdot \bar{x}_3 \bar{x}_4 \vee f(x_1, x_2, 0, 1) \cdot \bar{x}_3 x_4 \vee f(x_1, x_2, 1, 0) \cdot x_3 \bar{x}_4 \vee f(x_1, x_2, 1, 1) \cdot x_3 x_4 = \\ &= h_1(x_1, x_2) \bar{x}_3 \bar{x}_4 \vee h_2(x_1, x_2) \bar{x}_3 x_4 \vee h_1(x_1, x_2) x_3 \bar{x}_4 \vee h_2(x_1, x_2) x_3 x_4 = \end{aligned}$$

$$\begin{aligned}
 & h_1(x_1, x_2) \cdot \underbrace{(\bar{x}_3 \bar{x}_4 \vee x_3 \bar{x}_4 \vee x_3 x_4)}_{\psi(x_3, x_4)} \vee h_2(x_1, x_2) \cdot \underbrace{\bar{x}_3 x_4}_{\neg \psi(x_3, x_4)} = \\
 & (h_1(x_1, x_2) \cdot \psi(x_3, x_4) \vee h_2(x_1, x_2) \cdot \neg \psi(x_3, x_4)) = \\
 & \left(\underbrace{h_1(x_1, x_2) \cdot u \vee h_2(x_1, x_2) \cdot \bar{u}}_{\varphi(Y, u)} \right)_{u=\psi(x_3, x_4)} = \varphi(Y, \psi(x_3, x_4))
 \end{aligned}$$

Функция f допускает простую непересекающуюся декомпозицию $f(Y, Z) = \varphi(Y, \psi(Z))$. Реализация функции

$$f(x_1, x_2, x_3, x_4) = h_1(x_1, x_2) \bar{x}_3 \bar{x}_4 \vee h_2(x_1, x_2) \bar{x}_4 x_4 \vee h_1(x_1, x_2) x_3 \bar{x}_4 \vee h_1(x_1, x_2) x_3 x_4$$

с помощью мультиплексора приведена на рис. 4.19.

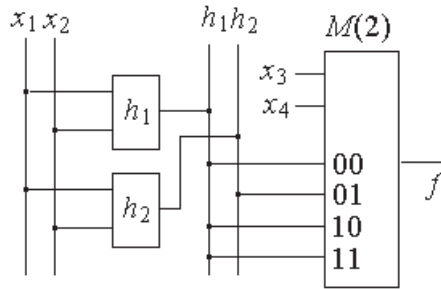


Рис. 4.19

4.6. Обнаружение неисправностей в схемах

Пусть мы конструируем схему из функциональных элементов для данной функции. В результате брака схема может реализовать другую функцию. Задача контроля состоит в том, чтобы определить:

- 1) исправна ли схема, то есть реализует ли схема данную функцию;
- 2) какую функцию реализует схема в случае неисправности.

Обнаружение неисправностей в схеме (тестирование схемы) есть важная и непростая задача. Для знакомства с ней мы рассмотрим следующий простой случай.

Пусть мы конструируем схему S из функциональных элементов для функции $f_0(x_1, \dots, x_n)$. В результате брака схема S может реализовать другую функцию из числа функций $f_1, f_2, \dots, f_r$ от n переменных. Задача контроля состоит в том, чтобы определить:

- 1) исправна ли схема, то есть реализует ли схема функцию f_0 ;
- 2) какую функцию из $f_1, f_2, \dots, f_r$ реализует схема в случае неисправности (табл. 4.1).

Таблица 4.1

$x_1, \dots, x_{n-1}, x_n$	f_0	f_1	f_2	$\dots$	f_r
0 ... 0 0	α_0	β_0	γ_0	$\dots$	δ_0
0 ... 0 1	α_1	β_1	γ_1	$\dots$	δ_1
...	...	...	...	...	...
$a_1 \dots a_{n-1} a_n$	α_p	β_p	γ_p	$\dots$	δ_p
...	...	...	...	...	...
1 ... 1 1	α_z	β_z	γ_z		δ_z

где $z = 2^n - 1$.

Определение. Тест (тестовый набор) для таблично заданной функции $f_0(x_1, \dots, x_n)$ есть совокупность наборов длины n из 0 и 1 (совокупность строк в таблице функции f), которая высекает из столбцов значения функций $f_0, f_1, f_2, \dots, f_r$, причем:

- 1) все высекаемые столбцы различны, и тогда *тест полный*;
- 2) все высекаемые столбцы отличны от столбца для f_0 , и тогда *тест проверяющий*.

Замечание. 1. Полный тест есть проверяющий тест.

2. Множество всех наборов длины n есть полный тест.

Определение. Сложность теста есть число входящих в него наборов.

Определение. Тест минимальный (тупиковый), если удаление из него любого набора приводит к совокупности наборов, которая тестом уже не является. Тест наименьший, если он имеет наименьшую сложность.

Замечание. Все наименьшие тесты находятся среди тупиковых тестов.

Пример построения всех тупиковых и наименьших тестов

Пусть строится схема для функции $f(x_1, \dots, x_n)$. Примем список следующих неисправностей.

s_{ij} , слипание входов x_i и x_j . Тогда реализуется функция

$$g(\dots, x_i, \dots, x_j, \dots) = f(\dots, x_i \vee x_j, \dots, x_i \vee x_j, \dots).$$

0_i , обрыв входа x_i . Тогда реализуется функция $g(\dots, x_i, \dots) = f(\dots, 0, \dots)$.

1_i , замыкание входа x_i . Тогда реализуется функция $g(\dots, x_i, \dots) = f(\dots, 1, \dots)$.

Построим тесты для функции $f_0 = 10110110$ для группы неисправностей $\{0_3, s_{12}\}$. Возможны только указанные неисправности, причем каждая схема может иметь только одну неисправность.

Для неисправности 0_3 функция $f_1(x_1, x_2, x_3) = f_0(x_1, x_2, 0)$.

Для неисправности s_{12} функция $f_2(x_1, x_2, x_3) = f_0(x_1 \vee x_2, x_1 \vee x_2, x_3)$.

Пусть $y_0, \dots, y_7$ есть восемь наборов длины 3 из 0 и 1 (табл. 4.2). Каждый тест имеет вид $y_{i_1} \& y_{i_2} \& \dots \& y_{i_p}$.

Построим все тупиковые проверяющие тесты.

f_0 от f_1 отличают строки y_1, y_5, y_7 . Положим $D_{f_0, f_1} = y_1 \vee y_5 \vee y_7$.

f_0 от f_2 отличают строки y_3, y_4, y_5 . Положим $D_{f_0, f_2} = y_3 \vee y_4 \vee y_5$.

Построим все тупиковые проверяющие тесты.

$$D_{f_0, f_1} \& D_{f_0, f_2} = (y_1 \vee y_5 \vee y_7)(y_3 \vee y_4 \vee y_5) =$$

$$y_1 y_3 \vee y_1 y_4 \vee y_1 y_5 \vee y_3 y_5 \vee y_4 y_5 \vee y_5 \vee y_3 y_7 \vee y_4 y_7 \vee y_5 y_7 = y_1 y_3 \vee y_1 y_4 \vee y_5 \vee y_3 y_7 \vee y_4 y_7.$$

Таблица 4.2

	$x_1 x_2 x_3$	f_0	f_1	f_2
y_0	0 0 0	1	1	1
y_1	0 0 1	0	1	0
y_2	0 1 0	1	1	1
y_3	0 1 1	1	1	0
y_4	1 0 0	0	0	1
y_5	1 0 1	1	0	0
y_6	1 1 0	1	1	1
y_7	1 1 1	0	1	0



Получили пять тупиковых проверяющих тестов. Наименьший проверяющий тест есть y_5 :

	$x_1 x_2 x_3$	$f_0 f_1 f_2$
y_5	1 0 1	1 0 0

Подает на вход схемы набор 101. Если на выходе 1, то схема реализует функцию f_0 . Если на выходе 0, то схема неисправна и функцию f_0 не реализует.

Построим все полные тупиковые тесты.

f_1 от f_2 отличают строки y_1, y_3, y_4, y_7 .

Положим $D_{f_1, f_2} = y_1 \vee y_3 \vee y_4 \vee y_7$. Тогда

$$D_{f_0, f_1} \& D_{f_0, f_2} \& D_{f_1, f_2} = (y_1 \vee y_5 \vee y_7)(y_3 \vee y_4 \vee y_5)(y_1 \vee y_3 \vee y_4 \vee y_7) =$$

$$(y_1 y_3 \vee y_1 y_4 \vee y_5 \vee y_3 y_7 \vee y_4 y_7)(y_1 \vee y_3 \vee y_4 \vee y_7) = y_1 y_3 \vee y_1 y_3 \vee y_1 y_5 \vee y_1 y_3 y_7 \vee y_1 y_4 y_7 \vee y_1 y_3 \vee y_1 y_3 y_4 \vee y_3 y_5 \vee y_3 y_7 \vee y_3 y_4 y_7 \vee y_1 y_3 y_4 \vee y_1 y_4 \vee y_4 y_5 \vee y_3 y_4 y_7 \vee y_4 y_7 \vee y_1 y_3 y_7 \vee y_1 y_4 y_7 \vee y_5 y_7 \vee y_3 y_7 \vee y_4 y_7 = y_1 y_3 \vee y_1 y_4 \vee y_1 y_5 \vee y_3 y_5 \vee y_3 y_7 \vee y_4 y_5 \vee y_4 y_7 \vee y_5 y_7.$$

Получили 8 полных тупиковых тестов. Они все наименьшие.

Например, $y_1 y_3$:

	$x_1 x_2 x_3$	$f_0 f_1 f_2$
y_1	0 0 1	0 1 0
y_3	0 1 1	1 1 0



Подает на вход схемы последовательно наборы y_1, y_3 , то есть наборы $\begin{matrix} 0 & 0 & 1 \\ 0 & 1 & 1 \end{matrix}$.

Пара $\begin{matrix} 0 \\ 1 \end{matrix}$ на выходе укажет, что схема реализует функцию f_0 , пара $\begin{matrix} 1 \\ 1 \end{matrix}$ – функцию f_1 , пара $\begin{matrix} 0 \\ 0 \end{matrix}$ – функцию f_2 .

Глава 5

Аксиоматическое исчисление высказываний

5.1. Определение исчисления высказываний

Высказывание есть утверждение, относительно которого можно точно сказать, истинно оно или ложно (при этом оно не может быть тем и другим одновременно). Высказывание «через две точки можно провести единственную прямую» истинно. Высказывание «нуль равен единице» ложно. Из простых высказываний с помощью логических операций конъюнкции, дизъюнкции, отрицания, импликации, эквивалентности можно строить высказывания более сложные. Их истинностные значения «истина» и «ложь» (И и Л) вычисляются согласно следующей таблице:

p	q	$p \& q$	$p \vee q$	$p \rightarrow q$	$p \equiv q$	$\neg p$
Л	Л	Л	Л	И	И	И
Л	И	Л	И	И	Л	И
И	Л	Л	И	Л	Л	Л
И	И	И	И	И	И	Л

Высказывание «если числовой ряд сходится, то его общий член стремится к нулю» построено из двух высказываний: «числовой ряд сходится» и «его общий член стремится к нулю» с помощью импликации.

Пусть $A(p_1, p_2, \dots, p_k)$ есть формула алгебры логики, которая построена из пропозициональных переменных $p_1, p_2, \dots, p_k$ с помощью конъюнкции, дизъюнкции, отрицания, импликации, эквивалентности. Замещая переменные $p_1, p_2, \dots, p_k$ некоторыми высказываниями, получим новое, которое примет значение «истина» или «ложь» (И или Л). Содержание самих высказываний при этом не существенно; важны только их истинностные значения. Поэтому в формуле $A(p_1, p_2, \dots, p_k)$ переменные можно сразу замещать знаками И и Л, а затем вычислять истинностное значение получившегося выражения, как это делается

в алгебре логики, если считать, что И отождествляется с 1, а Л с 0. Формула $A(p_1, p_2, \dots, p_k)$ реализует некоторую функцию алгебры логики от k переменных. Не будем различать и оговаривать их особо в дальнейшем, если о такой функции придется говорить.

Содержательно значения конъюнкции, дизъюнкции, отрицания, эквивалентности понятны. Значения импликации $p \rightarrow q$ на наборах (1,1) и (1,0) тоже понятны. Несогласие вызывают у слушателей курса логики в первый раз значения функции $p \rightarrow q$ на наборах (0,0) и (0,1). Выбранную таблицу значений импликации можно пояснить следующими рассуждениями. Функции $p \rightarrow q$ и $\neg q \rightarrow \neg p$ должны быть одинаковыми, ибо если утверждение $p \rightarrow q$ при некоторых конкретных высказываниях p и q истинно (как прямое утверждение), то высказывание $\neg q \rightarrow \neg p$ тоже истинно (как противоположное утверждение). Например, справедливо необходимое условие сходимости числовых рядов: если ряд сходится, то его общий член стремится к нулю. Справедливо и противоположное утверждение: если общий член ряда не стремится к нулю, то ряд расходится. Из этих же соображений верно и обратное: из истинности утверждения $\neg q \rightarrow \neg p$ следует истинность утверждения $p \rightarrow q$.

Функции $p \rightarrow q$ и $q \rightarrow p$ не могут быть равными, иначе импликация становится эквивалентностью, что невозможно, ибо из того, что p влечет q , вообще говоря, не следует, что q влечет p (пример с тем же необходимым условием сходимости рядов). Таким образом, импликация $p \rightarrow q$ определяется следующими условиями:

- 1) функции $p \rightarrow q$ и $\neg q \rightarrow \neg p$ одинаковы;
- 2) функции $p \rightarrow q$ и $q \rightarrow p$ различны.

Итак, функция $p \rightarrow q$ на наборе (1,1) равна единице, а на наборе (1,0) – нулю. Такова же и равная ей функция $\neg q \rightarrow \neg p$. На наборе (0,0) функция $\neg q \rightarrow \neg p$ равна единице; такова же и $p \rightarrow q$. Значения импликации $q \rightarrow p$ на наборах (0,0), (1,0), (1,1) определились. Осталось задать значение $p \rightarrow q$ на (0,1). Если примем, что $p \rightarrow q$ на (0,1) равно 0 (т. е. $0 \rightarrow 1 = 0$), то $q \rightarrow p$ на (1,0) тоже равно 0 (т. е. $1 \rightarrow 0 = 1$). Тогда функции $p \rightarrow q$ и $q \rightarrow p$ оказываются равными. Поэтому импликация $p \rightarrow q$ на (0,1) равна 1.

Построим формально-логическое исчисление, в котором описываются все тождественно истинные формулы алгебры логики и только они.

Алфавит

1. $p, q, r, p_1, q_1, r_1, \dots$ есть символы пропозициональных переменных.
2. $\&, \vee, \rightarrow, \neg$ есть логические связи (конъюнкция, дизъюнкция, импликация, отрицание соответственно).
3. $(,)$ есть скобка левая, запятая, скобка правая.

Формулы

1. Отдельно взятый символ пропозициональной переменной есть элементарная формула (атом).
2. Если A и B есть формулы, то выражения $(A \& B)$, $(A \vee B)$, $(A \rightarrow B)$, $(\neg A)$ есть формулы.

Примем соглашение об опускании скобок в соответствии со следующим приоритетом связок: $\neg$, $\&$, $\vee$, $\rightarrow$.

Подформулы

1. Если формула есть переменная, то единственной ее подформулой является она сама.
2. Если формула имеет вид $\neg A$, то ее подформулами являются она сама и все подформулы формулы A .
3. Если формула имеет вид $A * B$, где знак $*$ $\in \{\&, \vee, \rightarrow\}$, то ее подформулами являются она сама и все подформулы формул A и B .

В формуле $A \rightarrow B$ подформула A называется *посылкой*, а подформула B – *заключением*.

Проинтерпретируем логические связки $\&$, $\vee$, $\rightarrow$, $\neg$ соответствующими логическими операциями (функциями).

Если все переменные формулы A содержатся среди переменных $p_1, p_2, \dots, p_n$, то пишем $A(p_1, p_2, \dots, p_n)$. Если формула $A(p_1, p_2, \dots, p_n)$ построена из переменных $p_1, p_2, \dots, p_n$ и только из них, то скажем тогда, что $p_1, p_2, \dots, p_n$ есть полный список переменных формулы A .

Определение. Интерпретация формулы $A(p_1, \dots, p_n)$ есть набор $I = (a_1, \dots, a_n)$ из И и Л. $\square$

Индукцией по построению формулы A определим понятие значения $I(A)$ формулы $A(p_1, \dots, p_n)$ на интерпретации I .

1. Если A есть p_i , то $I(A) = a_i$.
2. Если A есть формула $(B * C)$, где знак $*$ $\in \{\&, \vee, \rightarrow\}$, или же формула $\neg B$, то $I(B * C) = I(B) * I(C)$; $I(\neg B) = \neg(I(B))$.

Для удобства вместо $I(A)$ будем писать также $A(a_1, \dots, a_n)$.

Определение. Формула A *общезначима* (тождественно истинна), если она истинна на любой интерпретации. Формула A *выполнима*, если существует интерпретация, на которой формула A истинна. Формула A *невыполнима* (тождественно ложна), если она ложна на всех интерпретациях. Формула A *опровержима*, если существует интерпретация, на которой формула A ложна.

Определение. Интерпретация I *удовлетворяет* формуле A (A выполняется на I), если A истинна на I (то есть $A(I) = И$). Интерпретация I *опровергает* A , если A ложна на I (то есть $A(I) = Л$).

Подстановка (П). Индуктивное определение.

1. Если формула A есть переменная p , то подстановка $\Pi_p^B(A)$ формулы B на место переменной p в формуле A есть B .
2. Если формула A переменной p не содержит, то подстановка $\Pi_p^B(A)$ есть снова формула A .
3. Если формула A имеет вид $\neg C$ или $C * E$, где $*$ есть один из знаков $\&$, $\vee$, $\rightarrow$, то подстановки

$$\Pi_p^B(\neg C) = \neg \Pi_p^B(C), \Pi_p^B(C * E) = \Pi_p^B(C) * \Pi_p^B(E).$$

Подстановка $\Pi_p^B(A)$ есть результат замещения переменной p всюду, где она встречается в формуле A , на формулу B .

Система аксиом (П. С. Новиков)

- A1) $p \rightarrow (q \rightarrow p)$.
 A2) $(p \rightarrow (q \rightarrow r)) \rightarrow ((p \rightarrow q) \rightarrow (p \rightarrow r))$.
 A3) $p \ \& \ q \rightarrow p$.
 A4) $p \ \& \ q \rightarrow q$.
 A5) $(p \rightarrow q) \rightarrow ((p \rightarrow r) \rightarrow (p \rightarrow q \ \& \ r))$.
 A6) $p \rightarrow p \vee q$.
 A7) $q \rightarrow p \vee q$.
 A8) $(p \rightarrow r) \rightarrow ((q \rightarrow r) \rightarrow (p \vee q \rightarrow r))$.
 A9) $(p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)$.
 A10) $p \rightarrow \neg \neg p$.
 A11) $\neg \neg p \rightarrow p$.



Заметим, что в качестве аксиом взяты тождественно истинные формулы.

Правила вывода (ПВ)

Правило подстановки. $\frac{A(p)}{\Pi_p^B(A(p))}$, т. е. от формулы $A(p)$ с переменной p можно перейти к формуле $\Pi_p^B(A(p))$.

Правило заключения (ПЗ). $\frac{A, A \rightarrow B}{B}$, т. е. от двух формул A и $A \rightarrow B$ можно перейти к формуле B . Это правило известно еще как правило «modus ponens» (утверждающий модус).

Исчисление высказываний (ИВ) составляют формальные символы, формулы, аксиомы и правила вывода.

Определение. Доказательство в ИВ есть конечная последовательность формул, каждая из которых есть либо аксиома, либо получена из предыдущих членов последовательности по правилу подстановки или заключения.

Определение. Формула A доказуема в ИВ, если в ИВ существует доказательство, последней формулой которого является A . $\square$

Всякая аксиома есть доказуемая формула, длина доказательства которой равна 1.

Приведем примеры доказуемых формул и некоторых вспомогательных (производных) правил вывода, применение которых к формулам при необходимости можно исключить, но использование их сделает формальную систему ИВ выразительнее и естественнее.

Доказательства формул сопроводим краткими пояснениями. Например, A3 будет означать, что формула является аксиомой A3; ПЗ(2,3) есть формула, полученная применением правила заключения (ПЗ) к формулам (2) и (3); $\Pi_q^{-p}(1)$ есть формула, полученная подстановкой в формулу (4) формулы $\neg p$ на место переменной q . Знак $\vdash$ будет означать: формула доказуема в ИВ.

Теорема 1. $\vdash p \rightarrow p$.

Доказательство.

(1) $(p \rightarrow (q \rightarrow r)) \rightarrow ((p \rightarrow q) \rightarrow (p \rightarrow r))$, A2;

(2) $(p \rightarrow (q \rightarrow p)) \rightarrow ((p \rightarrow q) \rightarrow (p \rightarrow p))$, $\Pi_r^p(1)$;

(3) $p \rightarrow (q \rightarrow p)$, A1;

(4) $(p \rightarrow q) \rightarrow (p \rightarrow p)$, ПЗ(2,3);

(5) $(p \rightarrow (q \rightarrow p)) \rightarrow (p \rightarrow p)$, $\Pi_q^{q \rightarrow p}(4)$;

(6) $p \rightarrow p$, ПЗ(3,5).

Формула $p \rightarrow p$ доказуема в ИВ, и последовательность формул (1), (2), ..., (6) есть ее доказательство.

Замечание. Формула $p \rightarrow p$ доказуема в любом аксиоматическом исчислении, содержащем аксиомы A1, A2, правило подстановки и правило заключения.

Правило одновременной подстановки

Пусть выражение $\prod_{p_1, p_2, \dots, p_k}^{B_1 B_2 \dots B_k}(A)$ означает формулу, представляющую собой результат одновременного замещения переменных $p_1, p_2, \dots, p_k$ в формуле A на формулы $B_1, B_2, \dots, B_k$ соответственно.

Утверждение.
$$\frac{\vdash A(p_1, p_2, \dots, p_n)}{\vdash \prod_{p_1, p_2, \dots, p_n}^{B_1 B_2 \dots B_n}(A(p_1, p_2, \dots, p_n))}.$$

Доказательство. Пусть $r_1, r_2, \dots, r_k$ есть попарно различные переменные, отличные от всех переменных, встречающихся в формулах $A, B_1, B_2, \dots, B_k$. Тогда формула $\prod_{p_1, p_2, \dots, p_k}^{B_1 B_2 \dots B_k}(A)$ есть результат сначала последовательного замещения переменных $p_1, p_2, \dots, p_k$ в формуле A на переменные $r_1, r_2, \dots, r_k$ соответственно, а затем в получившейся формуле есть результат последовательной замены переменных $r_1, r_2, \dots, r_k$ на формулы $B_1, B_2, \dots, B_k$ соответственно.

Теорема 2. $\vdash \neg(p \vee q) \rightarrow \neg p \ \& \ \neg q$.

Доказательство.

(1) $(p \rightarrow q) \rightarrow ((p \rightarrow r) \rightarrow (p \rightarrow q \ \& \ r))$, A5;

(2) $(\neg(p \vee q) \rightarrow \neg p) \rightarrow ((\neg(p \vee q) \rightarrow \neg q) \rightarrow (\neg(p \vee q) \rightarrow \neg p \ \& \ \neg q))$, $\Pi_p^{\neg(p \vee q)} \Pi_q^{\neg p} \Pi_r^{\neg q}(1)$;

(3) $p \rightarrow p \vee q$, A6;

(4) $(p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)$, A9;

(5) $(p \rightarrow p \vee q) \rightarrow (\neg(p \vee q) \rightarrow \neg p)$, $\Pi_p^{p \vee q}(4)$;

(6) $\neg(p \vee q) \rightarrow \neg p$, ПЗ(3,5);

(7) $\neg(p \vee q) \rightarrow \neg q \rightarrow (\neg(p \vee q) \rightarrow \neg p \ \& \ \neg q)$, ПЗ(2,6);

(8) $q \rightarrow p \vee q$, A7;

(9) $(q \rightarrow p \vee q) \rightarrow (\neg(p \vee q) \rightarrow \neg q)$, $\Pi_q^{p \vee q}(4)$;

(10) $\neg(p \vee q) \rightarrow \neg q$, ПЗ(8,9);

(11) $\neg(p \vee q) \rightarrow \neg p \ \& \ \neg q$, ПЗ(7,10).

5.2. Теорема дедукции в исчислении высказываний

Пусть Γ и Δ есть произвольные конечные совокупности формул в ИВ (возможно, пустые) и A, B, C есть произвольные формулы в ИВ.

Определение. Конечная последовательность формул ИВ есть вывод из совокупности формул Γ , если всякая формула последовательности либо принадле-

жит Γ , либо доказуема, либо получена из предыдущих формул последовательности по одному из правил вывода.

Определение. Формула A выводима из совокупности формул Γ , если существует вывод из совокупности Γ , последней формулой которого является A . $\square$

Заметим, что доказуемая формула выводима из любой (в том числе из пустой) совокупности формул.

Введем следующие обозначения:

- $\Gamma \vdash A$ – формула A выводима из совокупности формул Γ ;
- Т8 – теорема 8;
- ПВ7 – правило вывода 7.

Правило вывода 1. $A, B \vdash A \& B$.

Доказательство.

- (1) A , условие;
- (2) B , условие;
- (3) $\vdash (p \rightarrow q) \rightarrow ((p \rightarrow r) \rightarrow (p \rightarrow q \& r))$, А5;
- (4) $\vdash (A \rightarrow A) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow A \& B))$, $\Pi_{p,q,r}^{A \& B}$ (3);
- (5) $\vdash p \rightarrow p$, Т1;
- (6) $\vdash A \rightarrow A$;
- (7) $\vdash (A \rightarrow B) \rightarrow (A \rightarrow A \& B)$, ПЗ (4,6);
- (8) $\vdash p \rightarrow (q \rightarrow p)$, А1;
- (9) $\vdash B \rightarrow (A \rightarrow B)$, $\Pi_p^{B \& A}$ (8);
- (10) $A \rightarrow B$, ПЗ(2,9);
- (11) $A \rightarrow A \& B$, ПЗ(7,10);
- (12) $A \& B$, ПЗ(1,11).
 - a. $A \& B \vdash A$;
 - b. $A \& B \vdash B$.

Доказательство.

- (1) $A \& B$, условие;
- (2) $\vdash p \& q \rightarrow p$, А3;
- (3) $\vdash A \& B \rightarrow A$, $\Pi_{p,q}^{A \& B}$ (2);
- (4) A , ПЗ(1,3);
- (1) $A \& B$, условие;
- (2) $\vdash p \& q \rightarrow q$, А4;
- (3) $\vdash A \& B \rightarrow B$, $\Pi_{p,q}^{A \& B}$ (2);
- (4) B , ПЗ(1,3);

Правило вывода 2. a. $\frac{\Gamma \vdash A, \Gamma \vdash B}{\Gamma \vdash A \& B}$; b. $\frac{\Gamma \vdash A \& B}{\Gamma \vdash A, \Gamma \vdash B}$.

Доказательство.

- (1) Γ , условие;
- ...
- (2) A ;
- ...
- (3) B ;
- (4) $A \& B$, ПВ1(2,3).
- (1) Γ , условие;
- ...
- (2) $A \& B$;
- (3) $A \& B \rightarrow A$, $\Pi_{p,q}^{A \& B}$ (А3);
- (4) $A \& B \rightarrow B$, $\Pi_{p,q}^{A \& B}$ (А4);
- (5) A , ПЗ(2,3);
- (6) B , ПЗ(2,4).

Правило вывода 3. $\frac{\Gamma \vdash A}{\Gamma \vdash B \rightarrow A}.$

Доказательство.

- (1) Γ , условие; ...;
- (2) A ;
- (3) $\vdash A \rightarrow (B \rightarrow A)$, $\Pi_{p\ q}^{A\ B}(A1)$;
- (4) $B \rightarrow A$, ПЗ(2,3).

Теорема дедукции (ТД). Если Γ есть конечная совокупность формул, а A и B есть произвольные формулы в ИВ, для которых $\Gamma, A \vdash B$, то $\Gamma \vdash A \rightarrow B$.

Доказательство. Индукция по длине k вывода.

Базис. $k=1$. Возможны три случая.

1. B есть A . Тогда $\Gamma, A \vdash A$. Так как $\vdash A \rightarrow A$ (как $\Pi_p^A(T1)$), то $\Gamma \vdash A \rightarrow A$.
2. $\vdash B$. Тогда $\vdash A \rightarrow B$ по ПВЗ; поэтому $\Gamma \vdash A \rightarrow B$.
3. $B \in \Gamma$. Тогда $\Gamma \vdash B$ и по ПВЗ $\Gamma \vdash A \rightarrow B$.

Предположение индукции. Пусть теорема справедлива для всех выводов длины меньше k .

Шаг индукции. Покажем, что теорема справедлива для выводов длины k . Пусть $\Gamma, A \vdash B$ и пусть $B_1, B_2, \dots, B_{k-1}, B_k (= B)$ есть вывод длины k формулы B из совокупности Γ, A . Возможны следующие случаи.

1. B есть A .
2. $\vdash B$.
3. $B \in \Gamma$. Эти случаи рассмотрены в базисе.
4. $B (= B_k)$ получается из двух формул B_i и $B_i \rightarrow B_k$ вывода по правилу заключения, т. е. вывод B_k из Γ, A имеет вид

$B_1; B_2; \dots; B_i; \dots; B_i \rightarrow B_k; \dots; B_k (=B).$

Тогда $\Gamma, A \vdash B_i$, $\Gamma, A \vdash B_i \rightarrow B_k$. Так как длины этих выводов меньше k , то по предположению индукции:

- (1) $\Gamma \vdash A \rightarrow B_i$;
- (2) $\Gamma \vdash A \rightarrow (B_i \rightarrow B_k)$.

Далее имеем:

- (3) $\Gamma \vdash (A \rightarrow (B_i \rightarrow B_k)) \rightarrow ((A \rightarrow B_i) \rightarrow (A \rightarrow B_k))$, $\Pi_{p\ q\ r}^{A\ B_i\ B_k}(3)$;
- (4) $\Gamma \vdash (A \rightarrow B_i) \rightarrow (A \rightarrow B_k)$, ПЗ(2,3);
- (5) $\Gamma \vdash A \rightarrow B_k$, ПЗ(1,4), т. е. $\Gamma \vdash A \rightarrow B$.

Следствие. $\frac{A_1, A_2, \dots, A_k \vdash B}{\vdash A_1 \rightarrow (A_2 \rightarrow (\dots \rightarrow (A_k \rightarrow B) \dots))}.$

Замечание. Теорема дедукции доказуема в любом аксиоматическом исчислении, содержащем аксиомы $A1, A2$, правило подстановки и правило заключения.

5.3. Производные правила вывода

Пусть, как и в предыдущем параграфе, Γ и Δ есть произвольные конечные совокупности формул ИВ (возможно, пустые) и A, B, C есть произвольные формулы в ИВ.

Правило вывода 4 (правило силлогизма). $\frac{\Gamma \vdash A \rightarrow B, \Gamma \vdash B \rightarrow C}{\Gamma \vdash A \rightarrow C}$.

Доказательство.

- (1) $\Gamma \vdash A \rightarrow B$, условие;
- (2) $\Gamma \vdash B \rightarrow C$, условие;
- (3) $\Gamma \vdash (A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$, $\Pi_{pqr}^{ABC}(A2)$;
- (4) $\Gamma \vdash A \rightarrow (B \rightarrow C)$, ПВ3(2);
- (5) $\Gamma \vdash (A \rightarrow B) \rightarrow (A \rightarrow C)$ ПЗ(3,4);
- (6) $\Gamma \vdash A \rightarrow C$, ПЗ(1,5).

Правило вывода 5 (контрапозиция). $\frac{\Gamma \vdash A \rightarrow B}{\Gamma \vdash \neg B \rightarrow \neg A}$.

Доказательство.

- (1) $\Gamma \vdash A \rightarrow B$, условие;
- (2) $\Gamma \vdash (A \rightarrow B) \rightarrow (\neg B \rightarrow \neg A)$, $\Pi_{pq}^{AB}(A9)$;
- (3) $\Gamma \vdash \neg B \rightarrow \neg A$, ПЗ(1,2).

Правило вывода 6 (снятие двойного отрицания в заключении).

$\frac{\Gamma \vdash A \rightarrow \neg \neg B}{\Gamma \vdash A \rightarrow B}$.

Доказательство.

- (1) $\Gamma \vdash A \rightarrow \neg \neg B$, условие;
- (2) $\Gamma \vdash \neg \neg B \rightarrow B$, $\Pi_p^B(A10)$;
- (3) $\Gamma \vdash A \rightarrow B$, ПВ4(1,2).

Правило вывода 7 (снятие двойного отрицания в посылке).

$\frac{\Gamma \vdash \neg \neg A \rightarrow B}{\Gamma \vdash A \rightarrow B}$.

Доказательство.

- (1) $\Gamma \vdash \neg \neg A \rightarrow B$, условие;
- (2) $\Gamma \vdash A \rightarrow \neg \neg A$, $\Pi_p^A(A11)$;
- (3) $\Gamma \vdash A \rightarrow B$, ПВ4(1,2).

Правило вывода 8. $\frac{\Gamma \vdash A}{\Gamma, \Delta \vdash A}$.

Правило вывода 9. $\frac{\Gamma, B \vdash A, \Gamma \vdash B}{\Gamma \vdash A}$.

Доказательство. $\Gamma; \dots; B; \dots; A$; т.е. $\Gamma \vdash A$.

Правило вывода 10. $\frac{\Gamma, B \vdash A, \Delta \vdash B}{\Gamma, \Delta \vdash A}$

Доказательство. $\Gamma; \Delta; \dots; B; \dots; A$.

Правило вывода 11. $\frac{\Gamma \vdash A \rightarrow B}{\Gamma, A \vdash B}$.

Доказательство.

- (1) $\Gamma \vdash A \rightarrow B$, условие;
- (2) $\Gamma, A \vdash A$;
- (3) $\Gamma, A \vdash A \rightarrow B$, ПВ8(1);
- (4) $\Gamma, A \vdash B$, ПЗ(2,3).

Правило вывода 12. $\frac{\vdash A}{\Gamma \vdash A}$.

Теорема 3 (закон перестановки посылок).

$\vdash (p \rightarrow (q \rightarrow r)) \rightarrow (q \rightarrow (p \rightarrow r))$.

Доказательство. Покажем, что $p \rightarrow (q \rightarrow r)$, q , $p \vdash r$.

- (1) $p \rightarrow (q \rightarrow r)$, условие;
- (2) q , условие; (4) $q \rightarrow r$, ПЗ(1,3);
- (3) p , условие; (5) r , ПЗ(2,4).

Далее последовательно применяем теорему дедукции.

Правило вывода 13 (перестановка посылок). $\frac{\Gamma \vdash A \rightarrow (B \rightarrow C)}{\Gamma \vdash B \rightarrow (A \rightarrow C)}$.

Доказательство.

- (1) $\Gamma \vdash A \rightarrow (B \rightarrow C)$, условие;
- (2) $\Gamma \vdash (A \rightarrow (B \rightarrow C)) \rightarrow (B \rightarrow (A \rightarrow C))$, $\Pi_{p \ q \ r}^{A \ B \ C}(T3)$;
- (3) $\Gamma \vdash B \rightarrow (A \rightarrow C)$, ПЗ(1,2).

Теорема 4 (закон соединения посылок).

$\vdash (p \rightarrow (q \rightarrow r)) \rightarrow (p \ \& \ q \rightarrow r)$.

Доказательство. Покажем, что $p \rightarrow (q \rightarrow r)$, $p \ \& \ q \vdash r$.

- (1) $p \rightarrow (q \rightarrow r)$, условие; (4) q , ПВ2(2);
- (2) $p \ \& \ q$, условие; (5) $q \rightarrow r$, ПЗ(1,3);
- (3) p , ПВ2(2); (6) r , ПЗ(4,5).

Далее последовательно применяем теорему дедукции.

Правило вывода 14 (соединение посылок). $\frac{\Gamma \vdash A \rightarrow (B \rightarrow C)}{\Gamma \vdash (A \ \& \ B) \rightarrow C}$.

Доказательство.

- (1) $\Gamma \vdash A \rightarrow (B \rightarrow C)$, условие;
- (2) $\Gamma \vdash (A \rightarrow (B \rightarrow C)) \rightarrow (A \ \& \ B \rightarrow C)$, $\Pi_{p \ q \ r}^{A \ B \ C}(T4)$;
- (3) $\Gamma \vdash A \ \& \ B \rightarrow C$, ПЗ(1,2).

Теорема 5 (закон разъединения посылок).

$\vdash (p \ \& \ q \rightarrow r) \rightarrow (p \rightarrow (q \rightarrow r))$.

Доказательство. Покажем, что $p \& q \rightarrow r, p, q \vdash r$.

- (1) $p \& q \rightarrow r$, условие;
- (2) p , условие;
- (3) q , условие;
- (4) $p \& q$, ПВ2(2,3);
- (5) r , ПЗ(1,4).

Далее последовательно применяем теорему дедукции.

Правило вывода 15 (разъединение посылок). $\frac{\Gamma \vdash A \& B \rightarrow C}{\Gamma \vdash A \rightarrow (B \rightarrow C)}$.

Доказательство.

- (1) $\Gamma \vdash A \& B \rightarrow C$, условие;
- (2) $\Gamma \vdash (A \& B \rightarrow C) \rightarrow (A \rightarrow (B \rightarrow C))$, $\Pi_{p \ q \ r}^{ABC}(T5)$;
- (3) $\Gamma \vdash A \rightarrow (B \rightarrow C)$, ПЗ(1,2).

Правило вывода 16 (введение конъюнкции). $\frac{\Gamma \vdash C \rightarrow A, \Gamma \vdash C \rightarrow B}{\Gamma \vdash C \rightarrow A \& B}$.

Доказательство.

- (1) $\Gamma \vdash C \rightarrow A$, условие;
- (2) $\Gamma \vdash C \rightarrow B$, условие;
- (3) $\Gamma \vdash (C \rightarrow A) \rightarrow ((C \rightarrow B) \rightarrow (C \rightarrow A \& B))$, $\Pi_{p \ q \ r}^{CAB}(A5)$;
- (4) $\Gamma \vdash (C \rightarrow B) \rightarrow (C \rightarrow A \& B)$, ПЗ(1,3);
- (5) $\Gamma \vdash C \rightarrow A \& B$, ПЗ(2,4).

Правило вывода 17 (введение дизъюнкции). $\frac{\Gamma \vdash A \rightarrow C, \Gamma \vdash B \rightarrow C}{\Gamma \vdash A \vee B \rightarrow C}$.

Доказательство.

- (1) $\Gamma \vdash A \rightarrow C$, условие;
- (2) $\Gamma \vdash B \rightarrow C$, условие;
- (3) $\Gamma \vdash (A \rightarrow C) \rightarrow ((B \rightarrow C) \rightarrow (A \vee B \rightarrow C))$, $\Pi_{p \ q \ r}^{ABC}(A8)$;
- (4) $\Gamma \vdash (B \rightarrow C) \rightarrow (A \vee B \rightarrow C)$, ПЗ(1,3);
- (5) $\Gamma \vdash A \vee B \rightarrow C$, ПЗ(2,4).

Теорема 6. $\vdash p \& \neg p \rightarrow q$.

Доказательство.

- (1) $\vdash p \rightarrow (q \rightarrow p)$, A1;
- (2) $\vdash p \rightarrow (\neg q \rightarrow p)$, $\Pi_{p \ q}^{-q}(1)$;
- (3) $\vdash (\neg q \rightarrow p) \rightarrow (\neg p \rightarrow \neg\neg q)$, $\Pi_{p \ q}^{-q \ p}(A9)$;
- (4) $\vdash p \rightarrow (\neg p \rightarrow \neg\neg q)$, ПВ4(2,3), силлогизм;
- (5) $\vdash p \& \neg p \rightarrow \neg\neg q$, ПВ14(4), соединение посылок;
- (6) $\vdash p \& \neg p \rightarrow q$, ПВ6(5), снятие двойного отрицания.

Высказывание $p \& \neg p$ тождественно ложно, т. е. противоречиво, при любом p . Теорема 6 утверждает, что из противоречия следует любое высказывание. Из нее легко показать справедливость правила вывода $\frac{A, \neg A}{B}$, которое утверждает,

что в противоречивом исчислении (т. е. в исчислении, где доказуемо некоторое утверждение и его отрицание) можно вывести любое утверждение.

Теорема 7 (закон исключенного третьего). $\vdash p \vee \neg p$.

Доказательство.

- (1) $\vdash \neg(p \vee \neg p) \rightarrow \neg p \ \& \ \neg\neg p, \Pi_q^{\neg p}(\text{T2});$
- (2) $\vdash \neg p \ \& \ \neg\neg p \rightarrow \neg q, \Pi_{p \ q}^{\neg p \neg q}(\text{T6});$
- (3) $\vdash \neg(p \vee \neg p) \rightarrow \neg q, \text{ПВ4}(1,2), \text{силлогизм};$
- (4) $\vdash \neg\neg q \rightarrow \neg\neg(p \vee \neg p), \text{ПВ5}(3), \text{контрапозиция};$
- (5) $\vdash q \rightarrow \neg\neg(p \vee \neg p), \text{ПВ7}(4), \text{снятие двойного отрицания};$
- (6) $\vdash q \rightarrow (p \vee \neg p), \text{ПВ6}(5), \text{снятие двойного отрицания};$
- (7) $\vdash p \rightarrow p, \text{T1};$
- (8) $\vdash (p \rightarrow p) \rightarrow (p \vee \neg p), \Pi_q^{p \rightarrow p}(\text{6});$
- (9) $\vdash p \vee \neg p, \text{П3}(7,8).$

Правило вывода 18 (приведение к абсурду). $\frac{\Gamma, A \vdash B, \Gamma, A \vdash \neg B}{\Gamma \vdash \neg A}$.

Доказательство.

- (1) $\Gamma, A \vdash B$, условие;
- (2) $\Gamma, A \vdash \neg B$, условие;
- (3) $\Gamma \vdash A \rightarrow B$, ТД(1);
- (4) $\Gamma \vdash A \rightarrow \neg B$, ТД(2);
- (5) $\Gamma \vdash A \rightarrow B \ \& \ \neg B, \text{ПВ16}(3,4); \Pi_{p \ q}^{B \vee \neg B}(\text{T6});$
- (6) $\vdash B \ \& \ \neg B \rightarrow \neg(B \vee \neg B),$
- (7) $\Gamma \vdash A \rightarrow \neg(B \vee \neg B), \text{ПВ4}(5,6), \text{силлогизм};$
- (8) $\Gamma \vdash \neg(B \vee \neg B) \rightarrow \neg A, \text{ПВ5}(7), \text{контрапозиция};$
- (9) $\Gamma \vdash (B \vee \neg B) \rightarrow \neg A, \text{ПВ7}(8), \text{снятие двойного отрицания};$
- (10) $\vdash B \vee \neg B, \Pi_p^B(\text{T7});$
- (11) $\Gamma \vdash \neg A, \text{П3}(9,10).$

Правило приведения к абсурду используется в доказательствах от противного. Пусть по ходу некоторых рассуждений требуется доказать утверждение A . Допускаем противное: предполагаем справедливость его отрицания, т. е. справедливость утверждения $\neg A$, и доказываем, исходя из $\neg A$, некоторое утверждение B и его отрицание $\neg B$. Получили противоречие. Поэтому наше предположение о справедливости утверждения $\neg A$ было неверным, и потому справедливо его отрицание $\neg\neg A$, т. е. справедливо A .

Теорема 8. $\vdash p \vee q \rightarrow \neg(\neg p \ \& \ \neg q)$.

Доказательство.

- (1) $\vdash (p \rightarrow \neg(\neg p \ \& \ \neg q)) \rightarrow ((q \rightarrow \neg(\neg p \ \& \ \neg q)) \rightarrow$
 $(p \vee q \rightarrow \neg(\neg p \ \& \ \neg q))), \Pi_r^{\neg(\neg p \ \& \ \neg q)}(\text{A8});$
- (2) $\vdash \neg p \ \& \ \neg q \rightarrow \neg p, \Pi_{p \ q}^{\neg p \neg q}(\text{A3});$
- (3) $\vdash \neg p \ \& \ \neg q \rightarrow \neg q, \Pi_{p \ q}^{\neg p \neg q}(\text{A4});$
- (4) $\vdash \neg\neg p \rightarrow \neg(\neg p \ \& \ \neg q), \text{ПВ5}(2), \text{контрапозиция};$
- (5) $\vdash \neg\neg q \rightarrow \neg(\neg p \ \& \ \neg q), \text{ПВ5}(3), \text{контрапозиция};$
- (6) $\vdash p \rightarrow \neg(\neg p \ \& \ \neg q), \text{ПВ7}(4), \text{снятие двойного отрицания};$
- (7) $\vdash q \rightarrow \neg(\neg p \ \& \ \neg q), \text{ПВ7}(5), \text{снятие двойного отрицания};$

(8) $\vdash (q \rightarrow \neg(\neg p \ \& \ \neg q)) \rightarrow (p \vee q \rightarrow \neg(\neg p \ \& \ \neg q))$, ПЗ(1,6);

(9) $\vdash p \vee q \rightarrow \neg(\neg p \ \& \ \neg q)$, ПЗ(7,8).

Теорема 9. $\vdash \neg p \ \& \ \neg q \rightarrow \neg(p \vee q)$.

Доказательство.

(1) $\vdash p \vee q \rightarrow \neg(\neg p \ \& \ \neg q)$, Т8.

(2) $\vdash \neg\neg(\neg p \ \& \ \neg q) \rightarrow \neg(p \vee q)$, ПВ5(1), контрапозиция.

(3) $\vdash \neg p \ \& \ \neg q \rightarrow \neg(p \vee q)$, ПВ7(2), снятие двойного отрицания.

Теорема 10. $\vdash (p \rightarrow q) \rightarrow \neg p \vee q$.

Доказательство.

(1) $p, p \rightarrow q \vdash q$, правило заключения;

(2) $\vdash q \rightarrow \neg p \vee q$, $\Pi_p^{\neg p}(A7)$;

(3) $p, p \rightarrow q \vdash \neg p \vee q$, ПЗ(1,2);

(4) $\vdash p \rightarrow ((p \rightarrow q) \rightarrow \neg p \vee q)$, ТД(3);

(5) $\vdash \neg p \rightarrow \neg p \vee q$, $\Pi_p^{\neg p}(A6)$;

(6) $\vdash (p \rightarrow q) \rightarrow (\neg p \rightarrow \neg p \vee q)$, ПВ3(5);

(7) $\vdash \neg p \rightarrow ((p \rightarrow q) \rightarrow \neg p \vee q)$, ПВ13(6), перестановка посылок;

(8) $\vdash (p \vee \neg p) \rightarrow ((p \rightarrow q) \rightarrow \neg p \vee q)$, ПВ7(4,7), введение дизъюнкции;

(9) $\vdash p \vee \neg p$, Т7;

(10) $\vdash (p \rightarrow q) \rightarrow \neg p \vee q$, ПЗ(8,9).



5.4. Тождественно истинные и доказуемые формулы

Пусть $A(p_1, p_2, \dots, p_n)$ есть формула в ИВ и $p_1, p_2, \dots, p_n$ есть полный список ее переменных; $a = (a_1, a_2, \dots, a_n)$ есть набор из 0 и 1 длины n ; $P_a(A)$ означает значение формулы A на наборе a . Определим это понятие индукцией по построению формулы.

1. Если A есть переменная p_i , то

$$P_a(A) = P_{a_1, a_2, \dots, a_i, \dots, a_n}(p_i) = a_i.$$

2. Если A есть формула $B * C$ (где $*$ есть один из знаков $\&$, $\vee$, $\rightarrow$) или же формула $\neg B$, то $P_a(B * C) = P_a(B) * P_a(C)$; $P_a(\neg B) = \neg P_a(B)$.

Примем обозначения $p^1 = p$; $p^0 = \neg p$.

Лемма (о доказуемости формулы по значению на наборе).

$$1. \frac{P_a(A) = 1}{p_1^{a_1}, p_2^{a_2}, \dots, p_n^{a_n} \vdash A}.$$

$$2. \frac{P_a(A) = 0}{p_1^{a_1}, p_2^{a_2}, \dots, p_n^{a_n} \vdash \neg A}.$$

Доказательство. Индукция по построению формулы.

Пусть совокупность формул $\Gamma = \{p_1^{a_1}, p_2^{a_2}, \dots, p_n^{a_n}\}$.

Базис. Формула A есть переменная p_i . Тогда $P_a(p_i) = a_i$.

а) $a_i = 1$. Тогда $p_i^{a_i} = p_i = A$; откуда $\Gamma \vdash A$.

б) $a_i = 0$. Тогда $p_i^{a_i} = \neg p_i = \neg A$; откуда $\Gamma \vdash \neg A$.



Предположение индукции. Предположим, что лемма справедлива для всех формул глубины построения меньше k .

Шаг индукции. Покажем, что лемма справедлива для формул глубины построения k . Возможны следующие случаи.

а) A есть $B \& C$. Возможны следующие подслучаи.

1. $P_a(A) = 1$. Покажем, что $\Gamma \vdash A$. Так как $P_a(A) = P_a(B \& C) = P_a(B) \& P_a(C) = 1$, то $P_a(B) = 1, P_a(C) = 1$. Так как глубина построения формулы B и формулы C меньше k , то по предположению индукции имеем $\Gamma \vdash B, \Gamma \vdash C$, откуда $\Gamma \vdash B \& C$, ПВ2.

2. $P_a(A) = 0$. Покажем, что $\Gamma \vdash \neg A$. Так как $P_a(A) = P_a(B \& C) = P_a(B) \& P_a(C) = 0$, то $P_a(B) = 0$ или $P_a(C) = 0$. Пусть для определенности $P_a(B) = 0$. Так как глубина построения формулы B меньше k , то по предположению индукции

(1) $\Gamma \vdash \neg B$. Далее получаем

(2) $\vdash B \& C \rightarrow B, \Pi_{pq}^{BC}(A3);$

(3) $\vdash \neg B \rightarrow \neg(B \& C)$, ПВ5(2), контрапозиция;

(4) $\Gamma \vdash \neg(B \& C)$, ПЗ(1,3), т. е. $\Gamma \vdash \neg A$.

б) A есть $B \vee C$. Возможны следующие подслучаи.

1. $P_a(A) = 1$. Покажем, что $\Gamma \vdash A$. Так как $P_a(A) = P_a(B \vee C) = P_a(B) \vee P_a(C) = 1$, то $P_a(B) = 1$ или $P_a(C) = 1$. Пусть для определенности имеем $P_a(B) = 1$. Так как глубина построения формулы B меньше k , то по предположению индукции

(1) $\Gamma \vdash B$. Далее получаем

(2) $\vdash B \rightarrow B \vee C, \Pi_{pq}^{BC}(A6);$

(3) $\Gamma \vdash B \vee C$, ПЗ(1,2), т. е. $\Gamma \vdash A$.

2. $P_a(A) = 0$. Покажем, что $\Gamma \vdash \neg A$. Так как $P_a(A) = P_a(B \vee C) = P_a(B) \vee P_a(C) = 0$, то $P_a(B) = 0, P_a(C) = 0$. Так как глубина построения формулы B и формулы C меньше k , то по предположению индукции

(1) $\Gamma \vdash \neg B$;

(2) $\Gamma \vdash \neg C$. Далее получаем

(3) $\Gamma \vdash \neg B \& \neg C$, ПВ2(1,2);

(4) $\vdash \neg B \& \neg C \rightarrow \neg(B \vee C), \Pi_{pq}^{BC}(T9);$

(5) $\Gamma \vdash \neg(B \vee C)$, ПЗ(3,4), т. е. $\Gamma \vdash \neg A$.

с) A есть $B \rightarrow C$. Возможны следующие подслучаи.

1. $P_a(A) = 1$. Покажем, что $\Gamma \vdash A$. Так как $P_a(A) = P_a(B \rightarrow C) = P_a(B) \rightarrow P_a(C) = 1$, то $P_a(B) = 0$ или $P_a(C) = 1$.

Пусть $P_a(B) = 0$. Так как глубина построения формулы B меньше k , то по предположению индукции

(1) $\Gamma \vdash \neg B$. Далее

(2) $\Gamma \vdash B \& \neg B \rightarrow C, \Pi_{pq}^{BC}(T6);$

(3) $\Gamma \vdash B \rightarrow (\neg B \rightarrow C)$, ПВ14(2), разъединение посылок;

(4) $\Gamma \vdash \neg B \rightarrow (B \rightarrow C)$, ПВ13(3), перестановка посылок;

(5) $\Gamma \vdash B \rightarrow C$, ПЗ(1,4), т. е. $\Gamma \vdash A$.

Пусть $P_a(C) = 1$. Так как глубина построения формулы C меньше k , то по предположению индукции

(1) $\Gamma \vdash C$. Далее

(2) $\vdash C \rightarrow (B \rightarrow C), \Pi_{p,q}^{B \rightarrow C}(A1);$

(3) $\Gamma \vdash B \rightarrow C, \text{ПЗ}(1,2), \text{т. е. } \Gamma \vdash A.$

2. $P_a(A) = 0$. Покажем, что $\Gamma \vdash \neg A$. Так как $P_a(A) = P_a(B \rightarrow C) = P_a(B) \rightarrow P_a(C) = 0$, то $P_a(B) = 1, P_a(C) = 0$. Так как глубина построения формулы B и формулы C меньше k , то по предположению индукции

(1) $\Gamma \vdash B;$

(2) $\Gamma \vdash \neg C$. Далее

(3) $\Gamma \vdash ((B \rightarrow C) \rightarrow C) \rightarrow (\neg C \rightarrow \neg(B \rightarrow C)), \Pi_{p,q}^{B \rightarrow C \rightarrow C}(A9);$

(4) $\vdash (B \rightarrow C) \rightarrow (B \rightarrow C), \Pi_p^{B \rightarrow C}(T1);$

(5) $\vdash B \rightarrow ((B \rightarrow C) \rightarrow C), \text{ПВ13}(4), \text{перестановка посылок};$

(6) $\Gamma \vdash (B \rightarrow C) \rightarrow C, \text{ПЗ}(1,5);$

(7) $\Gamma \vdash \neg C \rightarrow \neg(B \rightarrow C), \text{ПВ5}(6), \text{контрапозиция};$

(8) $\Gamma \vdash \neg(B \rightarrow C), \text{ПЗ}(1,7), \text{т. е. } \Gamma \vdash \neg A.$

д) A есть $\neg B$. Возможны следующие случаи.

1. $P_a(A) = 1$. Покажем, что $\Gamma \vdash A$. Так как $P_a(A) = P_a(\neg B) = \neg P_a(B) = 1$, то $P_a(B) = 0$. Так как глубина построения формулы B меньше k , то по предположению индукции $\Gamma \vdash \neg B$, т. е. $\Gamma \vdash A$.

2. $P_a(A) = 0$. Покажем, что $\Gamma \vdash \neg A$. Как и выше, $P_a(B) = 1$. По предположению индукции

(1) $\Gamma \vdash B$. Далее

(2) $\vdash B \rightarrow \neg\neg B, \Pi_p^B(A10);$

(3) $\Gamma \vdash \neg\neg B$, т. е. $\Gamma \vdash \neg A$.

Лемма доказана.

Теорема. Всякая тождественно истинная формула доказуема в ИВ.

Доказательство. Пусть $A(p_1, p_2, \dots, p_n)$ есть тождественно истинная формула и $p_1, p_2, \dots, p_n$ есть полный список ее переменных; $a = (a_1, a_2, \dots, a_n)$ есть произвольный набор из 0 и 1 длины n . Так как $P_a(A) = 1$ для любого набора a , в том числе и для наборов $(a_1, \dots, a_{n-1}, 1)$ и $(a_1, \dots, a_{n-1}, 0)$, то по доказанной выше лемме

(1) $p_1^{a_1}, p_2^{a_2}, \dots, p_{n-1}^{a_{n-1}}, p_n \vdash A.$

(2) $p_1^{a_1}, p_2^{a_2}, \dots, p_{n-1}^{a_{n-1}}, \neg p_n \vdash A$. Далее

(3) $p_1^{a_1}, p_2^{a_2}, \dots, p_{n-1}^{a_{n-1}}, p_n \rightarrow A, \text{ТД}(1);$

(4) $p_1^{a_1}, p_2^{a_2}, \dots, p_{n-1}^{a_{n-1}}, \neg p_n \rightarrow A, \text{ТД}(2);$

(5) $p_1^{a_1}, p_2^{a_2}, \dots, p_{n-1}^{a_{n-1}}, p_n \vee \neg p_n \rightarrow A, \text{ПВ17}(3,4), \text{введение } \vee;$

(6) $\vdash p_n \vee \neg p_n, \Pi_p^{p_n}(T7);$

(7) $p_1^{a_1}, p_2^{a_2}, \dots, p_{n-1}^{a_{n-1}} \vdash A. \text{ПЗ}(5,6).$

Повторяя предыдущие рассуждения, получим

$p_1^{a_1}, p_2^{a_2}, \dots, p_{n-2}^{a_{n-2}} \vdash A. \text{ПЗ}(5,6).$

После конечного числа подобных рассуждений получим $\vdash A$.

Теорема. Всякая доказуемая в ИВ формула общезначима.

Доказательство. Индукция по длине k доказательства.

Базис. $k = 1$. Непосредственной проверкой убеждаемся, что все аксиомы (только они имеют длину доказательства $k = 1$) тождественно истинны.

Предположение индукции. Допустим, что все доказуемые формулы с длиной доказательства меньше k тождественно истинны.

Шаг индукции. Покажем, что все доказуемые в ИВ формулы с длиной доказательства k тождественно истинны. Пусть формула A доказуема в ИВ и ее доказательство $A_1, A_2, \dots, A_k (=A)$ имеет длину k . Возможны следующие случаи.

1. A есть аксиома. Тогда A тождественно истинна.

2. Формула $A_k (=A)$ есть $P_r^c(A_i(p))$, т. е. получена из формулы $A_i(p)$ в доказательстве $A_1, A_2, \dots, A_i(p), \dots, A_k$ с помощью подстановки. Так как длина доказательства $A_1, A_2, \dots, A_i(p)$ формулы $A_i(p)$ меньше k , то по предположению индукции формула $A_i(p)$, а вместе с ней и формула $A = A_i(C)$ тождественно истинны.

3. Формула A_k получена из формул A_i и $A_i \rightarrow A_k$ в доказательстве $A_1, \dots, A_i, \dots, A_i \rightarrow A_k, \dots, A_k (=A)$ с помощью правила заключения. Так как длины доказательств $A_1, \dots, A_i$ и $A_1, \dots, A_i, \dots, A_i \rightarrow A_k$ формул A_i и $A_i \rightarrow A_k$ меньше k , то по предположению индукции формулы A_i и $A_i \rightarrow A_k$ тождественно истинны. Тогда формула A_k тождественно истинна по определению операции импликации.

Из двух предыдущих теорем получаем следующее.

Теорема. Чтобы формула A была доказуема в ИВ, необходимо и достаточно, чтобы формула A была тождественно истинной.

5.5. Разрешимость, непротиворечивость, полнота, независимость аксиом

Проблема разрешимости для любого аксиоматического исчисления, в том числе и для исчисления высказываний, состоит в существовании алгоритма, который по любой формуле устанавливает, является она в этом исчислении доказуемой или не является. Если такого алгоритма не существует, то аксиоматическое исчисление алгоритмически неразрешимо.

Исчисление высказываний алгоритмически разрешимо. Разрешающий алгоритм состоит в проверке для данной формулы, является ли она тождественно истинной. Если да, то формула доказуема в ИВ; если нет, то формула в ИВ не доказуема.

Исчисление высказываний *полно относительно интерпретации*, т. е. в ИВ доказуемы все тождественно истинные формулы (и только они).

Определение. Аксиоматическое исчисление *непротиворечиво*, если в нем нет формулы A , для которой $\vdash A$ и $\vdash \neg A$ одновременно.

Исчисление высказываний непротиворечиво, ибо если допустить существование в ИВ формулы A , для которой $\vdash A$ и $\vdash \neg A$ одновременно, то получим, что формулы A и $\neg A$ тождественно истинны, что невозможно.

Определение. Аксиоматическое исчисление *внутренне полно*, если добавление к нему недоказуемой формулы в качестве аксиомы приводит к противоречивому исчислению.

Теорема. Исчисление высказываний внутренне полно.

Доказательство. Пусть $A(p_1, p_2, \dots, p_n)$ есть недоказуемая в ИВ формула и $p_1, p_2, \dots, p_n$ есть полный список переменных этой формулы. Добавим форму-

лу A к исчислению высказываний в качестве аксиомы; получим исчисление K . Тогда $\vdash_K A$ (т. е. A доказуема в K). Формула A не является тождественно истинной, иначе она была бы доказуема в ИВ. Поэтому существует набор $a = (a_1, a_2, \dots, a_n)$ из 0 и 1, для которого $P_a(A) = 0$. Зададим формулы

$$B_i = \begin{cases} p_i \vee \neg p_i, & \text{если } a_i = 1, \\ p_i \& \neg p_i, & \text{если } a_i = 0, \end{cases} \quad i = 1, 2, \dots, n.$$

Формула $C = A(B_1, B_2, \dots, B_n)$ тождественно ложна; тогда формула $\neg C$ тождественно истинна, и потому $\vdash \neg C$, а следовательно, $\vdash_K \neg C$. Так как $\vdash_K A(p_1, p_2, \dots, p_n)$, то $\vdash_K A(B_1, B_2, \dots, B_n)$ как подстановка в доказуемую формулу, т. е. $\vdash_K C$ и $\vdash_K \neg C$, откуда следует, что исчисление K противоречиво. Теорема доказана.

Заметим, что в противоречивом исчислении доказуема любая формула (и потому в смысле выводимости противоречивое исчисление тривиально). В самом деле, пусть B есть произвольная формула в аксиоматическом исчислении (которое противоречиво) и A есть такая формула в нем, что $\vdash A$ и $\vdash \neg A$ одновременно. Тогда $\vdash A \& \neg A$, и ввиду $\vdash A \& \neg A \rightarrow B$ по правилу заключения получаем $\vdash B$ для любой формулы B .

Определение. Независимой в системе аксиом называется аксиома, не выводимая из остальных аксиом.

Теорема. Все аксиомы исчисления высказываний независимы.

Доказательство. Покажем, что аксиома $A3 \ p \& q \rightarrow p$ независима от остальных аксиом. Пусть L есть система аксиом, включающая в себя все аксиомы ИВ, кроме $A3$. Определим дизъюнкцию, импликацию, отрицание обычным образом, а конъюнкцию определим, положив $p \& q = q$. При таком определении конъюнкции аксиома $A3 \ p \& q \rightarrow p$ на наборе $(0,1)$ принимает значение 0; остальные аксиомы тождественно равны 1.

Утверждение. Все формулы, выводимые из системы аксиом L при вышеопределенных логических операциях, тождественно равны 1.

Доказательство. Индукция по длине k вывода.

Базис. $k = 1$. Непосредственной проверкой убеждаемся, что все аксиомы из L при вышеопределенных логических операциях тождественно равны 1.

Предположение индукции. Допустим, что все формулы, выводимые из L с длиной вывода меньше k , тождественно равны 1.

Шаг индукции. Покажем, что всякая формула, выводимая из L с длиной вывода k , тождественно равна 1 (при вышеопределенных логических операциях). Пусть $\vdash_L A$ и $A_1, A_2, \dots, A_k$ есть доказательство длины k в системе L формулы A . Возможны следующие случаи.

1. A_k есть аксиома из L . Тогда A_k тождественно равна 1 (случай базиса).
2. A_k получена подстановкой формулы C в $A_i(p)$ в доказательстве $A_1, A_2, \dots, A_i(p), \dots, A_k$, т. е. $A_k = A_i(C)$. Так как длина доказательства формулы $A_i(p)$ меньше k , то по предположению индукции формула $A_i(p)$ тождественно равна 1, и потому формула $A_i(C)$ тождественно равна 1.
3. Формула A_k получена из формул A_i и $A_i \rightarrow A_k$ в доказательстве $A_1, \dots, A_i, \dots, A_i \rightarrow A_k, \dots, A_k$ с помощью правила заключения.

Так как длины доказательств $A_1, \dots, A_k$ и $A_1, \dots, A_i, \dots, A_i \rightarrow A_k$ формул A_i и $A_i \rightarrow A_k$ меньше k , то по предположению индукции формулы A_i и $A_i \rightarrow A_k$ тождественно равны 1. Так как операция $\rightarrow$ определялась обычным образом, то формула A_k тождественно равна 1.

Утверждение доказано. Продолжим доказательство теоремы.

Допустим, что аксиома А3 $p \& q \rightarrow p$ выводима из остальных аксиом. Тогда по утверждению она тождественно равна 1 (при вышеопределенных логических операциях). Противоречие. Следовательно, аксиома А3 независима от остальных аксиом.

Независимость остальных аксиом системы П. С. Новикова доказывается по такой же схеме.

В табл. 5.1 приводятся все аксиомы и для каждой аксиомы А определяется специальная логическая операция, для которой все аксиомы, кроме аксиомы А, общезначимы; аксиома А на приводимых значениях ее переменных общезначимой не является; правила вывода сохраняют общезначимость получающихся формул.

Таблица 5.1

	Аксиома и множество для интерпретации	Специальная логическая операция	Значения переменных	Истинность
A1	$A \rightarrow (B \rightarrow A)$ {0,1,2,3}	$A \rightarrow A = 1, A \rightarrow 1 = 1, 0 \rightarrow A = 1$ $A \& B = B \& A, A \& 1 = A, A \& 0 = 0$ $A \vee B = B \vee A, A \vee 1 = 1, A \vee 0 = A$ $\neg 1 = 0, \neg 0 = 1$ $A \& A = A, A \vee A = A$ $1 \rightarrow 0 = 0, 1 \rightarrow 2 = 0, 1 \rightarrow 3 = 0$ $2 \rightarrow 0 = 0, 2 \rightarrow 3 = 0, 2 \& 3 = 3$ $3 \rightarrow 0 = 0, 3 \rightarrow 2 = 1, 2 \vee 3 = 2$	$A = 3, B = 1$	1
A2	$(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$ {0,1,2}	$A \rightarrow A = 1, A \rightarrow 1 = 1, 0 \rightarrow A = 1$ $A \& B = B \& A, A \& 1 = A, A \& 0 = 0$ $A \vee B = B \vee A, A \vee 1 = 1, A \vee 0 = A$ $\neg 1 = 0, \neg 0 = 1$ $A \& A = A, A \vee A = A$ $1 \rightarrow 0 = 0, 1 \rightarrow 2 = 2, 2 \rightarrow 0 = 2$ $\neg 2 = 2$	$A = 2, B = 2, C = 0$	1
A3	$A \& B \rightarrow A, \{0,1\}$	$A \& B = B$	$A = 0, B = 1$	1
A4	$A \& B \rightarrow B, \{0,1\}$	$A \& B = A$	$A = 1, B = 0$	1
A5	$(A \rightarrow B) \rightarrow ((A \rightarrow C) \rightarrow (A \rightarrow B \& C)), \{0,1\}$	$A \& B = 0$	$A = 1, B = 1, C = 1$	1
A6	$A \rightarrow A \vee B, \{0,1\}$	$A \vee B = B$	$A = 1, B = 0$	1
A7	$B \rightarrow A \vee B, \{0,1\}$	$A \vee B = A$	$A = 0, B = 1$	1
A8	$(A \rightarrow C) \rightarrow ((B \rightarrow C) \rightarrow (A \vee B \rightarrow C)), \{0,1\}$	$A \vee B = 1$	$A = 0, B = 0, C = 0$	1
A9	$(A \rightarrow B) \rightarrow (\neg B \rightarrow \neg A)$	$\neg A = A$	$A = 0, B = 1$	1
A10	$A \rightarrow \neg \neg A, \{0,1\}$	$\neg A = 0$	$A = 1$	1
A11	$\neg \neg A \rightarrow A, \{0,1\}$	$\neg A = 1$	$A = 0$	1

5.6. Формулировка исчисления высказываний с единственным правилом вывода – правилом заключения

Построим исчисление высказываний ИВ1 без правила подстановки, более удобное для исследования. Формальные символы и формулы ИВ1 совпадают с аналогичными понятиями в ИВ. Пусть A, B, C есть произвольные формулы в ИВ. Введем следующие схемы аксиом.

- A1) $A \rightarrow (B \rightarrow A)$.
- A2) $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$.
- A3) $A \& B \rightarrow A$.
- A4) $A \& B \rightarrow B$.
- A5) $(A \rightarrow B) \rightarrow ((A \rightarrow C) \rightarrow (A \rightarrow B \& C))$.
- A6) $A \rightarrow A \vee B$.
- A7) $B \rightarrow A \vee B$.
- A8) $(A \rightarrow C) \rightarrow ((B \rightarrow C) \rightarrow (A \vee B \rightarrow C))$.
- A9) $(A \rightarrow B) \rightarrow (\neg B \rightarrow \neg A)$.
- A10) $A \rightarrow \neg\neg A$.
- A11) $\neg\neg A \rightarrow A$.

Единственным правилом вывода является правило заключения.

Теорема. Классы формул, доказуемых в ИВ и в ИВ1, совпадают.

Доказательство. Покажем, что класс формул, доказуемых в ИВ1, содержится в классе формул, доказуемых в ИВ. В самом деле, пусть $\vdash_1 A$ (т. е. формула A доказуема в ИВ1) и $A_1, A_2, \dots, A_k (=A)$ есть доказательство формулы A в ИВ1. Так как все A_i тождественно истинны, то $\vdash A_i$. Каждое вхождение формулы A_i в доказательстве формулы A в ИВ1 заменим на доказательство формулы A_i в ИВ. Полученная последовательность формул будет доказательством формулы A в ИВ.

Покажем теперь индукцией по длине k доказательства формулы в ИВ, что класс формул, доказуемых в ИВ, содержится в классе формул, доказуемых в ИВ1.

Базис. $k = 1$. Всякая аксиома в ИВ (только аксиомы имеют длину доказательства 1) доказуема в ИВ1, ибо всякая аксиома в ИВ является аксиомой и в ИВ1.

Предположение индукции. Пусть все доказуемые в ИВ формулы с длиной доказательства меньше k доказуемы в ИВ1.

Шаг индукции. Покажем, что все доказуемые в ИВ формулы с длиной доказательства k доказуемы в ИВ1. Пусть $\vdash A$ и $A_1, A_2, \dots, A_k (=A)$ есть доказательство длины k формулы A в ИВ. Возможны следующие случаи.

1. A_k есть аксиома в ИВ. Тогда A_k есть аксиома в ИВ1, и потому $\vdash_1 A_k$.
2. A_k получена подстановкой формулы C в формулу $A_i(p)$ в доказательстве $A_1, A_2, \dots, A_i(p), \dots, A_k$, т. е. $A_k = A_i(C)$. Так как длина доказательства формулы $A_i(p)$ в ИВ меньше k , то по предположению индукции формула $\vdash_1 A_i(p)$. В доказатель-

стве $B_1, B_2, \dots, B_m(p)$ ($=A_i(p)$) формулы $A_i(p)$ в ИВ1 заменим все вхождения p на формулу C . Полученная последовательность будет доказательством формулы A в ИВ1.

3. Формула A_k получена из формул A_i и $A_i \rightarrow A_k$ в доказательстве $A_1, \dots, A_i, \dots, A_i \rightarrow A_k, \dots, A_k$ с помощью правила заключения. Так как длины доказательств $A_1, \dots, A_i$ и $A_1, \dots, A_i, \dots, A_i \rightarrow A_k$ формул A_i и $A_i \rightarrow A_k$ в ИВ меньше k , то по предположению индукции $\vdash_1 A_i$ и $\vdash_1 A_i \rightarrow A_k$; откуда по правилу заключения $\vdash_1 A_k$. Теорема доказана.



Глава 6

Логика предикатов

6.1. Предикаты, кванторы



Предикат есть функция, определенная на некотором множестве и принимающая одно из двух значений: истина (И) или ложь (Л).

Из простых предикатов с помощью логических операций можно строить и более сложные предикаты.

Пусть $P(x_1, \dots, x_k)$ есть предикат, определенный на множестве D ; $\lceil P \rceil$ означает множество истинности предиката P , т. е. множество всех наборов $(a_1, \dots, a_k)$ длины k элементов множества D , на которых предикат P принимает значение И.

Пусть $P(x_1, \dots, x_k)$ и $R(x_1, \dots, x_k)$ есть два предиката, определенных на множестве D . Тогда $\lceil P \rceil \cup \lceil R \rceil$, $\lceil P \rceil \cap \lceil R \rceil$, $D^k - \lceil P \rceil$ есть множества истинности предикатов $P \vee R$, $P \& R$, $\neg P$ соответственно.

Пусть $P(x, x_1, \dots, x_k)$ есть $(k+1)$ -местный предикат, определенный на множестве D . Запись $(\exists x)P(x, x_1, \dots, x_k)$ будем понимать как «существует такой элемент x из D , для которого $P(x, x_1, \dots, x_k)$ истинно»; $(\forall x)P(x, x_1, \dots, x_k)$ есть «для всех x из D $P(x, x_1, \dots, x_k)$ истинно». Скажем, что предикаты

$$Q(x_1, \dots, x_k) = (\exists x)P(x, x_1, \dots, x_k),$$

$$R(x_1, \dots, x_k) = (\forall x)P(x, x_1, \dots, x_k)$$

получены из предиката $P(x, x_1, \dots, x_k)$ навешиванием *квантора* соответственно существования и общности на предикат $P(x, x_1, \dots, x_k)$ по переменной x . Переменная x в выражении $(Qx)P(x, x_1, \dots, x_k)$ находится в области действия квантора (Qx) . Навешивание квантора существования или квантора общности по некоторой переменной на $(k+1)$ -местный предикат, содержащий эту переменную, приводит к другому предикату, местность которого на единицу меньше.

Переменная x , находящаяся в области действия квантора (Qx) , называется связанной. В противном случае переменная x свободна.

Пример. $D = \{0, 1, 2, \dots\}$ есть множество натуральных чисел. Определим на множестве D следующие предикаты:

- $Pr(x)$, x есть простое число;
- $Ev(x)$, число x четно;

- $Div(x, y)$, число x делит число y ;
- $x = y$, число x равно числу y ;
- $x \leq y$, число x меньше или равно числу y ;
- $x = 2$, число x совпадает с двойкой;
- $Q(x, y)$ есть $\neg(x = 1) \ \& \ \neg(x = y) \ \& \ Div(x, y)$;
- $P(y)$ есть $(\exists x)Q(x, y)$.

Тогда $(\exists x)Pr(x)$ есть истинное, $(\forall x)Pr(x)$ ложное, $P(2)$ ложное, $P(4)$ истинное высказывания.

Пусть $D = \{a_1, a_2, \dots, a_k\}$ есть конечное множество из k элементов и $P(x)$ есть предикат, определенный на D .

Утверждение. Справедливы следующие равенства:

$$(\exists x)P(x) = P(a_1) \vee P(a_2) \vee \dots \vee P(a_k);$$

$$(\forall x)P(x) = P(a_1) \ \& \ P(a_2) \ \& \ \dots \ \& \ P(a_k).$$

Доказательство. Установим первое равенство. Пусть $(\exists x)P(x)$ истинно. Тогда существует $x = a_i \in D$, для которого $P(a_i)$ истинно. Отсюда $P(a_1) \vee \dots \vee P(a_k)$ истинно.

Пусть $P(a_1) \vee \dots \vee P(a_k)$ истинно. Тогда для некоторого i , $1 \leq i \leq k$, $P(a_i)$ истинно; поэтому существует $x = a_i \in D$, для которого $P(a_i)$ истинно, отсюда $(\exists x)P(x)$ истинно.

Установим второе равенство. Пусть $(\forall x)P(x)$ истинно. Тогда для всякого x из D $P(x)$ истинно, т. е. истинны $P(a_1), \dots, P(a_k)$, откуда истинно $P(a_1) \ \& \ \dots \ \& \ P(a_k)$. Пусть теперь $P(a_1) \ \& \ \dots \ \& \ P(a_k)$ истинно, т. е. истинны $P(a_1), \dots, P(a_k)$. Отсюда получаем, что для всякого x из D $P(x)$ истинно, т. е. $(\forall x)P(x)$ истинно.

Утверждение доказано.

Аналогично доказываются следующие равенства:

$$(\exists x)P(x, x_1, \dots, x_k) = \bigvee_{i=1}^k P(a_i, x_1, \dots, x_k);$$

$$(\forall x)P(x, x_1, \dots, x_k) = \big\&_{i=1}^k P(a_i, x_1, \dots, x_k).$$

Переменная x может стоять на любом аргументном месте предиката P . Если кванторов несколько, то они элиминируются (устраняются) постепенно. На-

пример, $(\forall x)(\exists y)P(x, y) = (\forall x)(\bigvee_{j=1}^k P(x, a_j)) = \big\&_{i=1}^k (\bigvee_{j=1}^k P(a_i, a_j)).$

Замечание. Пусть D есть некоторое множество и множество $K \subseteq D^n$. Проекция K на оси $i_1, \dots, i_p$ есть множество

$$\text{Пр}_{i_1, \dots, i_p} K = \{(x_{i_1}, \dots, x_{i_p}) \in D^p : \exists (x_1, \dots, x_{i_1}, \dots, x_{i_p}, \dots, x_n) \in K\}.$$

Проекция множества K на оси $i_1, \dots, i_p$ может быть получена вычеркиванием (стиранием) во всех наборах $(x_1, \dots, x_{i_1}, \dots, x_{i_p}, \dots, x_n)$ из K всех координат, кроме $x_{i_1}, \dots, x_{i_p}$.

Пусть $P(x_1, \dots, x_k, x)$ есть $(k+1)$ -местный предикат, определенный на множестве D . Пусть k -местный предикат $Q(x_1, \dots, x_k) = (\exists x)P(x_1, \dots, x_k, x)$. Пусть

$$\begin{aligned} \lceil P \rceil &= \{(x_1, \dots, x_k, x) \in D^{k+1} : P(x_1, \dots, x_k, x) \text{ истинно}\} \text{ и} \\ \lceil Q \rceil &= \{(x_1, \dots, x_k) \in D^k : Q(x_1, \dots, x_k) \text{ истинно}\} \end{aligned}$$

есть множества истинности предикатов P и Q . Тогда

$$\lceil Q \rceil = \text{Пр}_{1,2,\dots,k} \lceil P \rceil.$$

Пусть теперь k -местный предикат

$$R(x_1, \dots, x_k) = (\forall x)P(x_1, \dots, x_k, x) = \neg(\exists x)\neg P(x_1, \dots, x_k, x),$$

тогда

$$\lceil R \rceil = D^k - (\text{Пр}_{1,2,\dots,k} (D^{k+1} - \lceil P \rceil)).$$

Таков теоретико-множественный смысл квантора существования и квантора общности.

6.2. Общезначимость, выполнимость, невыполнимость, опровержимость формул логики предикатов

Определим понятие формулы в логике предикатов (ЛП).

Алфавит

1. $x, y, z, x_1, y_1, z_1, \dots$ – символы предметных переменных.
2. $a, b, c, a_1, b_1, c_1, \dots$ – символы предметных констант.
3. $f, g, h, f_1, h_1, g_1, \dots$ – символы функциональных переменных.
4. $P, Q, R, P_1, Q_1, R_1, \dots$ – символы предикатных переменных.
5. $\&, -, \rightarrow, \vdash, \exists, \forall$ – логические символы (связки).
6. $(,)$ – скобка левая, запятая, скобка правая.

Термы

1. Всякая предметная переменная и предметная константа есть терм.
2. Если $t_1, \dots, t_k$ есть термы, а f есть функциональная переменная, то $f(t_1, \dots, t_k)$ есть терм.

Если переменные терма t содержатся среди переменных списка $x_1, \dots, x_s$, то пишем $t(x_1, \dots, x_s)$.

Формулы

1. Если $t_1, \dots, t_k$ есть термы; $\{x_1, \dots, x_m\}$ есть множество всех переменных, входящих в термы $t_1, \dots, t_k$; P есть предикатная переменная, то $P(t_1, \dots, t_k)$ есть элементарная формула (атом), а $x_1, \dots, x_m$ есть все ее *свободные* (предметные) переменные.
2. Если A и B есть формулы, то $(A \& B)$, $(A - B)$, $(A \rightarrow B)$ есть формулы. Их свободные переменные есть свободные переменные формул A и B . Выражение $(\neg A)$ – тоже формула; ее свободные переменные есть свободные переменные формулы A .

3. Если формула $A(x)$ имеет множество свободных переменных $\{x, x_1, \dots, x_k\}$, то выражения $(\exists x)A(x)$, $(\forall x)A(x)$ есть формулы; переменные $x_1, \dots, x_k$ в этих формулах свободны, а переменная x *связана* (квантором).

Формула без свободных предметных переменных называется *замкнутой*.

Здесь действуют те же правила опускания скобок, что и в логике высказываний. Кванторы имеют высший приоритет. В формуле $(Qx)A$, где $Q \in \{\exists, \forall\}$, формула A есть область действия квантора (Qx) .

Подформулы

1. Подформулой элементарной формулы является она сама.
2. Подформулами формулы $\neg A$ являются она сама и все подформулы формулы A .
3. Подформулами формулы $A * B$, где знак $*$ $\in \{\&, \neg, \rightarrow\}$, являются она сама и все подформулы формул A и B .
4. Подформулами формулы $(Qx)A$ являются она сама и все подформулы формулы A .

Как и в исчислении высказываний, будем интерпретировать логические связки $\&$, $\neg$, $\rightarrow$, $\rightarrow$ как соответствующие логические операции.

Если предикатные, функциональные, свободные предметные переменные и предметные константы формулы A содержатся в списке $P_1, \dots, P_n, f_1, \dots, f_m, x_1, \dots, x_k, a_1, \dots, a_r$, где $P_1, \dots, P_n$ есть предикатные переменные; $f_1, \dots, f_m$ есть функциональные переменные; $x_1, \dots, x_k$ есть предметные переменные; $a_1, \dots, a_r$ есть предметные константы, то пишем $A(P_1, \dots, P_n, f_1, \dots, f_m, x_1, \dots, x_k, a_1, \dots, a_r)$.

Интерпретация формул. Пусть формула $A = A(P_1, \dots, P_n, f_1, \dots, f_m, x_1, \dots, x_k, a_1, \dots, a_r)$. Набор $(P_1, \dots, P_n, f_1, \dots, f_m, x_1, \dots, x_k, a_1, \dots, a_r)$ называется *сигнатурой* формулы A . Зафиксируем некоторое множество D , зададим на D предикаты $P_1, \dots, P_n$; функции $f_1, \dots, f_m$; предметы $x_1, \dots, x_k, a_1, \dots, a_r$. Набор $I = (D, P_1, \dots, P_n, f_1, \dots, f_m, x_1, \dots, x_k, a_1, \dots, a_r)$ называется *интерпретацией* формулы A . Множество D называется *предметной областью* интерпретации I . Формула A при таком выборе переменных (предикатных, функциональных, предметных) и констант превращается в высказывание $A = A(P_1, \dots, P_n, f_1, \dots, f_m, x_1, \dots, x_k, a_1, \dots, a_r)$, истинное или ложное.

Пример. Пусть формула

$$A(P, Q, R, y, z, a) = (\exists x)(\neg P(x, a) \& \neg P(x, z) \& Q(x, y) \& R(x, y));$$

$D = \{0, 1, 2, \dots\}$ есть множество натуральных чисел;

$P(x, y)$ есть $x = y$;

$Q(x, y)$ есть $x \leq y$;

$R(x, y)$ есть $Div(x, y)$ (x делит y);

$y = 8, z = 4, a = 1$.

Тогда высказывание

$$A = A(P, Q, R, y, z, a) = (\exists x)(\neg P(x, a) \& \neg P(x, z) \& Q(x, y) \& R(x, y)) = (\exists x)(x \neq 1 \& x \neq 4 \& x \leq 8 \& Div(x, 8)) \text{ истинно.}$$

Определение. (Значение $t[x_1, \dots, x_s]$ терма $t(x_1, \dots, x_s)$ на интерпретации I).

1. Если $t = x_i$, то $t[x_1, \dots, x_s] = x_i$.
2. Если $t = a_i$, то $t[x_1, \dots, x_s] = a_i$.
3. Если $t = f(t_1, \dots, t_u)$, то $t[x_1, \dots, x_s] = f(t_1[x_1, \dots, x_s], \dots, t_u[x_1, \dots, x_s])$.

Определение. Интерпретация

$$I = (D, P_1, \dots, P_n, f_1, \dots, f_m, x_1, \dots, x_k, a_1, \dots, a_r)$$

выполняет формулу

$$A = A(P_1, \dots, P_n, f_1, \dots, f_m, x_1, \dots, x_k, a_1, \dots, a_r)$$

(обозначения: $I \models A$, $A(I) = И$, $A(I) = 1$), если высказывание

$$A = A(I) = A(P_1, \dots, P_n, f_1, \dots, f_m, x_1, \dots, x_k, a_1, \dots, a_r)$$

истинно. Интерпретация I *опровергает* формулу A (обозначения: $I \not\models A$, $A(I) = Л$, $A(I) = 0$), если высказывание $A(I)$ ложно. $\square$

Значение формулы A при некоторой интерпретации I определяется индуктивно следующим образом.

1. Если A есть атом $P(t_1, \dots, t_u)$, то $I \models A \leftrightarrow P(t_1[x_1, \dots, x_k], \dots, t_u[x_1, \dots, x_k]) = И$.
2. Если A есть $B * C$, где знак $*$ $\in \{ \&, \vee, \rightarrow \}$, то $I \models A \leftrightarrow (I \models B) * (I \models C)$.
3. Если A есть $\neg B$, то $I \models A \leftrightarrow I \not\models B$.
4. Пусть A есть $(\exists x)B(x)$, и пусть A и $B(x)$ есть

$$A(P_1, \dots, P_n, f_1, \dots, f_m, x_1, \dots, x_k, a_1, \dots, a_r),$$

$$B(P_1, \dots, P_n, f_1, \dots, f_m, x, x_1, \dots, x_k, a_1, \dots, a_r)$$

соответственно. Пусть

$$I_A = (D, P_1, \dots, P_n, f_1, \dots, f_m, x_1, \dots, x_k, a_1, \dots, a_r),$$

$$I_B = (D, P_1, \dots, P_n, f_1, \dots, f_m, x, x_1, \dots, x_k, a_1, \dots, a_r)$$

есть интерпретации формул A и $B(x)$ соответственно. Тогда

$$I_A \models (\exists x)B(x) \leftrightarrow \exists x \in D I_B \models B(x).$$

Аналогично если A есть $(\forall x)B(x)$, то

$$I_A \models (\forall x)B(x) \leftrightarrow \forall x \in D I_B \models B(x).$$

Определение

1. Формула A *общезначима* на множестве D , если всякая интерпретация $I = (D, \dots)$ с предметной областью D выполняет A .
2. Формула A *выполнима* на множестве D , если существует интерпретация $I = (D, \dots)$ с предметной областью D , которая выполняет A .
3. Формула A *опровержима* на множестве D , если существует интерпретация $I = (D, \dots)$ с предметной областью D , которая опровергает A .
4. Формула A *невыполнима* на множестве D , если всякая интерпретация $I = (D, \dots)$ с предметной областью D опровергает A . $\square$

Пример. Пусть формула $A = (\forall x)(\exists y)P(x,y) \rightarrow (\exists y)(\forall x)P(x,y)$.

1. Положим интерпретацию $I = (D, P(x,y)) = (\{1,2\}, x \leq y)$. Высказывание $(\forall x)(\exists y) x \leq y \rightarrow (\exists y)(\forall x) x \leq y$ истинно из-за истинности заключения. Поэтому $I \models A$, т. е. интерпретация I превращает формулу A в истинное высказывание.

2. $I = (\{1,2\}, x = y)$. Высказывание $(\forall x)(\exists y) x = y \rightarrow (\exists y)(\forall x) x = y$ ложно, ибо его посылка истинна, а его заключение ложно. Поэтому $I \not\models A$, т. е. I опровергает A .

Определение.

1. Формула A *общезначима*, если всякая интерпретация I выполняет A .

2. Формула A *выполнима*, если существует интерпретация I , которая выполняет A .

3. Формула A *невыполнима*, если всякая интерпретация I опровергает A .

4. Формула A *опровержима*, если существует интерпретация I , которая опровергает A . $\square$

Заметим, что формулы $A(x_1, \dots, x_k)$ и $(\forall x_1) \dots (\forall x_k) A(x_1, \dots, x_k)$ общезначимы или не общезначимы одновременно. Поэтому при вычислении общезначимости формул можно ограничиться замкнутыми формулами (т. е. формулами без свободных предметных переменных).

Теорема. Чтобы формула логики предикатов была общезначима, необходимо и достаточно, чтобы ее отрицание было невыполнимо.

Доказательство. Пусть формула A общезначима. Допустим, что формула $\neg A$ выполнима. Тогда существует интерпретация I , для которой $I \models \neg A$. Поэтому $I \not\models A$. Противоречие с общезначимостью A .

Пусть теперь формула $\neg A$ невыполнима. Допустим, что формула A общезначимой не является. Тогда существует интерпретация I , которая опровергает A , т. е. $I \not\models A$. Поэтому $I \models \neg A$. Противоречие с невыполнимостью формулы $\neg A$.

Определение. Формула B есть *логическое следствие* формул $A_1, A_2, \dots, A_k$, если формула $A_1 \& A_2 \& \dots \& A_k \rightarrow B$ общезначима. $\square$

Заметим, что если формула логики предикатов не содержит кванторов, то ее можно рассматривать как формулу логики высказываний, т. е. как подстановку $\Pi_{p_1 p_2 \dots p_k}^{B_1 B_2 \dots B_k} (A(p_1, p_2, \dots, p_k))$ формул логики предикатов $B_1, B_2, \dots, B_k$ вместо всех позиционных переменных формул логики высказываний A .

Если предикатные, функциональные, свободные предметные переменные и предметные константы множества формул S содержатся в списке

$$Q_1, \dots, Q_n, g_1, \dots, g_m, x_1, \dots, x_k, a_1, \dots, a_r,$$

где $Q_1, \dots, Q_n$ есть предикатные переменные; $g_1, \dots, g_m$ есть функциональные переменные; $x_1, \dots, x_k$ есть предметные переменные; $a_1, \dots, a_r$ есть предметные константы множества формул S , то пишем

$$S(Q_1, \dots, Q_n, g_1, \dots, g_m, x_1, \dots, x_k, a_1, \dots, a_r).$$

Введем понятие выполнимости и общезначимости множества формул S . Пусть множество S есть

$$S(Q_1, \dots, Q_n, g_1, \dots, g_m, x_1, \dots, x_k, a_1, \dots, a_r).$$



Набор

$$(Q_1, \dots, Q_n, g_1, \dots, g_m, x_1, \dots, x_k, a_1, \dots, a_r)$$

называется *сигнатурой* множества формул S . Зафиксируем некоторое множество D , зададим на D предикаты $Q_1, \dots, Q_n$; функции $g_1, \dots, g_m$; предметы $x_1, \dots, x_k, a_1, \dots, a_r$. Набор

$$I = (D, Q_1, \dots, Q_n, g_1, \dots, g_m, x_1, \dots, x_k, a_1, \dots, a_r)$$

называется *интерпретацией* для S . Множество D называется *предметной областью* интерпретации I . Конъюнкция K_S множества формул S при таком выборе переменных (предикатных, функциональных, предметных) и констант превращается в высказывание

$$K_S = K_S(Q_1, \dots, Q_n, g_1, \dots, g_m, x_1, \dots, x_k, a_1, \dots, a_r),$$

истинное или ложное. Интерпретация $I = (D, Q_1, \dots, Q_n, g_1, \dots, g_m, x_1, \dots, x_k, a_1, \dots, a_r)$ *выполняет* множество формул S (обозначения: $I \models S, S(I) = И, S(I) = 1$), если I выполняет любую формулу из S (то есть $I \models K_S$). Интерпретация I *опровергает* множество формул S (обозначения $I \not\models S, S(I) = Л, S(I) = 0$, если I опровергает хотя бы одну формулу из S (то есть $I \not\models K_S$).

Определение. Интерпретация I есть *модель* для формулы A (для множества формул S), если $I \models A$ (I выполняет всякую формулу из S).

Интерпретация I есть *опровержение* формулы A (множества формул S), если I опровергает A (I опровергает хотя бы одну формулу в S).

6.3. Равносильность формул

Определение. Две формулы логики предикатов называются *равносильными* (эквивалентными, равными), если они принимают одинаковые истинностные значения при всех значениях входящих в эти формулы свободных переменных (множество D ; предикаты и функции, определенные на D ; предметы из D). □

Другими словами, две формулы из ЛП равносильны (эквивалентны), если на всякой интерпретации эти формулы одновременно истинны или ложны. Приведем примеры равносильных формул.

$$1. (\exists x)A(x) \Leftrightarrow (\exists y)A(y).$$

2. $(\forall x)A(x) \Leftrightarrow (\forall y)A(y)$, т. е. возможны переименования связанных предметных переменных.

$$3. (\exists x)(\exists y)A(x, y) \Leftrightarrow (\exists y)(\exists x)A(x, y).$$

4. $(\forall x)(\forall y)A(x, y) \Leftrightarrow (\forall y)(\forall x)A(x, y)$, т. е. соседние одноименные кванторы можно менять местами.

$$5. \neg(\forall x)A(x) \Leftrightarrow (\exists x)\neg A(x).$$

В самом деле, пусть D есть произвольное множество, на котором определены все свободные переменные и константы формулы A , кроме x . Тогда на указанном наборе переменных высказывание $\neg(\forall x)A(x)$ истинно $\forall$ не для всех x из D

высказывание $A(x)$ истинно $\forall$ существует $x = x_0 \in D$, для которого $A(x_0)$ ложно $\forall$ существует $x = x_0 \in D$, для которого $\neg A(x_0)$ истинно $\forall (\exists x) \neg A(x)$ истинно.

$$6. \neg(\exists x)A(x) \Leftrightarrow (\forall x)\neg A(x).$$

В самом деле, $(\forall x)\neg A(x) \Leftrightarrow \neg(\neg(\forall x)\neg A(x)) \Leftrightarrow \neg(\exists x)\neg\neg A(x) \Leftrightarrow \neg(\exists x)A(x)$.

$$7. (\forall x)A(x) \Leftrightarrow \neg(\exists x)\neg A(x).$$

В самом деле, $(\forall x)A(x) \Leftrightarrow \neg(\neg(\forall x)A(x)) \Leftrightarrow \neg(\exists x)\neg A(x)$.

$$8. (\exists x)A(x) \Leftrightarrow \neg(\forall x)\neg A(x).$$

В самом деле, $(\exists x)A(x) \Leftrightarrow \neg(\neg(\exists x)A(x)) \Leftrightarrow \neg(\forall x)\neg A(x)$.

9. $(\exists x)(A(x) \& H) \Leftrightarrow (\exists x)A(x) \& H$, где формула H не содержит переменную x свободно.

В самом деле, $(\exists x)(A(x) \& H)$ истинно $\forall$ существует $x = x_0 \in D$, для которого $A(x_0)$ и H истинны $\forall (\exists x)A(x) \& H$ истинно.

10. $(\forall x)(A(x) \vee H) \Leftrightarrow (\forall x)A(x) \vee H$, где формула H не содержит переменную x свободно.

В самом деле, $(\forall x)(A(x) \vee H) \Leftrightarrow \neg(\neg(\forall x)(A(x) \vee H)) \Leftrightarrow \neg(\exists x)\neg(A(x) \vee H) \Leftrightarrow \neg(\exists x)(\neg A(x) \& \neg H) \Leftrightarrow \neg(\exists x)\neg A(x) \vee \neg\neg H \Leftrightarrow (\forall x)\neg\neg A(x) \vee H \Leftrightarrow (\forall x)A(x) \vee H$.

$$11. (\exists x)(A(x) \vee B(x)) \Leftrightarrow (\exists x)A(x) \vee (\exists x)B(x).$$

В самом деле, $(\exists x)(A(x) \vee B(x))$ истинно $\forall$ существует $x = x_0$, для которого $A(x_0)$ или $B(x_0)$ истинно $\forall (\exists x)A(x) \vee (\exists x)B(x)$ истинно.

$$12. (\forall x)(A(x) \& B(x)) \Leftrightarrow (\forall x)A(x) \& (\forall x)B(x).$$

В самом деле, пусть $C(x) \Leftrightarrow A(x) \& B(x)$. Тогда $(\forall x)C(x) \Leftrightarrow \neg(\neg(\forall x)C(x)) \Leftrightarrow \neg(\exists x)\neg(A(x) \& B(x)) \Leftrightarrow \neg(\exists x)(\neg A(x) \vee \neg B(x)) \Leftrightarrow \neg((\exists x)\neg A(x) \vee (\exists x)\neg B(x)) \Leftrightarrow \neg(\exists x)\neg A(x) \& \neg(\exists x)\neg B(x) \Leftrightarrow (\forall x)A(x) \& (\forall x)B(x)$.

Формулы 11 и 12 утверждают, что квантор существования можно распределять по дизъюнкции, а квантор общности – по конъюнкции. Что касается распределения квантора существования по конъюнкции, а квантора общности по дизъюнкции, то здесь общезначимы формулы

$$(\exists x)(A(x) \& B(x)) \rightarrow (\exists x)A(x) \& (\exists x)B(x);$$

$$(\forall x)A(x) \vee (\forall x)B(x) \rightarrow (\forall x)(A(x) \vee B(x)).$$

Обратные следования неверны.

$$13. (\exists x)A(x) \& (\exists x)B(x) \Leftrightarrow (\exists x)(\exists y)(A(x) \& B(y)).$$

В самом деле, высказывание $(\exists x)A(x) \& (\exists x)B(x)$ истинно $\Leftrightarrow$ существуют элементы x_0 и y_0 , для которых $A(x_0)$ и $B(y_0)$ истинны $\Leftrightarrow (\exists x)(\exists y)(A(x) \& B(y))$ истинно.

$$14. (\forall x)A(x) \vee (\forall x)B(x) \Leftrightarrow (\forall x)(\forall y)(A(x) \vee B(y)).$$

В самом деле, $(\forall x)A(x) \vee (\forall x)B(x) \Leftrightarrow (\forall x)A(x) \vee (\forall y)B(y) \Leftrightarrow (\forall x)(A(x) \vee (\forall y)B(y)) \Leftrightarrow (\forall x)(\forall y)(A(x) \vee B(y))$.

15. $(\forall x)(C \rightarrow B(x)) \Leftrightarrow C \rightarrow (\forall x)B(x)$, где формула C не содержит переменной x свободно.

В самом деле, $(\forall x)(C \rightarrow B(x)) \Leftrightarrow (\forall x)(\neg C \vee B(x)) \Leftrightarrow \neg C \vee (\forall x)B(x) \Leftrightarrow C \rightarrow (\forall x)B(x)$.

16. $(\forall x)(B(x) \rightarrow C) \Leftrightarrow (\exists x)B(x) \rightarrow C$, где формула C не содержит переменной x свободно.

В самом деле, $(\forall x)(B(x) \rightarrow C) \Leftrightarrow (\forall x)(\neg B(x) \vee C) \Leftrightarrow (\forall x)\neg B(x) \vee C \Leftrightarrow \neg(\exists x)B(x) \vee C \Leftrightarrow (\exists x)B(x) \rightarrow C$.

17. $(\exists x)(C \rightarrow B(x)) \Leftrightarrow C \rightarrow (\exists x)B(x)$, где формула C не содержит переменной x свободно.

18. $(\exists x)(B(x) \rightarrow C) \Leftrightarrow (\forall x)B(x) \rightarrow C$, где формула C не содержит переменной x свободно.

Замечание. Если в формулах с 1 по 18 эквивалентность двух формул $A \Leftrightarrow B$ понимать как $(A \rightarrow B) \& (B \rightarrow A)$, то формулы с 1 по 18 станут общезначимыми.

Релятивизованные кванторы

Пусть $A(x)$ и $B(x)$ есть формулы логики предикатов, имеющие свободную переменную x и, возможно, другие свободные переменные. Пусть

$(\exists x)_{A(x)} B(x)$ означает $(\exists x)(A(x) \& B(x))$,
 $(\forall x)_{A(x)} B(x)$ означает $(\forall x)(A(x) \rightarrow B(x))$.

Кванторы $(\exists x)_{A(x)}$ и $(\forall x)_{A(x)}$ называются *релятивизованными*.

Справедливы следующие равносильности.

$$\neg(\exists x)_{A(x)} B(x) \Leftrightarrow (\forall x)_{A(x)} \neg B(x); \neg(\forall x)_{A(x)} B(x) \Leftrightarrow (\exists x)_{A(x)} \neg B(x).$$

В самом деле,

$$\begin{aligned} \neg(\exists x)_{A(x)} B(x) &= \neg(\exists x)(A(x) \& B(x)) = (\forall x)\neg(A(x) \& B(x)) = (\forall x)(\neg A(x) \vee \neg B(x)) = \\ &= (\forall x)(A(x) \rightarrow \neg B(x)) = (\forall x)_{A(x)} \neg B(x); \\ \neg(\forall x)_{A(x)} B(x) &= \neg(\forall x)(A(x) \rightarrow B(x)) = (\exists x)\neg(A(x) \rightarrow B(x)) = (\exists x)(A(x) \& \neg B(x)) = \\ &= (\exists x)_{A(x)} \neg B(x). \end{aligned}$$

6.4. Нормальные формы

6.4.1. Префиксная нормальная форма

Определение. Формула A логики предикатов задана в *префиксной* нормальной форме, если она имеет вид

$$(Q_1 x_1) \dots (Q_k x_k) B(x_1, \dots, x_k), \text{ где } Q_i \in \{\exists, \forall\}, i = 1, 2, \dots, k,$$

а B есть бескванторная формула.

Теорема. Для всякой формулы A в ЛП существует равносильная ей формула в префиксной нормальной форме.

Доказательство. Индукция по построению формулы A . Пусть число логических связок в формуле A равно k .

Базис. $k = 0$. Тогда A есть элементарная формула без кванторов; поэтому формула A находится в префиксной нормальной форме.

Предположение индукции. Допустим, что теорема верна для всякой формулы с числом логических связок меньше k .

Шаг индукции. Покажем, что теорема справедлива для всякой формулы с числом логических связок k . Пусть формула A имеет k логических связок. Возможны следующие случаи.

1. A есть $(Qx)B(x)$, где $Q \in \{\exists, \forall\}$. Так как глубина построения формулы $B(x)$ меньше k , то по предположению индукции формула $B(x)$ представима в префиксной нормальной форме $C(x)$. Тогда $(Qx)C(x)$ есть префиксная нормальная форма формулы A .

2. A есть $B * C$, где знак $*$ $\in \{\&, \vee, \rightarrow\}$. Так как число логических связок в формулах B и C меньше k , то по предположению индукции существуют префиксные нормальные формы E и F для формул B и C соответственно. Пользуясь доказанными в п. 3.3 равносильностями 1–18, выносим в формуле $E * F$ кванторы за скобки и получаем искомую префиксную нормальную форму для формулы A .

3. A есть $\neg B$. Так как число логических связок в формуле B меньше k , то по предположению индукции формула B имеет префиксную нормальную форму $(Q_1x_1)\dots(Q_kx_k)C$, где C есть бескванторная формула. Тогда формула $\neg B$ имеет префиксную нормальную форму $(Q'_1x_1)\dots(Q'_kx_k)\neg C$, где Q'_i есть $\exists$, если Q_i есть $\forall$, и Q'_i есть $\forall$, если Q_i есть $\exists$, $i = 1, 2, \dots, k$.

Пример. $(\forall y)(\neg P(x) \rightarrow Q(y)) \rightarrow (\exists y)(\neg(\forall z)(P(y) \vee Q(z))) =$
 $((\forall y)(\neg P(x) \rightarrow Q(y)) \vee (\exists y)(\exists z)\neg(P(y) \vee Q(z))) =$
 $(\exists y)\neg(\neg P(x) \vee Q(y)) \vee (\exists y)(\exists z)(\neg\neg P(y) \& \neg Q(z)) =$
 $(\exists y)(\neg P(x) \rightarrow Q(y)) \vee (\exists z)(P(y) \& \neg Q(z)) =$
 $(\exists y)(\exists z)(\neg P(x) \rightarrow Q(y)) \vee P(y) \& \neg Q(z)).$

6.4.2. Стандартная форма Сколема

Пусть A есть замкнутая формула логики предикатов. Приведем ее к префиксной нормальной форме

$$A_1 = (Q_1x_1)\dots(Q_kx_k)B_1(x_1, \dots, x_k).$$

Пусть Q_r в формуле A есть первый слева квантор существования, т. е.

$$A_1 = (\forall x_1)\dots(\forall x_{r-1})(\exists x_r)(Q_{r+1}x_{r+1})\dots(Q_kx_k)B_1(x_1, \dots, x_{r-1}, x_r, x_{r+1}, \dots, x_k).$$

Выберем новый, ранее не встречавшийся $(r-1)$ -местный функциональный символ f и перейдем к формуле

$$A_2 = (\forall x_1)\dots(\forall x_{r-1})(Q_{r+1}x_{r+1})\dots(Q_kx_k)\underbrace{B_1(x_1, \dots, x_{r-1}, f(x_1, \dots, x_{r-1}), x_{r+1}, \dots, x_k)}_{B_2(x_1, \dots, x_{r-1}, x_{r+1}, \dots, x_k)}.$$

Если $r = 1$, то в формуле A_1 уберем квантор $(\exists x_1)$ и вместо x_1 , находящегося в области действия квантора $(\exists x_1)$, подставим в A_1 константу a , ранее в A_1 не встречавшуюся.

Утверждение 1. Формула A_1 невыполнима тогда и только тогда, когда A_2 невыполнима.

Доказательство. Пусть формула A_1 невыполнима. Покажем, что формула A_2 невыполнима. Допустим противное: A_2 выполнима, т. е. существует множество D , предикаты и функции, определенные на D , предметы из D – такие, что высказывание

$$(\forall x_1) \dots (\forall x_{r-1}) (Q_{r+1} x_{r+1}) \dots (Q_k x_k) B_l(x_1, \dots, x_{r-1}, f(x_1, \dots, x_{r-1}), x_{r+1}, \dots, x_k)$$

истинно.

Тогда для всяких $x_1, \dots, x_{r-1}$ из D существует $x_r = f(x_1, \dots, x_{r-1})$ из D – такое, что высказывание

$$(Q_{r+1} x_{r+1}) \dots (Q_k x_k) B_l(x_1, \dots, x_{r-1}, x_r, x_{r+1}, \dots, x_k)$$

истинно, то есть высказывание

$$(\forall x_1) \dots (\forall x_{r-1}) (\exists x_r) (Q_{r+1} x_{r+1}) \dots (Q_k x_k) B_l(x_1, \dots, x_{r-1}, x_r, x_{r+1}, \dots, x_k)$$

истинно.

Поэтому формула A_1 выполнима. Противоречие с невыполнимостью A_1 . Следовательно, формула A_2 невыполнима.

Пусть формула A_2 невыполнима. Покажем, что формула A_1 невыполнима. Допустим противное: A_1 выполнима. Тогда существует множество D , предикаты и функции, определенные на D , предметы из D , для которых высказывание

$$(\forall x_1) \dots (\forall x_{r-1}) (\exists x_r) (Q_{r+1} x_{r+1}) \dots (Q_k x_k) B_l(x_1, \dots, x_{r-1}, x_r, x_{r+1}, \dots, x_k)$$

истинно.

Тогда для всяких $x_1, \dots, x_{r-1}$ из D существует x_r из D , т. е. существует функция $x_r = f(x_1, \dots, x_{r-1}) : D^{r-1} \rightarrow D$, для которой высказывание

$$(Q_{r+1} x_{r+1}) \dots (Q_k x_k) B_l(x_1, \dots, x_{r-1}, f(x_1, \dots, x_{r-1}), x_{r+1}, \dots, x_k)$$

истинно. Тогда высказывание

$$(\forall x_1) \dots (\forall x_{r-1}) (Q_{r+1} x_{r+1}) \dots (Q_k x_k) B_l(x_1, \dots, x_{r-1}, f(x_1, \dots, x_{r-1}), x_{r+1}, \dots, x_k)$$

истинно, т. е. формула A_2 выполнима. Противоречие с невыполнимостью A_2 . Следовательно, формула A_2 невыполнима.

Утверждение 1 доказано.

Отметим теперь первое слева вхождение квантора существования в формуле A_2 . Прделаем аналогичное устранение этого квантора. Получим формулу A_3 . В ней проведем устранение первого слева квантора существования. И так далее, пока все кванторы существования не будут устранены (за счет введения новых функциональных символов). В результате получим новую формулу D без кванторов существования, называемую *стандартной формой Сколема*. Функции, возникающие при построении этой стандартной формы, называются *сколемовыми*.

Утверждение 2. Формула A невыполнима тогда и только тогда, когда ее стандартная форма Сколема невыполнима.

Доказательство следует из доказанного выше утверждения 1, справедливого при каждом новом устранении из формулы A квантора существования.

Замечание. 1. Если формула F противоречива и S есть стандартная форма Сколема F , то S (по доказанной выше теореме) также противоречива, т. е. $F = S$.

Если же F непротиворечива, то в общем случае F не эквивалентна S . Например, пусть $F = (\exists x)P(x)$ и $S = P(a)$, т. е. S есть стандартная форма F . Зададим интерпретацию I на области $D = \{1, 2\}$ следующим образом.

$$P(a) = \text{Л}; P(1) = \text{Л}; P(2) = \text{И}.$$

Тогда в интерпретации I формула F будет истинной, а формула S – ложной. Таким образом, $F \neq S$.

2. Формула F может иметь более чем одну стандартную форму Сколема, ибо вид кванторной приставки в префиксной нормальной форме для F зависит от порядка выноса кванторов в кванторную приставку.

Пример. 1. $A = (\exists x_1)(\forall x_2)(\exists x_3)(\forall x_4)(P(x_1, x_2, x_3) \rightarrow P(x_2, x_3, x_4))$.

Вместо x_1 ставим константу a . Получаем формулу

$$(\forall x_2)(\exists x_3)(\forall x_4)(P(a, x_2, x_3) \rightarrow P(x_2, x_3, x_4)).$$

Переменную x_3 заменяем на $f(x_2)$, в результате получаем формулу

$$B = (\forall x_2)(\forall x_4)(P(a, x_2, f(x_2)) \rightarrow P(x_2, f(x_2), x_4)).$$

Формула B есть стандартная форма Сколема формулы A .

2. $A = (\exists x)(\forall y)(\exists u)(\exists v)(\forall z)(\exists w)P(x, y, u, v, z, w)$.

Вместо x ставим константу a . Получаем формулу

$$A_1 = (\forall y)(\exists u)(\exists v)(\forall z)(\exists w)P(a, y, u, v, z, w).$$

Переменную u заменяем на $f(y)$, в результате получаем формулу

$$A_2 = (\forall y)(\exists v)(\forall z)(\exists w)P(a, y, f(y), v, z, w).$$

Переменную v заменяем на $g(y)$, в результате получаем формулу

$$A_3 = (\forall y)(\forall z)(\exists w)P(a, y, f(y), g(y), z, w).$$

Переменную w заменяем на $h(y, z)$, в результате получаем формулу

$$A_4 = (\forall y)(\forall z)P(a, y, f(y), g(y), z, h(y, z)).$$

Формула A_4 не содержит кванторов существования, и потому A_4 есть стандартная форма Сколема формулы A .

Замечание. В стандартной форме Сколема бескванторную ее часть обычно представляют в конъюнктивной нормальной форме и рассматривают как множество дизъюнктивных слагаемых (дизъюнктов). Кванторы общности для этого множества дизъюнктов опускают.

Пример. $F = (\forall x)((\forall y) \neg P(x, y) \ \& \ (\exists z)Q(x, z, u)) \vee (\exists w)R(x, z, w))$.

Приведем формулу A к префиксной нормальной форме.

$$\begin{aligned} F &= (\forall x)((\exists y)(\forall z)(\neg P(x, y) \ \& \ Q(x, z)) \vee (\exists y)(\exists w)R(x, y, w)) = \\ &= (\forall x)(\exists y)((\forall z)(\neg P(x, y) \ \& \ Q(x, z)) \vee (\exists w)R(x, y, w)) = \\ &= (\forall x)(\exists y)(\forall z)(\exists w)((\neg P(x, y) \ \& \ Q(x, z)) \vee R(x, y, w)). \end{aligned}$$

Приведем бескванторную часть полученной формулы (она называется матрицей) к конъюнктивной нормальной форме.

$$F = (\forall x)(\exists y)(\forall z)(\exists w)((\neg(P(x,y) \vee R(x,y,w)) \& (Q(x,z) \vee R(x,y,w))).$$

Далее приведем эту формулу к сколемовой стандартной форме.

$$F_1 = (\forall x)(\forall z)(\exists w)((\neg(P(x,f(x)) \vee R(x,f(x),w)) \& (Q(x,z) \vee R(x,f(x),w))), \\ F_2 = (\forall x)(\forall z)((\neg(P(x,f(x)) \vee R(x,f(x),g(x,z))) \& (Q(x,z) \vee R(x,f(x),g(x,z))))).$$

Формула F_2 есть сколемова стандартная форма формулы F .

Можно представить матрицу стандартной формы множеством дизъюнктов и рассматривать множество дизъюнктов S как конъюнкцию всех дизъюнктов из S , где все переменные в S считаются связанными кванторами всеобщности. Тогда стандартная форма Сколема может быть представлена множеством $\{ \neg(P(x,f(x)) \vee R(x,f(x),g(x,z))), Q(x,z) \vee R(x,f(x),g(x,z)) \}$.

6.5. Проблема разрешимости в логике предикатов

Основной вопрос математической логики есть проблема разрешимости ее формул, т. е. вопрос об установлении общезначимости (или формальной доказуемости) формул этой логики. Проблема разрешимости в ЛП состоит в получении ответа на вопрос: «существует ли алгоритм, который по любой формуле логики предикатов устанавливал бы, является эта формула общезначимой или не является».

Теорема (А. Черч). Проблема распознавания общезначимости формул логики предикатов алгоритмически неразрешима.

Б. А. Трахтенброт установил алгоритмическую неразрешимость ЛП на конечных классах: проблема разрешимости в ЛП остается неразрешимой, если при установлении общезначимости формул ЛП ограничиться конечными множествами.

До конца этого параграфа речь будет идти о формулах в ЛП без функциональных символов.

Хотя и не существует алгоритма, который по любой предъявленной формуле в ЛП устанавливал бы, является эта формула общезначимой или не является, возможны попытки отыскания таких алгоритмов для некоторых классов формул логики предикатов. Были найдены, например, алгоритмы, устанавливающие общезначимость класса замкнутых формул, префиксные нормальные формы которых содержат только кванторы существования. Существует алгоритм распознавания общезначимости класса замкнутых формул, префиксные нормальные формы которых содержат только кванторы общности. Алгоритмически разрешим класс формул, построенных только из одноместных предикатов (монадическая логика). Монадическая логика остается разрешимой, даже если допустить кванторы по предикатным переменным. Оказался алгоритмически разрешимым ряд других классов формул логики предикатов. Ис-

следования по установлению разрешающих алгоритмов в ЛП показали, что относительно мало классов формул имеют разрешающие алгоритмы. Многие классы формул оказались алгоритмически неразрешимыми. Работа по отысканию разрешающих алгоритмов продолжается и поныне.

Покажем разрешимость вышеупомянутых примеров классов формул логики предикатов (без функциональных символов).

Формулы $A(x_1, \dots, x_n)$ и $(\forall x_1) \dots (\forall x_n) A(x_1, \dots, x_n)$ общезначимы или не общезначимы одновременно. Кроме того, для всякой формулы логики предикатов существует ей эквивалентная формула в префиксной нормальной форме. Поэтому проблему разрешимости в логике предикатов достаточно решать для замкнутых формул в префиксной нормальной форме.

6.5.1. Проблема разрешимости $\exists$ -формул

Определение. $\exists$ -формула есть замкнутая формула логики предикатов в префиксной нормальной форме, содержащая только кванторы существования. $\square$

Класс $\exists$ -формул алгоритмически разрешим, т. е. существует алгоритм, который для всякой $\exists$ -формулы A устанавливает, является формула A общезначимой или не является. Разрешающий алгоритм вытекает из следующей теоремы.

Теорема. $\exists$ -формула общезначима тогда и только тогда, когда она тождественно истинна на одноэлементном множестве.

Доказательство. Если формула A общезначима, она общезначима на всяком множестве, в том числе на одноэлементном множестве.

Пусть теперь $\exists$ -формула

$$B = (\exists x_1) \dots (\exists x_m) A(P_1, \dots, P_n, x_1, \dots, x_m),$$

где бескванторная формула A со свободными предикатными и предметными переменными $P_1, \dots, P_n, x_1, \dots, x_m$ тождественно истинна на одноэлементном множестве. Покажем, что формула B общезначима. Допустим противное: формула B общезначимой не является, т. е. существует множество S , предикаты $P_1, \dots, P_n$, определенные на S , для которых высказывание

$$B = (\exists x_1) \dots (\exists x_m) A(P_1, \dots, P_n, x_1, \dots, x_m) \text{ ложно.}$$

Тогда отрицание этого высказывания

$$\neg B = (\forall x_1) \dots (\forall x_m) \neg A(P_1, \dots, P_n, x_1, \dots, x_m) \text{ истинно.}$$

Пусть элемент $a \in S$. Тогда высказывание

$$\neg A(P_1, \dots, P_n, a, \dots, a) \text{ истинно.}$$

Пусть $M = \{a\}$ есть одноэлементное множество. Определим на множестве M предикаты $Q_i(x_1, \dots, x_{n_i})$ равенствами $Q_i(a, \dots, a) = P_i(a, \dots, a)$, $i = 1, 2, \dots, n$. Тогда

$$\neg A(Q_1, \dots, Q_n, a, \dots, a) \text{ истинно,}$$

$$A(Q_1, \dots, Q_n, a, \dots, a) \text{ ложно,}$$

$$(\exists x_1) \dots (\exists x_m) A(Q_1, \dots, Q_n, x_1, \dots, x_m) \text{ ложно на } M,$$

т. е. формула $(\exists x_1) \dots (\exists x_m) A(P_1, \dots, P_n, x_1, \dots, x_m)$ не является тождественно истинной на одноэлементном множестве. Противоречие. Следовательно, формула В общезначима.

6.5.2. Проблема разрешимости $\forall$ -формул

Определение. $\forall$ -формула есть замкнутая формула логики предикатов в префиксной нормальной форме, содержащая только кванторы общности. $\square$

Класс $\forall$ -формул алгоритмически разрешим, т. е. существует алгоритм, который устанавливает для всякой $\forall$ -формулы А, является формула А общезначимой или не является. Разрешающий алгоритм вытекает из следующей теоремы.

Теорема. $\forall$ -формула с m кванторами общности общезначима тогда и только тогда, когда она тождественно истинна на всяком множестве, состоящем из не более чем m элементов.

Доказательство. Если формула А общезначима, то она общезначима на всяком множестве, в том числе на всяком множестве, состоящем из не более чем m элементов.

Пусть теперь А-формула

$$A = (\forall x_1) \dots (\forall x_m) B(P_1, \dots, P_n, x_1, \dots, x_m),$$

где бескванторная формула В со свободными предикатными и предметными переменными $P_1, \dots, P_n, x_1, \dots, x_m$ тождественно истинна на всяком множестве, состоящем из не более чем m элементов. Покажем, что формула А общезначима. Допустим противное: формула А общезначимой не является, т. е. существует множество S , предикаты $P_1, \dots, P_n$, определенные на S , для которых высказывание

$$A = (\forall x_1) \dots (\forall x_m) B(P_1, \dots, P_n, x_1, \dots, x_m) \text{ ложно.}$$

Отрицание этого высказывания

$$\neg A = (\exists x_1) \dots (\exists x_m) \neg B(P_1, \dots, P_n, x_1, \dots, x_m) \text{ истинно.}$$

Тогда существуют элементы $a_1, \dots, a_m$, для которых высказывание

$$\neg B(P_1, \dots, P_n, a_1, \dots, a_m) \text{ истинно.}$$

Пусть $M = \{a_1, \dots, a_m\}$ есть множество, содержащее не более чем m элементов. Определим на множестве M предикаты $Q_i(x_1, \dots, x_{n_i})$ равенствами

$$Q_i(t_1, \dots, t_{n_i}) = P_i(t_1, \dots, t_{n_i}), i = 1, 2, \dots, n; t_1, \dots, t_{n_i} \in M.$$

Тогда

$$\begin{aligned} &\neg B(Q_1, \dots, Q_n, t_1, \dots, t_{n_i}) \text{ истинно,} \\ &(\exists x_1) \dots (\exists x_m) \neg B(Q_1, \dots, Q_n, x_1, \dots, x_m) \text{ истинно на } M, \\ &\neg(\exists x_1) \dots (\exists x_m) \neg B(Q_1, \dots, Q_n, x_1, \dots, x_m) \text{ ложно на } M, \\ &(\forall x_1) \dots (\forall x_m) B(P_1, \dots, P_n, x_1, \dots, x_m) \text{ ложно на } M, \end{aligned}$$

т. е. формула A не является тождественно истинной на множестве, состоящем из не более чем m элементов. Противоречие. Поэтому формула A общезначима.

6.5.3. Проблема разрешимости монадической логики

Определение. Монадическая формула есть формула логики предикатов, построенная только из одноместных предикатных переменных.

Класс монадических формул алгоритмически разрешим, т. е. существует алгоритм, который устанавливает для всякой монадической формулы A , общезначима A или необщезначима. Разрешающий алгоритм вытекает из следующей теоремы.

Теорема. Пусть A есть замкнутая монадическая формула в префиксной нормальной форме, построенная из n одноместных предикатных переменных. Тогда A общезначима тогда и только тогда, когда A тождественно истинна на всяком множестве, состоящем из не более чем 2^n элементов.

Доказательство. Если формула A общезначима, то она общезначима на всяком множестве, в том числе на всяком множестве, состоящем из не более чем 2^n элементов.

Пусть теперь монадическая формула A тождественно истинна на всяком множестве, состоящем из не более чем 2^n элементов. Пусть

$$A = (Q_1 x_1) \dots (Q_m x_m) B(P_1, \dots, P_n, x_1, \dots, x_m),$$

где $Q_i \in \{\exists, \forall\}$; B есть бескванторная формула; $P_1, \dots, P_n$ есть полный список одноместных предикатных переменных формулы A . Покажем, что формула A общезначима. Допустим противное: формула A общезначимой не является: существует множество M , одноместные предикаты $P_1, \dots, P_n$, определенные на M , для которых высказывание

$$(Q_1 x_1) \dots (Q_m x_m) B(P_1, \dots, P_n, x_1, \dots, x_m) \text{ ложно на } M.$$

Тогда его отрицание

$$(Q'_1 x_1) \dots (Q'_m x_m) \neg B(P_1, \dots, P_n, x_1, \dots, x_m) \text{ истинно на } M.$$

Здесь Q'_i есть $\exists$, если Q_i есть $\forall$, и Q'_i есть $\forall$, если Q_i есть $\exists$, $i = 1, \dots, m$.

Пусть x_0 есть произвольный элемент из S . Пусть набор $(a_1, \dots, a_n) = (P_1(x_0), \dots, P_n(x_0))$ есть набор истинностных значений предикатов $P_i(x_0)$. Определим на множестве M отношение эквивалентности $x \sim y$, положив для элементов x и y из M

$$x \sim y \leftrightarrow (P_1(x), \dots, P_n(x)) = (P_1(y), \dots, P_n(y)).$$

Множество M тогда разобьется на не более чем 2^n попарно непересекающихся классов эквивалентности

$$C_{a_1 \dots a_n} = \{x \in M : P_i(x) = a_i, i = 1, \dots, n\}.$$

Пусть $E = \{e_1, \dots, e_k\}$ есть множество построенных классов эквивалентности. Определим предикаты $R_i(x)$ на M , положив $R_i(e_j) = P_i(a_j)$, $i = 1, 2, \dots, n$; $j = 1, 2, \dots$,

$k \leq 2^n$; a_j есть произвольный элемент из e_j . Так как множество E состоит из не более чем 2^n элементов, то высказывание

$$(Q_1 x_1) \underbrace{(Q_2 x_2) \dots (Q_m x_m) B(R_1, \dots, R_n, x_1, \dots, x_m)}_{T(x_1)} \quad (*)$$

истинно на множестве E , состоящем из не более чем 2^n элементов. Высказывание

$$(Q'_1 x_1) \underbrace{(Q'_2 x_2) \dots (Q'_m x_m) \neg B(R_1, \dots, R_n, x_1, \dots, x_m)}_{S(x_1)} \quad (**)$$

истинно на M . То есть $(Q_1 x_1)T(x_1)$ истинно на E , и $(Q'_1 x_1)S(x_1)$ истинно на M .

Утверждение. Если $(Qx)T(x)$ истинно на E и $(Q'x)S(x)$ истинно на M , то существуют элементы $e_j \in E$ и $a_j \in e_j$, для которых $T(e_j)$ истинно на E и $S(a_j)$ истинно на M .

Доказательство. Возможны следующие два случая.

1. (Qx) есть $(\exists x)$. Тогда $(Q'x)$ есть $(\forall x)$, откуда $(\exists x)T(x)$ истинно на E и $(\forall x)S(x)$ истинно на M . По смыслу квантора существования существует элемент $x = e_j \in E$, для которого $T(e_j)$ истинно на E . По смыслу квантора общности $S(x)$ истинно для всякого $x \in M$. Выберем это $x = a_j \in e_j$. Тогда $S(a_j)$ истинно на M .

2. (Qx) есть $(\forall x)$. Тогда $(Q'x)$ есть $(\exists x)$, откуда $(\forall x)T(x)$ истинно на E и $(\exists x)S(x)$ истинно на M . По смыслу квантора существования существует элемент $x = a_j \in M$, для которого $S(a_j)$ истинно на M . По смыслу квантора общности $T(x)$ истинно для всякого $x \in E$, в том числе для элемента e_j , содержащего a_j . Тогда $T(e_j)$ истинно на M .

В обоих случаях существуют элементы $e_j \in E$ и $a_j \in e_j$, для которых $T(e_j)$ истинно на E и $S(a_j)$ истинно на M . Утверждение доказано.

Продолжим доказательство теоремы. Применим доказанное утверждение к формулам $(*)$ и $(**)$ и найдем элементы $e_1 \in E$ и $a_1 \in e_1$, для которых высказывание

$$(Q_2 x_2) \dots (Q_m x_m) B(R_1, \dots, R_n, e_1, x_2, \dots, x_m) \text{ истинно на } E,$$

$$(Q'_2 x_2) \dots (Q'_m x_m) \neg B(P_1, \dots, P_n, a_1, x_2, \dots, x_m) \text{ истинно на } M.$$

Аналогично, применяя доказанное утверждение, последовательно находим элементы $e_1, \dots, e_m$ из E и элементы $a_1, \dots, a_m$ из M , $a_j \in e_j$, $j = 1, 2, \dots, m$, для которых высказывание

$$B(R_1, \dots, R_n, e_1, e_2, \dots, e_m) \text{ истинно на } E,$$

$$\neg B(P_1, \dots, P_n, a_1, a_2, \dots, a_m) \text{ истинно на } M.$$

Эти формулы имеют одну и ту же алгебраическую структуру: они построены из одной и той же булевой формулы $F(q_1, \dots, q_r)$, в которой пропозициональные переменные q_i замещаются на $R_j(e_k)$ в первой формуле и на $P_j(a_k)$ во второй. Так как $R_j(e_k) = P_j(a_k)$, то обе формулы эквивалентны. Противоречие, ибо одна из них есть отрицание другой. Поэтому формула A общезначима. Теорема доказана.

В 70-е годы XX века был разработан универсальный язык программирования Пролог, в основу которого была положена логика предикатов. Вычисление общезначимости формул в Прологе проводится в приведенном Дж. А. Робинсоном логическом формализме, называемом методом резолюций. В последующем мы изложим этот метод и покажем, как он используется в универсальном языке программирования Пролог.

6.6. Отношения

Пусть $A_1, A_2, \dots, A_n$ есть произвольные множества, вообще говоря, разнородные.

Определение. n -арное отношение ρ^n на множествах $A_1, A_2, \dots, A_n$ есть подмножество декартова произведения $A_1 \times A_2 \times \dots \times A_n$.

Замечание. n -арное отношение ρ^n на множестве A есть подмножество декартова произведения $A \times A \times \dots \times A$ (n раз). Индекс n арности (местности) отношения иногда опускается.

Иногда отношение определяют на множестве $A_1 \times A_2 \times \dots \times A_n$.

Возможна предикатная $\rho(x_1, x_2, \dots, x_n)$ и множественная $(x_1, x_2, \dots, x_n) \in \rho$ формы записи отношений. Отношение ρ называют также предикатом. Для бинарного отношения используются записи $x \rho y$ и $\rho(x, y)$. Унарное отношение $\rho \subseteq E$ есть подмножество из E .

Набор $a = (a_1, a_2, \dots, a_n) \in \rho$ (допустима запись $\rho(x_1, x_2, \dots, x_n)$) называется элементом отношения.

Замечание. График функции $f: A \rightarrow B$ есть некоторое отношение, определенное на $A \times B$.

Определение. Отношение конечно, если оно состоит из конечного числа элементов.

Конечное отношение удобно задавать матрицей, строки которой есть эле-

менты отношения $\rho =$

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{r1} & a_{r2} & \dots & a_{rn} \end{bmatrix}.$$

Так как отношения есть множества, то над отношениями возможны операции объединения, пересечения, дополнения.

Утверждение. Семейство всех отношений, определенных на некотором множестве $A_1 \times A_2 \times \dots \times A_n$, относительно операций $\cup, \cap, \neg$ образует булеву алгебру.

Декартово произведение отношений ρ^n и σ^m на множествах $A_1, \dots, A_n$ и $A_{n+1}, \dots, A_{n+m}$ соответственно есть отношение

$$\theta^{n+m} = \rho^n \times \sigma^m = \{(a_1, \dots, a_n, a_{n+1}, \dots, a_{n+m}) \in A_1 \times A_2 \times \dots \times A_n \times A_{n+1}, \dots, A_{n+m} : (a_1, \dots, a_n) \in \rho^n, (a_{n+1}, \dots, a_{n+m}) \in \sigma^m\}.$$

Пример. $\rho = \begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix}, \sigma = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 2 & 1 \\ 2 & 2 & 1 & 2 \end{bmatrix}, \rho \times \sigma = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 2 & 1 \\ 0 & 1 & 2 & 2 & 1 & 2 \\ 1 & 2 & 0 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 & 2 & 1 \\ 1 & 2 & 2 & 2 & 1 & 2 \end{bmatrix}.$

Проекция $\text{Pr}_{j_1, j_2, \dots, j_k}$ P отношения $P \subseteq A_1 \times A_2 \times \dots \times A_n$ на оси $j_1, j_2, \dots, j_k$ ($1 \leq j_1 < j_2 < \dots < j_k \leq n$) есть все те наборы длины k , которые получаются из наборов отношения P , где стерты все компоненты, кроме компонент на местах $j_1, \dots, j_k$.

Функция $f: A_1 \times A_2 \times \dots \times A_{n-1} \rightarrow A_n$ униформизирует отношение $P \subseteq A_1 \times A_2 \times \dots \times A_n$ (по оси n), если $\forall (a_1, \dots, a_{n-1}) \in \text{Pr}_{1, \dots, n-1} P$ набор $(a_1, \dots, a_{n-1}, f(a_1, \dots, a_{n-1})) \in P$.

Можно говорить об униформизации отношения P функцией f по любой другой оси.

Для двумерного случая функция $f: A \rightarrow B$ униформизирует (по второй оси) бинарное отношение P из $A \times B$, если $\forall a \in \text{Pr}_1 P$ пара $(a, f(a)) \in P$. Функция $f: B \rightarrow A$ униформизирует (по первой оси) бинарное отношение P из $A \times B$, если $\forall b \in \text{Pr}_2 P$ пара $(f(b), b) \in P$.

Например, если $A = \{0, 1, 2, 3\}, B = \{0, 1, 2, 3, 4\}$, отношение

$P = \{(0, 1), (0, 3), (0, 4), (1, 0), (1, 4), (2, 1), (2, 3), (3, 0), (3, 4)\}$, то

функция $f: A \rightarrow B$, для которой $f(0)=3, f(1)=0, f(2)=1, f(3)=4$, униформизирует отношение P по второй оси (рис. 17.1).

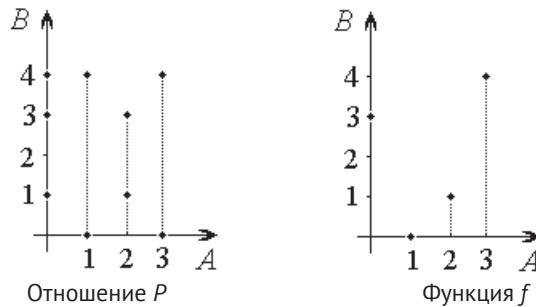


Рис. 6.1

На множестве отношений можно определить другие операции, обогащающие алгебру отношений и облегчающие их обработку.

Пусть $p = \begin{pmatrix} 1 & 2 & \dots & n \\ j_1 & j_2 & \dots & j_n \end{pmatrix}$ есть подстановка элементов множества $M = \{1, 2, \dots, n\}$, то есть взаимно-однозначная функция $p: M \rightarrow M$. Операция перестановки ко-

ординат в отношении ρ^n порождает отношение $p(\rho^n)$ на $A_{p(1)} \times A_{p(2)} \times \dots \times A_{p(n)}$, для которого $(a_1, \dots, a_n) \in \rho^n \leftrightarrow (a_{p(1)}, \dots, a_{p(n)}) \in p(\rho^n)$.

Например, если $\rho = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{r1} & a_{r2} & \dots & a_{rn} \end{bmatrix}$, то $p(\rho) = \begin{bmatrix} a_{1,p(1)} & a_{1,p(2)} & \dots & a_{1,p(n)} \\ a_{2,p(1)} & a_{2,p(2)} & \dots & a_{2,p(n)} \\ \dots & \dots & \dots & \dots \\ a_{r,p(1)} & a_{r,p(2)} & \dots & a_{r,p(n)} \end{bmatrix}$.

Если $p = \begin{pmatrix} 1 & 2 & \dots & n-1 & n \\ n & n-1 & \dots & 2 & 1 \end{pmatrix}$, то отношение $p(\rho^n)$ называется обратным к ρ

и обозначается ρ^{-1} . Очевидно, что $(a_1, a_2, \dots, a_n) \in \rho \leftrightarrow (a_n, \dots, a_2, a_1) \in \rho^{-1}$.

Для обратных отношений справедливы следующие равенства.

1. $(\rho^{-1})^{-1} = \rho$.
2. $\rho \subseteq \sigma \rightarrow \rho^{-1} \subseteq \sigma^{-1}$.
3. $\left(\bigcap_{i \in I} \rho \right)^{-1} = \bigcap_{i \in I} \rho^{-1}$.
4. $(\bar{\rho})^{-1} = \overline{\rho^{-1}}$.

В частности, для бинарных отношений $x \leq y, x < y, x \geq y, x > y$ справедливы соотношения: $(\leq)^{-1} = \geq, (<)^{-1} = >, (\geq)^{-1} = \leq, (>)^{-1} = <$.

6.7. Суперпозиция функций

Пусть E есть произвольное множество, декартово произведение $E^n = E \times E \times \dots \times E$, объект $f(x_1, \dots, x_n) : E^n \rightarrow E$ есть функция n переменных (n -местная функция f^n), $P_E = \{f^n : E^n \rightarrow E\}, n = 0, 1, \dots$ есть множество всех функций, определенных на множестве E . Нуль-местная функция есть константа из E .

Определение. Пусть F есть некоторое множество функций из P_E . Тогда

1. Всякая функция из F есть суперпозиция над F .
2. Если функция $f(x_1, \dots, x_n) \in F$ и каждое из $A_1, \dots, A_n$ есть либо суперпозиция над F , либо переменная, то $f(A_1, \dots, A_n)$ есть суперпозиция над F .

Определение. Класс M функций из P_E функционально замкнут, если вместе с любыми своими функциями класс M содержит и всякую суперпозицию над ними.

Определение. Замыкание $[M]$ множества M из P_E есть множество всех суперпозиций над M .

Замечание. 1. $M \subseteq [M]$. 2. $[[M]] = [M]$. 3. $M_1 \subseteq M_2 \rightarrow [M_1] \subseteq [M_2]$.

Пусть $f(x_1, \dots, x_n) : E^n \rightarrow E$ есть функция n переменных. Примем, что $D(f)$ есть область определения функции f ; $R(f)$ есть область значений функции f ; $f(E^n)$ есть область значений функции f на E^n . Если $C^n \subseteq E^n$, то $f(C^n)$ есть область тех значений функции f , когда аргументы функции f пробегают множество C . Запись $f|C$ означает ограничение функции f на множество C . При этом f есть расширение функции $f|C$.

6.8. Операции Мальцева над функциями

Пусть $f(x_1, \dots, x_n), g(x_1, \dots, x_m)$ есть функции из P_E . Введем следующие операции (Мальцева).

- $\zeta f(x_1, \dots, x_n) = f_\zeta(x_1, \dots, x_n) = f(x_2, \dots, x_n, x_1)$ есть циклическая перестановка аргументов;
- $\tau f(x_1, \dots, x_n) = f_\tau(x_1, \dots, x_n) = f(x_2, x_1, x_3, \dots, x_n)$ есть транспозиция аргументов x_1 и x_2 ;
- $\Delta f(x_1, \dots, x_n) = f_\Delta(x_1, \dots, x_n) = f(x_1, x_1, x_2, \dots, x_{n-1})$ есть отождествление переменных;
- $\nabla f(x_1, \dots, x_n) = f_\nabla(x_1, \dots, x_n) = f(x_2, \dots, x_{n+1})$ есть введение фиктивной переменной;
- $f(x_1, \dots, x_n) * g(x_1, \dots, x_m) = f(g(x_1, \dots, x_m), x_{m+1}, \dots, x_{m+n-1})$ есть композиция функций f и g .

Утверждение (Мальцев, 1966). 1. Всякая суперпозиция функций над $F \subseteq P_E$ может быть получена из функций системы F с помощью операций $\zeta, \tau, \Delta, \nabla, *$.

2. Функция, полученная из функций системы F с помощью операций $\zeta, \tau, \Delta, \nabla, *$, может быть получена как суперпозиция над функциями из F .

Определение. Алгебра Поста $(\Phi_E, \{\zeta, \tau, \Delta, \nabla, *\})$ есть множество функций $\Phi_E \subseteq P_E$, замкнутое относительно операций $\zeta, \tau, \Delta, \nabla, *$ (то есть замкнутое относительно суперпозиции).

Замечание. Алгебра Поста функций k -значной логики P_k есть алгебра функций, определенных на множестве $E_k = \{0, 1, \dots, k-1\}$.

6.9. Алгебра отношений (реляционная алгебра)

Пусть E есть произвольное множество, и пусть R_E есть класс всех отношений, определенных на E .

Замечание. Класс всех n -арных отношений, определенных на множестве E , есть множество $P(E^n)$ всех подмножеств множества E^n . Класс $P(E^n)$ относительно операций $\cup, \cap, \neg$ образует булеву алгебру (то есть удовлетворяет булевым соотношениям).

Операции Мальцева над отношениями

Пусть $\rho(x_1, \dots, x_n), \sigma(x_1, \dots, x_m)$ есть отношения, определенные на множестве E . Введем следующие операции (Мальцева):

- $\zeta \rho(x_1, \dots, x_n) = \rho_\zeta(x_1, \dots, x_n) = \rho(x_2, \dots, x_n, x_1)$ есть циклическая перестановка аргументов;
- $\tau \rho(x_1, \dots, x_n) = \rho_\tau(x_1, \dots, x_n) = \rho(x_2, x_1, x_3, \dots, x_n)$ есть транспозиция аргументов x_1 и x_2 ;
- $\Delta \rho(x_1, \dots, x_n) = \rho_\Delta(x_1, \dots, x_n) = \rho(x_1, x_1, x_2, \dots, x_{n-1})$ есть отождествление переменных;
- $\nabla \rho(x_1, \dots, x_n) = \rho_\nabla(x_1, \dots, x_n) = \rho(x_2, \dots, x_{n+1})$ есть введение фиктивной переменной;
- $\rho(x_1, \dots, x_n) * \sigma(x_1, \dots, x_m) = \rho_*(x_1, \dots, x_{n+m-2}) = \{(a_1, \dots, a_{n+m-2}) \in E^{n+m-2} : \exists a \in E (a_1, \dots, a_{n-1}, a) \in \rho \ \& \ (a, a_n, a_{n+1}, \dots, a_{n+m-2}) \in \sigma\}$ есть свертка отношений ρ и σ .

Замечание. 1. С помощью операций ζ и τ в отношении может быть получена произвольная перестановка переменных.

2. С помощью операций ζ , τ , Δ отождествление переменных может быть осуществлено на любых аргументных местах отношения.

3. С помощью операций ζ , τ , ∇ фиктивные переменные могут быть введены на любых аргументных местах отношения.

4. С помощью операций ζ , τ , $*$ свертка может осуществляться по любой переменной в обоих отношениях.

5. Кроме перечисленных в теории и практике программирования вводятся и другие операции над отношениями.

Пример. 1. Перестановка переменных. $A = \{0, 1, 2\}$.

$$\rho(x_1, x_2, x_3, x_4) = \begin{matrix} x_1 & x_2 & x_3 & x_4 \\ \begin{bmatrix} 0 & 1 & 0 & 2 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \end{matrix} \quad \rho_1(x_1, x_2, x_3, x_4) = \begin{matrix} x_3 & x_4 & x_1 & x_2 \\ \begin{bmatrix} 0 & 2 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{matrix}.$$

2. Отождествление переменных x_2 и x_4 . $A = \{0, 1, 2\}$.

$$\rho(x_1, \dots, x_5) = \begin{matrix} x_1 & x_2 & x_3 & x_4 & x_5 \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 2 \\ 0 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 2 & 0 \\ 1 & 2 & 0 & 1 & 1 \\ 2 & 2 & 1 & 2 & 2 \end{bmatrix} \end{matrix} \rightarrow \begin{matrix} \begin{bmatrix} 0 & 1 & 0 & 1 & 2 \\ 2 & 2 & 1 & 2 & 0 \\ 2 & 2 & 1 & 2 & 2 \end{bmatrix} \end{matrix}, \quad \rho_1(x_1, \dots, x_5) = \begin{matrix} x_1 & x_2 & x_3 & x_5 \\ \begin{bmatrix} 0 & 1 & 0 & 2 \\ 2 & 2 & 1 & 0 \\ 2 & 2 & 1 & 2 \end{bmatrix} \end{matrix}.$$

3. Введение фиктивной переменной x_0 . $A = \{0, 1\}$.

$$\rho(x_1, x_2, x_3) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \quad \nabla \rho = \sigma(x_0, x_1, x_2, x_3) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}.$$

$$\sigma(x_0, x_1, x_2, x_3) = \rho(x_1, x_2, x_3),$$

$$\sigma(0, 1, 0, 0) = \rho(1, 0, 0), \quad \sigma(1, 1, 0, 0) = \rho(1, 0, 0).$$

4. Свертка. $\rho = \begin{bmatrix} 0 & 2 & 0 \\ 1 & 2 & 1 \\ 0 & 3 & 2 \\ 1 & 1 & 3 \end{bmatrix}, \quad \sigma = \begin{bmatrix} 0 & 1 \\ 0 & 2 \\ 1 & 3 \\ 2 & 0 \end{bmatrix}, \quad \rho * \sigma = \begin{bmatrix} 0 & 2 & 1 \\ 0 & 2 & 2 \\ 1 & 2 & 3 \\ 1 & 3 & 0 \end{bmatrix}.$

Определение. Проекция отношения $\rho \in E^n$ на оси $1 \leq j_1 < \dots < j_k \leq n$ есть k -арное отношение σ , получаемое сохранением в ρ координат $j_1, \dots, j_k$ и удалением всех остальных.

Пример.

$$\rho = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ 2 & 0 & 1 & 1 & 0 & 3 \\ 0 & 1 & 2 & 4 & 0 & 1 \\ 0 & 1 & 2 & 4 & 0 & 2 \end{bmatrix}, \quad \text{Pr}_{2,4,5}(\rho) = \begin{bmatrix} x_2 & x_4 & x_5 \\ 0 & 1 & 0 \\ 1 & 4 & 0 \\ 1 & 4 & 0 \end{bmatrix} = \begin{bmatrix} x_2 & x_4 & x_5 \\ 0 & 1 & 0 \\ 1 & 4 & 0 \end{bmatrix}.$$

Замечание. Введенные операции могут рассматриваться на отношениях, определенных на разнородных множествах: $\rho \subseteq A_1 \times A_2 \times \dots \times A_n$.

6.10. Алгебра отношений k -значной логики

Пусть $E_k = \{0, 1, \dots, k-1\}$.

Определение. Коалгебра Поста $(MO, \{\zeta, \tau, \Delta, \nabla, *\})$ (алгебра отношений R_k , определенных на E_k) есть множество отношений MO , замкнутое относительно операций $\zeta, \tau, \Delta, \nabla, *$.

Определение. Функция $f(x_1, \dots, x_n)$ из P_k сохраняет h -арное отношение $\rho \subseteq E_k^h$, если $\forall n$ наборов

$$\begin{aligned} (a_{11}, \dots, a_{1h}) &\in \rho, \\ (a_{21}, \dots, a_{2h}) &\in \rho, \\ &\dots \\ (a_{n1}, \dots, a_{nh}) &\in \rho, \\ \mathbf{a}^1 & \quad \mathbf{a}^h \end{aligned} \quad \text{набор } (f(\mathbf{a}^1), \dots, f(\mathbf{a}^h)) \in \rho.$$

Вместо « f сохраняет ρ » говорят также « f есть полиформизм для ρ » и « ρ есть инвариант для f ».

Определение. Пусть $MF \subseteq P_k$ есть некоторое множество функций (k -значной логики) из P_k и $MO \subseteq R_k$ есть некоторое множество отношений из R_k , определенных на E_k .

Множество полиформизмов

$$\text{Pol}(MO) = \{f \in P_k : \forall \rho \in MO \text{ } f \text{ есть полиформизм для } \rho\}.$$

Множество инвариантов

$$\text{Inv}(MF) = \{\rho \in R_k : \forall f \in MF \text{ } \rho \text{ есть инвариант для } f\}.$$

Утверждение. 1. Если $(MF, \{\zeta, \tau, \Delta, \nabla, *\})$ есть алгебра функций k -значной логики (алгебра Поста) из P_k , то $(\text{Inv}(MF), \{\zeta, \tau, \Delta, \nabla, *\})$ есть алгебра отношений (коалгебра Поста) из R_k .

2. Если $(MO, \{\{\zeta, \tau, \Delta, \nabla, *\}\})$ есть алгебра отношений (коалгебра Поста) из R_k , то $(\text{Pol}(MO), \{\{\zeta, \tau, \Delta, \nabla, *\}\})$ есть алгебра функций k -значной логики (алгебра Поста) из P_k .

Замечание. Между алгебрами функций из P_k и алгебрами отношений из R_k можно установить взаимно-однозначное соответствие.

Аксиоматическое исчисление предикатов

7.1. Определение исчисления предикатов

Дадим формально-логическую (т. е. аксиоматическую) характеристику общезначимых формул логики предикатов, а именно построим аксиоматическое исчисление, в котором доказуемы общезначимые формулы логики предикатов, и только они.

Формальные символы, формулы, подформулы в исчислении предикатов (ИП) полностью совпадают с аналогичными понятиями в логике предикатов. Примем для простоты только, что при построении формул в рассматриваемом случае не используются функциональные термы.

Пусть A, B, C есть произвольные формулы исчисления предикатов (совпадающие с формулами логики предикатов).

Схемы аксиом

- A1) $A \rightarrow (B \rightarrow A)$.
- A2) $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$.
- A3) $A \& B \rightarrow A$.
- A4) $A \& B \rightarrow B$.
- A5) $(A \rightarrow B) \rightarrow ((A \rightarrow C) \rightarrow (A \rightarrow B \& C))$.
- A6) $A \rightarrow A \vee B$.
- A7) $B \rightarrow A \vee B$.
- A8) $(A \rightarrow C) \rightarrow ((B \rightarrow C) \rightarrow (A \vee B \rightarrow C))$.
- A9) $(A \rightarrow B) \rightarrow (\neg B \rightarrow \neg A)$.
- A10) $A \rightarrow \neg\neg A$.
- A11) $\neg\neg A \rightarrow A$.

Определение. Терм t свободен для переменной x в формуле $A(x)$, если никакое свободное вхождение x в $A(x)$ не лежит в области действия квантора $(\exists u)$ или $(\forall u)$, где u есть отличная от x переменная из терма t .

Примеры

- a) x_i свободен для x_i в $A(x_i)$, но не свободен для x_i в $(\forall x_i) A(x_i)$.

б) Терм $f(x_1, x_2)$ свободен для x_1 в формуле $(\forall x_2) A(x_1, x_2) \rightarrow B(x_1)$, но не свободен для x_1 в формуле $(\exists x_3) A(x_1, x_2) \rightarrow B(x_1)$.

с) Всякий терм, не содержащий переменных, свободен для любой переменной в любой формуле.

д) Терм t свободен для любой переменной в формуле A , если никакая переменная терма t не является связанной переменной в A .

е) x_i свободно для x_i в любой формуле.

ф) Всякий терм свободен для x_i в A , если A не содержит свободных вхождений x_i .

Аксиомы Бернаиса (АБ)

Пусть $A(x)$ есть произвольная формула ИП, содержащая свободную переменную x , и t есть терм, свободный для x в $A(x)$.

АБ1) $(\forall x)A(x) \rightarrow A(t)$.

АБ2) $A(t) \rightarrow (\exists x)A(x)$.

Накладываемые в аксиомах Бернаиса требования на терм t гарантируют, что при подстановке этого терма вместо переменной x не произойдет связывания свободных переменных терма t .

Правила вывода

$\frac{A, A \rightarrow B}{B}$ есть правило заключения (ПЗ). Здесь A и B есть произвольные формулы ИП.

$\frac{A \rightarrow B(x)}{A \rightarrow (\forall x)B(x)}$ есть правило навешивания квантора общности ($\forall$ -правило, или $\forall$ -ПР). Здесь A и $B(x)$ есть произвольные формулы ИП, причем формула A не содержит переменной x свободно.

$\frac{A(x) \rightarrow B}{(\exists x)A(x) \rightarrow B}$ есть правило навешивания квантора существования ($\exists$ -правило, или $\exists$ -ПР). Здесь $A(x)$ и B есть произвольные формулы ИП, причем формула B не содержит переменной x свободно.

В случае применения $\forall$ - или $\exists$ -правила говорят, что переменная x в формулах $A \rightarrow B(x)$ или $A(x) \rightarrow B$ затрагивается (используется, варьируется).

Формальные символы, формулы, система аксиом и правила вывода составляют исчисление предикатов.

Определение. Доказательство в ИП есть конечная последовательность формул, каждая из которых есть либо аксиома, либо получена из предыдущих членов последовательности по одному из правил вывода.

Определение. Формула A доказуема в ИП, если в нем существует доказательство, последней формулой которого является A . $\square$

Всякая аксиома есть доказуемая в ИП формула, длина доказательства которой равна 1.

Пусть Γ и Δ есть произвольные конечные совокупности формул ИП (возможно, пустые) и A, B, C есть произвольные формулы в ИВ.

Определение. Конечная последовательность формул ИП есть *вывод из совокупности формул* Γ , если всякая формула последовательности либо принадлежит Γ , либо доказуема, либо получена из предыдущих формул последовательности по одному из правил вывода.

Определение. Формула A *выводима* в ИП из совокупности формул Γ , если в ИП существует вывод из совокупности Γ , последней формулой которого является A .

Заметим, что доказуемая формула выводима из любой (в том числе из пустой) совокупности формул.

Теорема. Если формула A получена подстановкой некоторых формул ИП в доказуемую формулу исчисления высказываний, то формула A доказуема в ИП лишь с помощью правила заключения.

Доказательство. Пусть формула из ИП

$$A = \Pi_{p_1 p_2 \dots p_n}^{B_1 B_2 \dots B_n} C(p_1, p_2, \dots, p_n),$$

где $B_1, B_2, \dots, B_n$ есть формулы из ИП; $C(p_1, p_2, \dots, p_n)$ есть доказуемая в ИВ формула и $p_1, p_2, \dots, p_n$ есть полный список (пропозициональных) переменных формулы C . Согласно второй формулировке ИВ, формула C доказуема в ИВ лишь с помощью правила заключения.

Пусть $D_1, D_2, \dots, D_r (=C)$ есть доказательство формулы C в ИВ (с помощью только правила заключения); $q_1, q_2, \dots, q_m$ есть полный список переменных во всех формулах D_i ($i = 1, 2, \dots, r$); $E_1, E_2, \dots, E_m$ есть произвольные попарно различные формулы ИП, отличные от всех ранее упомянутых в теореме формул. Сопоставим каждому q_i формулу F_i согласно следующему равенству.

$$F_i = \begin{cases} E_i, & \text{если } q_i \text{ отлично от всех } p_1, p_2, \dots, p_n; \\ B_j, & \text{если } q_i \text{ есть } p_j; \end{cases}$$

$$i = 1, 2, \dots, m; j = 1, 2, \dots, n.$$

В доказательстве $D_1, D_2, \dots, D_r$ в ИВ заменим каждое q_i в формулах $D_1, D_2, \dots, D_r$ на F_i . Полученная последовательность формул ИП будет доказательством в ИП формулы A лишь с помощью правила заключения.

Следствие. В ИП справедливы все вспомогательные правила вывода, доказуемые в ИВ.

7.2. Теорема дедукции в исчислении предикатов

Теорема (дедукции). Если формула B выводима в ИП из (конечной) совокупности формул Γ и формулы A , причем в этом выводе не затрагивается никакая свободная переменная формулы A , то формула $A \rightarrow B$ выводима из Γ .

Доказательство. Индукция по длине k вывода. Заметим, что по условию теоремы правила навешивания кванторов на свободные переменные формулы A не используются.

Базис. $k = 1$. Возможны следующие случаи.

1. V есть A . Тогда $\vdash A \rightarrow A$, поэтому $\Gamma \vdash A \rightarrow A$.
2. $\vdash V$. Тогда $\vdash A \rightarrow V$ (ПВЗ). Поэтому $\Gamma \vdash A \rightarrow V$.
3. $V \in \Gamma$. Тогда $\Gamma \vdash V$ и по ПВЗ $\Gamma \vdash A \rightarrow V$.

Предположение индукции. Пусть теорема справедлива для всех выводов длины меньше k .

Шаг индукции. Покажем, что теорема справедлива для выводов длины k . Пусть $\Gamma, A \vdash V$ и пусть $V_1, V_2, \dots, V_{k-1}, V_k (= V)$ есть вывод длины k формулы V из совокупности Γ, A . Возможны следующие случаи.

1. V есть A .
2. $\vdash V$.
3. $V \in \Gamma$.

Эти случаи рассмотрены в базисе.

4. $V (= V_k)$ получается из двух формул V_i и $V_i \rightarrow V_k$ вывода по правилу заключения. Этот случай рассматривается аналогично доказательству теоремы дедукции в исчислении высказываний.

5. Пусть $\Gamma, A \vdash V$ и $V_1, V_2, \dots, V_i, \dots, V_k$, где $V_i = C \rightarrow D(x)$, $V_k = V = C \rightarrow (\forall x)D(x)$, есть вывод длины k формулы V из Γ, A . Причем V получено из V_i с помощью $\forall$ -правила, при этом формула C не содержит x свободно. Формула A тоже не содержит x свободно, иначе свободная переменная формулы A использовалась бы в выводе формулы V . Так как длина вывода $V_1, V_2, \dots, V_i$ из Γ, A меньше k , то по предположению индукции

- (1) $\Gamma \vdash A \rightarrow (C \rightarrow D(x))$. Далее
- (2) $\Gamma \vdash A \& C \rightarrow D(x)$, ПВ14(1), соединение посылок;
- (3) $\Gamma \vdash A \& C \rightarrow (\forall x)D(x)$, $\forall$ -ПР(2);
- (4) $\Gamma \vdash A \rightarrow (C \rightarrow (\forall x)D(x))$, ПВ15(3), разъединение посылок,

то есть $\Gamma \vdash A \rightarrow V$.

6. Пусть $\Gamma, A \vdash V$ и $V_1, V_2, \dots, V_i, \dots, V_k$, где $V_i = D(x) \rightarrow C$, $V_k = V = (\exists x)D(x) \rightarrow C$, есть вывод длины k формулы V из Γ, A . Причем V получено из V_i с помощью $\exists$ -правила, при этом формула C не содержит x свободно. Формула A тоже не содержит x свободно. Так как длина вывода $V_1, V_2, \dots, V_i$ из Γ, A меньше k , то по предположению индукции

- (1) $\Gamma \vdash A \rightarrow (D(x) \rightarrow C)$. Далее
- (2) $\Gamma \vdash D(x) \rightarrow (A \rightarrow C)$, ПВ13(1), перестановка посылок;
- (3) $\Gamma \vdash (\exists x)D(x) \rightarrow (A \rightarrow C)$, $\exists$ -ПР(2);
- (4) $\Gamma \vdash A \rightarrow ((\exists x)D(x) \rightarrow C)$, ПВ13(2), перестановка посылок,

то есть $\Gamma \vdash A \rightarrow V$. Теорема доказана.

Приведем несколько примеров доказуемых в ИП формул и несколько вспомогательных правил вывода.

Правило вывода 19. $(\forall x)A(x) \vdash A(x)$.

Доказательство.

- (1) $(\forall x)A(x)$, условие;
- (2) $(\forall x)A(x) \rightarrow A(x)$, АБ1;
- (3) $A(x)$, ПЗ(1,3).

Теорема 11. $\vdash (\exists x) \neg A(x) \rightarrow \neg (\forall x) A(x)$.

Доказательство.

- (1) $\neg A(x), (\forall x) A(x) \vdash A(x)$, ПВ19;
- (2) $\neg A(x), (\forall x) A(x) \vdash \neg A(x)$;
- (3) $\neg A(x) \vdash \neg (\forall x) A(x)$, ПВ18(1,2), приведение к абсурду;
- (4) $\vdash \neg \neg A(x) \rightarrow \neg (\forall x) A(x)$, ТД(3);
- (5) $\vdash (\exists x) \neg A(x) \rightarrow \neg (\forall x) A(x)$, $\exists$ -ПР(4).

Правило вывода 20.

- а) $A(x) \vdash (\forall x) A(x)$; б) $A(x) \vdash (\exists x) A(x)$.

Доказательство.

- (1) $A(x)$, условие;
- (2) $\vdash C$, где C есть произвольная доказуемая в ИП формула без свободной переменной x ;

- (3) $\vdash A(x) \rightarrow (C \rightarrow A(x))$, А1;
- (4) $C \rightarrow A(x)$, ПЗ(1,3);
- (5) $C \rightarrow (\forall x) A(x)$, $\forall$ -ПР(4), (x затрагивается);
- (6) $(\forall x) A(x)$, ПЗ(2,5).

Второе правило доказывается с помощью аксиомы Бернаиса.

Теорема 12. $\vdash \neg (\forall x) A(x) \rightarrow (\exists x) \neg A(x)$.

Доказательство.

- (1) $\neg (\forall x) A(x), A(x) \vdash (\forall x) A(x)$, ПВ20а, (в формуле $(\forall x) A(x)$ переменная x затрагивается, в $A(x)$ нет);

- (2) $\neg (\forall x) A(x), A(x) \vdash \neg (\forall x) A(x)$, в $A(x)$ x не затрагивается;
- (3) $\neg (\forall x) A(x) \vdash \neg A(x)$, ПВ18(1,2), приведение к абсурду;
- (4) $\vdash \neg A(x) \rightarrow (\exists x) \neg A(x)$, АБ2;
- (5) $\neg (\forall x) A(x) \vdash (\exists x) \neg A(x)$, ПЗ(3,4);
- (6) $\vdash \neg (\forall x) A(x) \rightarrow (\exists x) \neg A(x)$, ТД(5).

Теорема 13. $\vdash \neg (\exists x) A(x) \rightarrow (\forall x) \neg A(x)$.

Доказательство.

- (1) $\neg (\exists x) A(x), A(x) \vdash (\exists x) A(x)$, ПВ20б; (x не затрагивается);
- (2) $\neg (\exists x) A(x), A(x) \vdash \neg (\exists x) A(x)$;
- (3) $\neg (\exists x) A(x) \vdash \neg A(x)$, ПВ18(1,2), приведение к абсурду;
- (4) $\vdash \neg (\exists x) A(x) \rightarrow \neg A(x)$, ТД(3);
- (5) $\vdash \neg (\exists x) A(x) \rightarrow (\forall x) \neg A(x)$, $\forall$ -ПР(4).

Теорема 14. $\vdash (\forall x) \neg A(x) \rightarrow \neg (\exists x) A(x)$.

Доказательство.

- (1) $A(x), (\forall x) \neg A(x) \vdash \neg A(x)$, ПВ19;
- (2) $A(x), (\forall x) \neg A(x) \vdash A(x)$;
- (3) $A(x) \vdash \neg (\forall x) \neg A(x)$, ПВ18(1,2), приведение к абсурду;
- (4) $\vdash A(x) \rightarrow \neg (\forall x) \neg A(x)$, ТД(3);
- (5) $\vdash (\exists x) A(x) \rightarrow \neg (\forall x) \neg A(x)$, $\exists$ -ПР(4);
- (6) $\vdash \neg \neg (\forall x) \neg A(x) \rightarrow \neg (\exists x) A(x)$, ПВ5(5), контрапозиция;
- (7) $\vdash (\forall x) \neg A(x) \rightarrow \neg (\exists x) A(x)$, ПВ7(6), снятие двойного отрицания.

Пусть выражение $A \equiv B$ есть сокращение для $(A \rightarrow B) \& (B \rightarrow A)$. Показано, что $\vdash \neg(\forall x)A(x) \equiv (\exists x)\neg A(x)$; $\vdash \neg(\exists x)A(x) \equiv (\forall x)\neg A(x)$.

Аналогично в ИП можно доказать все равносильности, установленные ранее для формул логики предикатов.

7.3. Непротиворечивость исчисления предикатов

Определение. Исчисление предикатов *непротиворечиво*, если ни для какой его формулы не доказуемы одновременно ни она сама, ни ее отрицание. □

Построим соответствие h , согласно которому каждой формуле ИП сопоставим некоторую формулу исчисления высказываний следующим образом.

1. Если A есть элементарная формула $P(x_1, \dots, x_n)$, то $h(A) = P$.
2. Если формулам A и B из ИП сопоставлены формулы $h(A)$ и $h(B)$ в ИВ, то $h(\neg A) = \neg h(A)$, $h(A * B) = h(A) * h(B)$, где $*$ есть один из знаков: $\&$, $\vee$, $\rightarrow$.
3. Если формуле $A(x)$ сопоставлена формула $h(A)$, то $h((\exists x)A(x)) = h(A)$; $h((\forall x)A(x)) = h(A)$, т. е. кванторы вместе с их переменными стираются.

Если A есть формула в ИП, то $h(A)$ получается из A стиранием в A всех кванторов и всех предметных переменных.

Лемма. Если формула A доказуема в ИП, то формула $h(A)$ доказуема в ИВ.

Доказательство. Индукция по длине k доказательства.

Базис. $k = 1$. Возможны следующие случаи.

1. $\vdash C$ в ИП, причем формула C есть аксиома, полученная по одной из схем аксиом A1–A11. Тогда $h(C)$ есть формула в ИВ, построенная по той же схеме аксиом.

2. $\vdash C$, причем C есть $(\forall x)A(x) \rightarrow A(y)$ есть аксиома Бернаиса. Тогда $h(C) = h((\forall x)A(x) \rightarrow A(y)) = h(A) \rightarrow h(A)$. Поэтому $h(C)$ доказуема в ИВ. Если C есть $A(y) \rightarrow (\exists x)A(x)$ есть вторая аксиома Бернаиса, то доказуемость $A(C)$ в ИВ следует из тех же соображений.

Предположение индукции. Допустим, что лемма справедлива для всех доказуемых в ИП формул с длиной доказательства меньше k .

Шаг индукции. Покажем, что лемма справедлива для всех доказуемых в ИП формул с длиной доказательства k . Пусть $\vdash A$ в ИП и $A_1, A_2, \dots, A_k (=A)$ есть доказательство формулы A в ИП длины k . Возможны следующие случаи.

1. A есть аксиома. Тогда это рассмотренный выше случай базиса.

2. $\vdash A$ в ИП и $A_1, A_2, \dots, A_i, \dots, A_i \rightarrow A_k, \dots, A_k (=A)$ есть доказательство формулы A в ИП длины k , причем A_k получено из A_i и $A_i \rightarrow A_k$ по правилу заключения. Так как длины доказательств $A_1, A_2, \dots, A_i$ и $A_1, A_2, \dots, A_i, \dots, A_i \rightarrow A_k$ формул A_i и $A_i \rightarrow A_k$ меньше k , то по предположению индукции

(1) $\vdash h(A_i)$ в ИВ;

(2) $\vdash h(A_i \rightarrow A_k)$ в ИВ.

Так как $h(A_i \rightarrow A_k) = h(A_i) \rightarrow h(A_k)$, то

(3) $\vdash h(A_i) \rightarrow h(A_k)$ в ИВ; (4) $h(A_k)$, ПЗ(1,3).

3. $\vdash A$ в ИП и $A_1, A_2, \dots, A_i, \dots, A_k (=A)$ есть доказательство формулы A в ИП длины k , причем формула A_k , равная $C \rightarrow (\forall x)B(x)$, получена из формулы $A_i = C \rightarrow B(x)$

с помощью $\forall$ -правила. Так как длина доказательства $A_1, A_2, \dots, A_i$ формулы A_i меньше k , то по предположению индукции $\vdash h(A_i)$. Так как $h(A_i) = h(C \rightarrow B(x)) = h(C \rightarrow (\forall x)B(x)) = h(A_k)$, то $\vdash h(A_k)$ в ИВ.

4. $\vdash A$ в ИП и $A_1, A_2, \dots, A_i, \dots, A_k (=A)$ есть доказательство формулы A в ИП длины k , причем формула A_k , равная $(\exists x)B(x) \rightarrow C$, получена из формулы $A_i = B(x) \rightarrow C$ с помощью $\exists$ -правила. Далее аналогично предыдущему случаю.

Теорема. ИП непротиворечиво.

Доказательство. Если $\vdash A$ и $\vdash \neg A$ в ИП, то $\vdash h(A)$ и $\vdash \neg h(A)$ в ИВ. Противоречие с непротиворечивостью ИВ.

7.4. Семантическая полнота исчисления предикатов относительно класса общезначимых формул

Теорема. Всякая доказуемая в ИП формула общезначима.

Доказательство. Индукция по длине k доказательства формулы.

Базис. $k = 1$. Возможны следующие случаи.

1. $\vdash A$ и A есть аксиома, полученная по одной из схем аксиом A1–A11. Тогда A общезначима как формула, полученная подстановкой в тождественно истинную формулу алгебры логики.

2. $\vdash B$ и B есть аксиома $(\forall x)A(x) \rightarrow A(t)$. Покажем, что формула B общезначима. Пусть I есть интерпретация со множеством M , на котором определены все свободные переменные формулы B . Возможны следующие подслучаи:

а) при выбранных значениях переменных высказывание $(\forall x)A(x)$ ложно. Тогда высказывание $(\forall x)A(x) \rightarrow A(t)$ истинно (здесь t есть значение термина t на интерпретации I);

б) высказывание $(\forall x)A(x)$ истинно. Тогда высказывание $A(x)$ истинно при любых x из M , в том числе и при любом значении t термина t на интерпретации I , т. е. $A(t)$ истинно. Поэтому высказывание $(\forall x)A(x) \rightarrow A(t)$ истинно.

3. $\vdash B$ и B есть аксиома $A(t) \rightarrow (\exists x)A(x)$. Покажем, что формула B общезначима. Пусть I есть интерпретация со множеством M , на котором определены свободные переменные формулы B . Возможны следующие подслучаи:

а) при выбранных значениях переменных высказывание $A(t)$ ложно. Тогда высказывание $A(t) \rightarrow (\exists x)A(x)$ истинно;

б) высказывание $A(t)$ истинно. Тогда высказывание $(\exists x)A(x)$ истинно (при $x = t$), и потому $(\forall x)A(x) \rightarrow A(t)$ истинно.

Предположение индукции. Пусть теорема справедлива для всякой доказуемой формулы с длиной доказательства меньше k .

Шаг индукции. Покажем, что теорема справедлива для всякой доказуемой формулы с длиной доказательства k . Пусть $\vdash A$ и $A_1, A_2, \dots, A_i, \dots, A_k (=A)$ есть доказательство формулы A в ИП длины k . Возможны следующие случаи.

1. $\vdash A$ и A есть аксиома. A общезначима по случаю базиса.

2. $\vdash A$ в ИП и $A_1, A_2, \dots, A_i, \dots, A_i \rightarrow A_k, \dots, A_k (=A)$ есть доказательство формулы A в ИП длины k . Причем A_k получено из A_i и $A_i \rightarrow A_k$ по правилу заключения. Так

как длины доказательств $A_1, A_2, \dots, A_i$ и $A_1, A_2, \dots, A_i, \dots, A_i \rightarrow A_k$ формул A_i и $A_i \rightarrow A_k$ меньше k , то по предположению индукции формулы A_i и $A_i \rightarrow A_k$ общезначимы. По смыслу импликации формула A_k тоже общезначима.

3. $\vdash A$ в ИП и $A_1, A_2, \dots, A_i, \dots, A_k (=A)$ есть доказательство формулы A в ИП длины k , причем формула A_k , равная $C \rightarrow (\forall x)B(x)$, получена из формулы $A_i = C \rightarrow B(x)$ с помощью $\forall$ -правила. Так как длина доказательства $A_1, A_2, \dots, A_i$ формулы A_i меньше k , то по предположению индукции формула A_i , равная $C \rightarrow B(x)$, общезначима. Покажем, что формула A_k , равная $C \rightarrow (\forall x)B(x)$, тоже общезначима.

Допустим, что это не так и формула A_k общезначимой не является, т. е. существует множество M , переменные формулы A_k , определенные на M , для которых получившееся высказывание $C \rightarrow (\forall x)B(x)$ ложно. Тогда высказывание C истинно, а высказывание $(\forall x)B(x)$ ложно, откуда $(\exists x) \neg B(x)$ истинно, т. е. существует предмет $x = x_0 \in M$, для которого $\neg B(x_0)$ истинно, а потому $B(x_0)$ ложно. Тогда $C \rightarrow B(x_0)$ ложно. Противоречие с общезначимостью формулы $A_i = C \rightarrow B(x)$. Следовательно, формула A_k общезначима.

4. $\vdash A$ в ИП и $A_1, A_2, \dots, A_i, \dots, A_k (=A)$ есть доказательство формулы A в ИП длины k , причем формула A_k , равная $(\exists x)B(x) \rightarrow C$, получена из формулы $A_i = B(x) \rightarrow C$ с помощью $\exists$ -правила. Общезначимость формулы A доказывается аналогично предыдущему случаю.

Приступим теперь к доказательству основной теоремы о совпадении класса формул, доказуемых в ИП, и класса формул, общезначимых в логике предикатов. Общезначимость доказуемых формул уже установлена. Покажем теперь, что всякая общезначимая формула доказуема в ИП.

7.4.1. Непротиворечивые расширения исчисления предикатов

Определение. Два класса K и K' формул в ИП дедуктивно эквивалентны, если $K \vdash K'$ и $K' \vdash K$ в ИП.

Пример. Формулы $B = A(x_1, \dots, x_n)$ и $C = (\forall x_1) \dots (\forall x_n) A(x_1, \dots, x_n)$ дедуктивно эквивалентны. В самом деле, покажем, что $B \vdash C$:

- (1) $A(x_1, \dots, x_n)$;
- (2) $\vdash D$, где D есть произвольная доказуемая в ИП формула, не содержащая переменных $x_1, \dots, x_n$;
- (3) $D \rightarrow A(x_1, \dots, x_n)$, ПВЗ(1);
- (4) $D \rightarrow (\forall x_1) \dots (\forall x_n) A(x_1, \dots, x_n)$, последовательное применение $\forall$ -правила к (3);
- (5) $(\forall x_1) \dots (\forall x_n) A(x_1, \dots, x_n)$, ПЗ(2,4), т. е. $B \vdash C$.

Покажем теперь, что $C \vdash B$:

- (1) $(\forall x_1) \dots (\forall x_n) A(x_1, \dots, x_n)$;
- (2) $\vdash (\forall x_1) \dots (\forall x_n) A \rightarrow (\forall x_2) \dots (\forall x_n) A$, АБ1;
- (3) $(\forall x_2) \dots (\forall x_n) A(x_1, \dots, x_n)$, ПЗ(1,2).

После конечного числа подобных шагов получим

- (4) $A(x_1, \dots, x_n)$, т. е. $C \vdash B$.

Будем в дальнейшем рассматривать ИП, в котором аксиомы A1–A11 замкнуты квантором общности. По предыдущему замечанию полученная система аксиом дедуктивно эквивалентна ранее рассмотренной.

Определение. Класс K замкнутых формул ИП *непротиворечив*, если из K в ИП нельзя вывести A и $\neg A$ одновременно ни для какой формулы A .

Лемма. Если замкнутая формула $\neg A$ не выводима из непротиворечивого класса K замкнутых формул, то класс $K' = K \cup \{A\}$ непротиворечив.

Доказательство. Допустим, что в ИП существует формула B , для которой $K' \vdash B$ и $K' \vdash \neg B$. Тогда по правилу ПВ18 приведения к абсурду получим $K \vdash \neg A$. Противоречие с условием. Следовательно, класс $K \cup \{A\}$ непротиворечив.

Определение. Класс K замкнутых формул ИП *полон*, если для любой замкнутой формулы A либо $K \vdash A$, либо $K \vdash \neg A$.

Лемма. Если класс K замкнутых формул непротиворечив, то его можно расширить до непротиворечивого полного класса K_c замкнутых формул.

Доказательство. Пусть $B_1, B_2, \dots$ есть пересчет всех замкнутых формул ИП. Определим последовательность $K_0, K_1, \dots, K_n, \dots$ классов замкнутых формул следующим образом.

$$K_0 = K; K_{n+1} = \begin{cases} K_n, & \text{если } K_n \vdash \neg B_{n+1}; \\ K_n \cup \{B_{n+1}\}, & \text{если } K_n \nvdash \neg B_{n+1}. \end{cases}$$

Ясно, что $K_0 \subseteq K_1 \subseteq \dots \subseteq K_n \subseteq K_{n+1} \subseteq \dots$

Утверждение 1. Все классы K_n непротиворечивы.

Доказательство. Индукция по n .

Базис. $n = 0$. Класс K_0 непротиворечив по условию.

Предположение индукции. Допустим, что класс K_n непротиворечив.

Шаг индукции. Покажем, что класс K_{n+1} непротиворечив. Возможны следующие случаи.

1. $K_{n+1} = K_n$. Тогда класс K_{n+1} непротиворечив по предположению индукции.
2. $K_{n+1} = K_n \cup \{B_{n+1}\}$. Тогда класс K_n не выводит $\neg B_{n+1}$, и потому класс K_{n+1} непротиворечив по предыдущей лемме.

Утверждение 1 доказано. Продолжим доказательство леммы.

Пусть $K_c = K_0 \cup K_1 \cup \dots = \bigcup_{n=0}^{\infty} K_n$.

Утверждение 2. Класс K_c непротиворечив.

Доказательство. Допустим противное: класс K_c противоречив, т. е. существует формула A , для которой $K_c \vdash A$ и $K_c \vdash \neg A$. Оба вывода имеют конечную длину. Пусть K_r есть класс, который содержит все формулы из K_c , встречающиеся в этих выводах. Тогда $K_r \vdash A$ и $K_r \vdash \neg A$. Противоречие с непротиворечивостью класса K_r . Следовательно, класс K_c непротиворечив.

Утверждение 3. Класс K_c полон.

Доказательство. Пусть A есть произвольная замкнутая формула в ИП. Тогда A есть B_{j+1} при некотором j . Одно из двух: либо $K_j \vdash \neg B_{j+1}$, т. е. $K_j \vdash \neg A$, и тогда ввиду $K_j \subseteq K_c$ будет $K_c \vdash \neg A$, либо неверно, что $K_j \vdash \neg B_{j+1}$, и тогда $K_{j+1} = K_j \cup \{B_{j+1}\} = K_j \cup \{A\}$, откуда $K_{j+1} \vdash A$, а так как $K_{j+1} \subseteq K_c$, то $K_c \vdash A$.

Итак, для любой замкнутой формулы A либо $K_c \vdash A$, либо $K_c \vdash \neg A$. Утверждение 3 доказано.

Получили, что класс формул K_c непротиворечив и полон. Лемма доказана.

Формализмы F и G . Пусть K есть непротиворечивый класс замкнутых формул, F есть формализм, полученный из ИП добавлением к нему формул класса K в качестве аксиом. Формализм F непротиворечив. Построим формализм G , расширив формализм F натуральными числами. Опишем формализм G подробнее.

7.4.2. Формализмы G и G_k

Формальные символы формализма G

1. $x, y, z, \dots$ есть символы предметных переменных.
2. $0, 1, 2, \dots$ есть символы натуральных чисел.
3. $P, Q, R, P_1, Q_1, R_1, \dots$ есть символы предикатных переменных.
4. $\&, \vee, \rightarrow, \neg, \exists, \forall$ есть логические символы (связки).
5. $(,)$ есть скобки и запятая.

Термы формализма G

1. Всякая предметная переменная и предметная константа есть терм.
2. Всякое натуральное число есть терм.

Формулы формализма G

1. Если $t_1, \dots, t_k$ есть термы и P есть предикатная переменная, то $P(t_1, \dots, t_k)$ есть элементарная формула.
2. Все другие формулы получаются из элементарных с помощью булевых операций и операций навешивания кванторов.

Аксиомы исчисления G состоят из трех следующих групп.

Первая группа аксиом для G есть замыкания квантором общности аксиом, полученных по схемам аксиом $A1-A11$. При этом A, B, C есть формулы из G .

Аксиомы Бернаиса составляют вторую группу аксиом для G :

$$AB1. (\forall x)A(x) \rightarrow A(t). \quad AB2. A(t) \rightarrow (\exists x)A(x),$$

где $A(x)$ есть формула в G со свободной переменной x , t есть терм.

Формулы класса K составляют третью группу аксиом для G .

Правила вывода для G совпадают с правилами вывода для ИП.

Утверждение. Формализм G непротиворечив.

Доказательство. Допустим, что формализм G противоречив. Пусть формула B из G такова, что $\vdash_G B$ и $\vdash_G \neg B$; $n_1, n_2, \dots, n_r$ есть все символы натуральных чисел, встречающиеся в выводах формул B и $\neg B$; $y_1, y_2, \dots, y_r$ есть попарно различные предметные переменные, отличные от всех переменных, встречающихся в доказательствах формул B и $\neg B$. Заменим в этих доказательствах каждое n_i на y_i . При этом встречающиеся в доказательствах формул B и $\neg B$ аксиомы $(\forall x)A(x) \rightarrow A(n_i)$ и $A(n_i) \rightarrow (\exists x)A(x)$ из G перейдут в аксиомы $(\forall x)A(x) \rightarrow A(y_i)$ и $A(y_i) \rightarrow (\exists x)A(x)$ из ИП. Формулы B и $\neg B$ перейдут в формулы C и $\neg C$ в ИП. Доказательства для B и $\neg B$ в G перейдут в доказательства C и $\neg C$ в ИП, что про-

тиворечит непротиворечивости ИП. Следовательно, формализм G непротиворечив. Утверждение доказано.

Формализмы G_n . Пусть

$$(\exists x_1)B_1(x_1), (\exists x_2)B_2(x_2), \dots, (\exists x_k)B_k(x_k), \dots$$

есть пересчет всех замкнутых формул из G вида $(\exists x)V(x)$, где $V(x)$ есть формула из G с единственной свободной переменной x . Индукцией по k зададим последовательность формализмов G_k , $k = 0, 1, 2, \dots$

Базис. $k = 0$. $G_0 = G$. Пусть A_0 есть система аксиом для G_0 .

$k = 1$. Пусть p_1 есть наименьшее натуральное число, превосходящее все натуральные числа, встречающиеся в формуле $(\exists x_1)V_1(x_1)$, и формула $E_1 = (\exists x_1)V_1(x_1) \rightarrow V_1(p_1)$. Пусть $A_1 = A_0 \cup \{E_1\}$ есть система аксиом для G_1 .

Предположение индукции. Пусть формализм G_{k-1} и система аксиом A_{k-1} для него уже построены.

Шаг индукции. Построим формализм G_k и его систему аксиом A_k . Пусть p_k есть наименьшее натуральное число, превосходящее $p_1, p_2, \dots, p_{k-1}$ и все натуральные числа, встречающиеся в формуле $(\exists x_k)V_k(x_k)$; формула $E_k = (\exists x_k)V_k(x_k) \rightarrow V_k(p_k)$; $A_k = A_{k-1} \cup \{E_k\}$ есть система аксиом для исчисления G_k , т. е. G_k получается добавлением к G_{k-1} формулы E_k в качестве аксиомы.

Формализмы G_k , $k = 0, 1, 2, \dots$, построены. Обозначим через T_k множество доказуемых в G_k формул. Так как

$$A_0 \subset A_1 \subset \dots \subset A_k \subset A_{k+1} \subset \dots, \text{ то } \\ T_0 \subseteq T_1 \subseteq \dots \subseteq T_k \subseteq T_{k+1} \subseteq \dots,$$

т. е. всякая формула, доказуемая в G_k , доказуема и во всяком последующем исчислении.

Утверждение. Формализмы G_k непротиворечивы.

Доказательство. Индукция по k .

Базис. $k = 0$. Так как $G_0 = F$ и формализм F непротиворечив, то формализм G_0 непротиворечив.

Предположение индукции. Допустим, что формализм G_{k-1} непротиворечив.

Шаг индукции. Покажем, что формализм G_k непротиворечив. Допустим противное: формализм G_k противоречив, тогда в G_k существует формула A , для которой

- (1) $\vdash_{G_k} A$;
- (2) $\vdash_{G_k} \neg A$. Далее
- (3) $\vdash_{G_k} A \ \& \ \neg A$, ПВ2(1,2);
- (4) $\vdash_{G_k} A \ \& \ \neg A \rightarrow B$, $\Pi_{x,y}^{AB}(x \ \& \ \neg x \rightarrow y)$;
- (5) $\vdash_{G_k} B$, ПЗ(3,4); возьмем $B = \neg E_k$. Тогда
- (6) $\vdash_{G_k} \neg E_k$. Так как $A_k = A_{k-1} \cup \{E_k\}$, то $A_{k-1}, E_k \vdash_{G_k} \neg E_k$, а потому
- (7) $E_k \vdash_{G_k} \neg E_k$, откуда
- (8) $\vdash_{G_k} E_k \rightarrow \neg E_k$, ТД(7);

$$(9) \vdash_{G_k} (E_k \rightarrow \neg E_k) \rightarrow \neg E_k, \Pi_x^{Ek}((x \rightarrow \neg x) \rightarrow \neg x);$$

$$(10) \vdash_{G_k} \neg E_k, \text{ПЗ}(8,9); \text{т. е.}$$

$$(11) \vdash_{G_k} \neg((\exists x_k)B_k(x_k) \rightarrow B_k(p_k)); \text{откуда имеем}$$

$$(12) \vdash_{G_k} (\exists x_k)B_k(x_k) \& \neg B_k(p_k);$$

$$(13) \vdash_{G_k} (\exists x_k)B_k(x_k), \text{ПВ1}(12);$$

$$(14) \vdash_{G_k} \neg B_k(p_k), \text{ПВ1}(12).$$

Заменим в доказательстве формулы $\neg B_k(p_k)$ в G_{k-1} натуральное число p_k на переменную x_k , не встречающуюся среди других переменных в доказательстве формулы $\neg B_k(p_k)$. Тогда последовательность формул (1)–(14) преобразуется в доказательство в формализме G_{k-1} формулы $\neg B_k(x_k)$, а именно:

$$(15) \vdash_{G_{k-1}} \neg B_k(x_k). \text{Далее справедливо}$$

$$(16) \vdash_{G_{k-1}} C, \text{ где } C \text{ есть произвольная замкнутая доказуемая в ИП формула};$$

$$(17) \vdash_{G_{k-1}} C \rightarrow \neg B_k(x_k), \text{ПВ3}(15);$$

$$(18) \vdash_{G_{k-1}} C \rightarrow (\forall x_k) \neg B_k(x_k), \forall\text{-ПР}(17);$$

$$(19) \vdash_{G_{k-1}} (\forall x_k) \neg B_k(x_k), \text{ПЗ}(16,18);$$

$$(20) \vdash_{G_{k-1}} (\forall x_k) \neg B_k(x_k) \rightarrow \neg(\exists x_k) B_k(x_k), \text{T14};$$

$$(21) \vdash_{G_{k-1}} \neg(\exists x_k) B_k(x_k), \text{ПЗ}(19,20).$$



Получили, что в G_{k-1} доказуемы формула $(\exists x_k)B_k(x_k)$ и ее отрицание. Противоречие с непротиворечивостью G_{k-1} . Следовательно, формализм G_k непротиворечив. Утверждение доказано.

Формализмы Н и L. Пусть

$$Ax = A_0 \cup A_1 \cup \dots \cup A_k \cup \dots = \bigcup_{k=0}^{\infty} A_k$$

есть система аксиом для формализма Н. Множество

$$Tr = T_0 \cup T_1 \cup \dots \cup T_k \cup \dots = \bigcup_{k=0}^{\infty} T_k$$



доказуемых в формализме Н формул состоит из всех формул, доказуемых в формализмах $G_k, k = 0, 1, 2, \dots$

Утверждение. Формализм Н непротиворечив.

Доказательство. Допустим противное: формализм Н противоречив, т. е. в Н существует формула А, доказуемая в Н одновременно с $\neg A$. Оба доказательства содержат конечное число формул, каждая из которых доказуема в Н. Так как $Tr = \bigcup_{k=0}^{\infty} T_k$, то найдется номер r , для которого все формулы в обоих доказательствах содержатся в T_r . Тогда $\vdash A$ и $\vdash \neg A$. Противоречие с непротиворечивостью G_r . Утверждение доказано.

Пусть L есть непротиворечивое полное расширение формализма Н.

Лемма. Если $A(x)$ есть формула с единственной свободной переменной x , то $\vdash_L (\forall x)A(x) \leftrightarrow (\vdash_L A(0), \vdash_L A(1), \dots, \vdash_L A(k), \dots)$.

Доказательство. Пусть

$$(1) \vdash_L (\forall x)A(x). \text{Далее}$$



(2) $\vdash_L (\forall x)A(x) \rightarrow A(0)$, AB1;

(3) $\vdash_L A(0)$, ПЗ(1,2);

(4) $\vdash_L (\forall x)A(x) \rightarrow A(1)$, AB1;

(5) $\vdash_L A(1)$, ПЗ(1,4);

и т. д. Получили $\vdash_L A(0), \vdash_L A(1), \dots, \vdash_L A(k), \dots$

Пусть теперь $\vdash_L A(k)$, $k = 0, 1, 2, \dots$ Допустим, что в L формула $(\forall x)A(x)$ не доказуема. Так как формализм L полон, то

(1) $\vdash_L \neg(\forall x)A(x)$. Далее

(2) $\vdash_L \neg(\forall x)A(x) \rightarrow (\exists x)\neg A(x)$, T12;

(3) $\vdash_L (\exists x)\neg A(x)$, ПЗ(1,2).

Последняя формула есть некоторая формула $(\exists x_k)B_k(x)$. Тогда

(4) $\vdash_L (\exists x_k)B_k(x)$. Далее имеем

(5) $\vdash_L (\exists x_k)B_k(x) \rightarrow B_k(p_k)$, аксиома E_k ;

(6) $\vdash_L B_k(p_k)$, ПЗ(4,5);

(7) $\vdash_L \neg A(p_k)$.

Получили, что в формализме L доказуемы формула $A(p_k)$ и ее отрицание. Противоречие с непротиворечивостью L .

Лемма. Если $A(x)$ есть формула с единственной свободной переменной x , то $\vdash_L (\exists x)A(x) \leftrightarrow (\vdash_L A(0) \vee \vdash_L A(1) \vee \dots \vee \vdash_L A(k) \vee \dots)$.

Доказательство. Пусть $\vdash_L (\exists x)A(x)$. Допустим противное: формулы $A(k)$ не доказуемы в L ни при каком k . Так как формализм L полон, то

$\vdash_L \neg A(0), \vdash_L \neg A(1), \dots, \vdash_L \neg A(k), \dots$.

Тогда

(1) $\vdash (\forall x)\neg A(x)$. Далее

(2) $\vdash (\forall x)\neg A(x) \rightarrow \neg(\exists x)A(x)$, T14;

(3) $\vdash \neg(\exists x)A(x)$, ПЗ(1,2).

Получили, что в исчислении L доказуемы формула $\neg(\exists x)A(x)$ и ее отрицание одновременно. Противоречие с непротиворечивостью L . Следовательно, справедливо $\vdash_L A(0) \vee \vdash_L A(1) \vee \dots$

Пусть теперь существует такое натуральное число p , для которого

(1) $\vdash_L A(p)$. Далее

(2) $\vdash_L A(p) \rightarrow (\exists x)A(x)$, AB2;

(3) $\vdash_L (\exists x)A(x)$, ПЗ(1,2). Лемма доказана.



Лемма. Всякий непротиворечивый класс K замкнутых формул ИП выполним на множестве натуральных чисел.

Доказательство. Пусть $\square = \{0, 1, 2, \dots\}$ есть множество натуральных чисел. Индукцией по построению формул из класса K зададим на $\square$ предикаты так, чтобы доказуемые в L формулы стали истинными высказываниями, т. е. покажем, что формулы класса T_L , доказуемые в исчислении L , выполнимы на множестве натуральных чисел. Пусть $P(a_1, a_2, \dots, a_n)$ есть элементарная формула в L , где P есть предикатная буква; $a_1, a_2, \dots, a_n$ есть символы натуральных чисел. В силу полноты L либо $\vdash_L P(a_1, a_2, \dots, a_n)$, либо $\vdash_L \neg P(a_1, a_2, \dots, a_n)$. Пусть предикат

$$P(a_1, a_2, \dots, a_n) = \begin{cases} \text{И, если } \vdash_L P(a_1, a_2, \dots, a_n), \\ \text{Л, если } \vdash_L \neg P(a_1, a_2, \dots, a_n). \end{cases}$$

Индукцией по глубине k построения формулы покажем, что при так определенных предикатах замкнутые доказуемые в L формулы принимают истинные значения.

Базис. $k = 1$. Пусть $\vdash_L A$. Тогда A есть элементарная формула $P(a_1, a_2, \dots, a_n)$, причем $\vdash_L P(a_1, a_2, \dots, a_n)$. Поэтому $P(a_1, a_2, \dots, a_n) = \text{И}$.

Предположение индукции. Допустим, что все доказуемые в L замкнутые формулы с глубиной построения меньше k при вышеопределенных на N предикатах принимают значение И.

Шаг индукции. Покажем, что все доказуемые в L замкнутые формулы с глубиной построения k при вышеопределенных на N предикатах принимают значение И.

Пусть $\vdash_L A$, причем формула A имеет глубину построения k . Возможны следующие случаи.

1. A есть $(\forall x)B(x)$. Тогда

(1) $\vdash_L (\forall x)B(x)$. Далее имеем

(2) $\vdash_L (\forall x)B(x) \leftrightarrow (\vdash_L B(0), \vdash_L B(1), \dots, \vdash_L B(p), \dots)$;

(3) $\vdash_L B(0), \vdash_L B(1), \dots, \vdash_L B(p), \dots$

Так как глубины построения формул $B(p)$, $p = 0, 1, 2, \dots$, меньше k , то по предположению индукции высказывания

$B(0) = \text{И}, B(1) = \text{И}, \dots, B(p) = \text{И}, \dots$

По смыслу квантора общности высказывание $(\forall x)B(x)$ истинно.

2. $\vdash_L A$ и A есть $(\exists x)B(x)$. Тогда

(1) $\vdash_L (\exists x)B(x)$. Далее имеем

(2) $\vdash_L (\exists x)B(x) \leftrightarrow (\vdash_L B(0) \vee \vdash_L B(1) \vee \dots \vee \vdash_L B(p) \vee \dots)$;

(3) $\vdash_L B(0) \vee \vdash_L B(1) \vee \dots \vee \vdash_L B(p) \vee \dots$

Пусть для определенности $\vdash_L B(p)$. Так как глубина построения формулы $B(p)$ меньше k , то по предположению индукции высказывание $B(p)$ истинно. По смыслу квантора общности высказывание $(\forall x)B(x)$ истинно.

3. $\vdash_L A$ и A есть $B \& C$. Тогда $\vdash_L B \& C$, откуда по ПБ2 получаем $\vdash_L B$ и $\vdash_L C$. Так как глубины построения формул B и C меньше k , то по предположению индукции высказывания B и C истинны. Тогда $B \& C$ истинно, т. е. высказывание A истинно.

Случаи, когда A есть $B \vee C$, $B \rightarrow C$, $\neg B$, рассматриваются аналогично.

Итак, все доказуемые в L замкнутые формулы при вышеопределенных на N предикатах принимают истинные значения. Так как формулы класса K доказуемы в L как аксиомы, то класс K замкнутых формул выполним на множестве натуральных чисел. Иначе говоря, непротиворечивый класс замкнутых формул ИП имеет модель.

Теорема. Всякая замкнутая общезначимая в логике предикатов формула доказуема в ИП.

Доказательство. Пусть A есть общезначимая формула логики предикатов. Допустим противное: формула A в ИП не доказуема. Тогда класс формул $K = \{\neg A\}$ непротиворечив и по предыдущей лемме формула $\neg A$ выполнима на множестве натуральных чисел. Противоречие с общезначимостью формулы A . Следовательно, общезначимая формула A доказуема в ИП.

Из двух вышедших теорем следует следующая теорема.

Теорема (Геделя о полноте). Замкнутая формула доказуема в ИП тогда и только тогда, когда она общезначима.

Замечание. К исчислению предикатов можно добавить специальный предикат равенства $x = y$ и две характеризующие его аксиомы равенства (AP):

$$AP1) x = x;$$

$$AP2) x = y \rightarrow (A(x) \rightarrow A(y)),$$

где $A(x)$ есть произвольная формула ИП, имеющая свободную переменную x (возможно, и другие переменные).

В ИП с равенством справедлива теорема о совпадении класса общезначимых формул (с равенством) и класса формул, доказуемых в ИП (с равенством).

ИП с равенством можно расширить далее, допустив при построении элементарных формул использование функциональных термов. И здесь справедлива вышеприведенная теорема.

Заметим еще, что если формула $A(P_1, \dots, P_n)$ с указанным множеством предикатных переменных доказуема в исчислении предикатов, то можно построить такое ее доказательство, в котором не встречаются предикатные переменные, отличные от $P_1, \dots, P_n$.





Исчисление секвенций

8.1. О правилах вывода

В секвенциальном исчислении высказываний

В классическом аксиоматическом исчислении Гильберта выделены минимально необходимые дедуктивные средства для вывода всякой общезначимой формулы логики предикатов, удобные для исследования свойств самого формализма, но приводящие к довольно громоздким вычислениям при выводе даже простых общезначимых формул. Это делает использование формализма Гильберта для установления общезначимости формул логики предикатов практически невозможным. Чтобы сделать установление общезначимости формулы более привычным, Гентцен предложил формализм с правилами вывода, довольно естественно описывающими свойства логических операций, с помощью которых эти формулы построены, благодаря чему вывод в формализме Гентцена делается более простым и наглядным. Этот формализм называют также естественным выводом Гентцена. Приведем несколько предварительных соображений относительно гентценовских правил вывода.

Пусть $A_1, A_2, \dots, A_n$ и $B_1, B_2, \dots, B_m$ есть произвольные совокупности формул ИВ. Будем вместо выводимости $A_1 \& A_2 \& \dots \& A_n \vdash B_1 \vee B_2 \vee \dots \vee B_m$ (что эквивалентно $A_1, A_2, \dots, A_n \vdash B_1 \vee B_2 \vee \dots \vee B_m$) записывать $A_1, A_2, \dots, A_n \mid\Rightarrow B_1; B_2; \dots; B_m$. Выражение $A_1, A_2, \dots, A_n \mid\Rightarrow B_1; B_2; \dots; B_m$ называется *секвенцией* (Гентцена). При $n = 0$ пустая конъюнкция $A_1 \& A_2 \& \dots \& A_n$ рассматривается как истина. При $m = 0$ пустая дизъюнкция $B_1 \vee B_2 \vee \dots \vee B_m$ рассматривается как ложь. Левую часть секвенции назовем *антецедент*, правую – *сукцедент*. Очевидно, что

$$\begin{aligned} A_1, A_2, \dots, A_n \mid\Rightarrow B_1; B_2; \dots; B_m &\leftrightarrow \\ A_1 \& A_2 \& \dots \& A_n \vdash B_1 \vee B_2 \vee \dots \vee B_m &\leftrightarrow \\ \vdash A_1 \& A_2 \& \dots \& A_n \rightarrow B_1 \vee B_2 \vee \dots \vee B_m. \end{aligned}$$

Содержательно секвенция $A_1, A_2, \dots, A_n \mid\Rightarrow B_1; B_2; \dots; B_m$ означает, что дизъюнкция $B_1 \vee B_2 \vee \dots \vee B_m$ логически следует из конъюнкции $A_1 \& A_2 \& \dots \& A_n$.

Назовем секвенцию $A_1, A_2, \dots, A_n \mid\Rightarrow B_1; B_2; \dots; B_m$ (обозначим ее через S) *справедливой (верной)*, если $A_1 \& A_2 \& \dots \& A_n \vdash B_1 \vee B_2 \vee \dots \vee B_m$.

В секвенции $A_1, A_2, \dots, A_n \mid \Rightarrow B_1; B_2; \dots; B_m$ совокупности $A_1, A_2, \dots, A_n$ и $B_1; B_2; \dots; B_m$ можно рассматривать как множества. Поэтому формулы в них можно переставлять, можно добавлять или удалять повторы формул, ибо применение таких операций к формулам вида $A_1 \& A_2 \& \dots \& A_n \rightarrow B_1 \vee B_2 \vee \dots \vee B_m$ приводит к эквивалентным формулам. В связи с таким множественным пониманием секвенции выражения $\neg(A_1, A_2, \dots, A_n)$ и $\neg(B_1; B_2; \dots; B_m)$ будем воспринимать как $\neg(A_1 \& A_2 \& \dots \& A_n)$ и $\neg(B_1 \vee B_2 \vee \dots \vee B_m)$ соответственно.

В последующих доказательствах воспользуемся доказанными ранее равносильностями и правилами вывода, не оговаривая этого особо.

Пусть Γ и Δ есть произвольные конечные совокупности формул ИВ (возможно, пустые). Считаем, что в левой части секвенции эти формулы представляют собой конъюнкцию составляющих формул (т. е. записываются через запятую), а в правой части дизъюнкцию (т. е. записываются через точку с запятой).

Устанавливая факт общезначимости данной формулы, попытаемся построить опровержение для этой формулы (т. е. интерпретацию, на которой формула ложна). Упрощая секвенции в дереве строящегося вывода, будем судить о ложности формул данной секвенции справа от знака $\mid \Rightarrow$ и об их истинности слева от знака $\mid \Rightarrow$. Если при этом удастся построить интерпретацию, при которой формулы слева от знака $\models$ истинны, а справа ложны, то это будет означать, что данная секвенция, рассматриваемая как формула логики, ложна (ибо ее посылка истинна, а заключение ложно), а потому общезначимой не является и в формализме не доказуема. При подобном анализе данной формулы (от корня к листьям) потребуется переходить от знаменателя используемого правила вывода к его числителю, а построение дерева вывода (от листьев к корню) на основе проведенного анализа будет использовать правила вывода от числителя к знаменателю. Поэтому содержательный смысл последующих восьми правил вывода поясним от знаменателя к числителю.

Определение. Пусть $A_1, A_2, \dots, A_n; B_1, B_2, \dots, B_m; C_1, C_2, \dots, C_r; D_1, D_2, \dots, D_s$ есть произвольные конечные совокупности формул. Тогда правило вывода

$$\frac{A_1, A_2, \dots, A_n \mid \Rightarrow B_1; B_2; \dots; B_m}{C_1, C_2, \dots, C_r \mid \Rightarrow D_1; D_2; \dots; D_s}$$

справедливо в секвенциальном исчислении высказываний (СИВ), если в ИВ справедливо правило вывода

$$\frac{A_1, A_2, \dots, A_n \mid \Rightarrow B_1 \vee B_2 \vee \dots \vee B_m}{C_1, C_2, \dots, C_r \mid \Rightarrow D_1 \vee D_2 \vee \dots \vee D_s}.$$

Утверждение 1

- а) $\frac{A_1, A_2, \dots, A_n \mid \Rightarrow B_1; B_2; \dots; B_m}{A_1, A_2, \dots, A_n, \neg B_1 \mid \Rightarrow B_2; \dots; B_m};$
- б) $\frac{A_1, A_2, \dots, A_n, B_1 \mid \Rightarrow \neg B_2; \dots; B_m}{A_1, A_2, \dots, A_n \mid \Rightarrow \neg B_1; B_2; \dots; B_m}.$



Доказательство. Из двух правил вывода докажем первое. Второе доказыва-
ется аналогично.

$$\begin{aligned} A_1, A_2, \dots, A_n \models B_1; B_2; \dots; B_m &\leftrightarrow A_1 \& A_2 \& \dots \& A_n \vdash B_1 \vee B_2 \vee \dots \vee B_m \leftrightarrow \\ A_1, A_2, \dots, A_n \vdash \neg B_1 \rightarrow (B_2 \vee \dots \vee B_m &\leftrightarrow A_1, A_2, \dots, A_n, \neg B_1 \vdash B_2 \vee \dots \vee B_m \leftrightarrow \\ A_1, A_2, \dots, A_n, \neg B_1 \models B_2; \dots; B_m. \end{aligned}$$

Замечание. Оба правила вывода обратимы: от знаменателя правила можно
переходить к числителю, и наоборот.

Следствие. Справедливы обратимые правила вывода введения отрицания
слева и справа от знака $\models$:

$$\frac{\Gamma \models \Delta; A}{\Gamma, \neg A \models \Delta} (\neg \models); \quad \frac{\Gamma, A \models \Delta}{\Gamma \models \Delta; \neg A} (\models \neg).$$

Повторное применение приведенных правил вывода позволяет снимать
(и ставить) двойные отрицания в формулах. В скобках указаны обозначения
этих правил: введение отрицания.

Замечание. Содержательно: для первого правила вывода в секвенции $\Gamma, \neg A \models \Delta$ формула $\neg A$ (слева от знака $\models$) истинна, если формула A (справа от
знака $\models$ в секвенции $\Gamma \models \Delta; A$) ложна. Для второго правила в секвенции $\Gamma \models \Delta; \neg A$ формула $\neg A$ ложна, если формула A в секвенции $\Gamma, A \models \Delta$ истинна.

Утверждение 2. Справедливо обратимое правило вывода:

$$\frac{A_1, \dots, A_n, A, B \models B_1; \dots; B_m}{A_1, \dots, A_n, A \& B \models B_1; \dots; B_m}.$$

Доказательство.

$$\begin{aligned} A_1, \dots, A_n, A, B \models B_1; \dots; B_m &\leftrightarrow A_1 \& \dots \& A_n \& A \& B \vdash B_1 \vee \dots \vee B_m \leftrightarrow \\ A_1 \& \dots \& A_n, A \& B \vdash B_1 \vee \dots \vee B_m &\leftrightarrow A_1, \dots, A_n, A \& B \models B_1; \dots; B_m. \end{aligned}$$

Следствие. Справедливо обратимое правило введения конъюнкции слева
от знака $\models$:

$$\frac{\Gamma, A, B \models \Delta}{\Gamma, A \& B \models \Delta} (\& \models).$$

Замечание. Содержательно: в секвенции $\Gamma, A \& B \models \Delta$ формула $A \& B$ (сле-
ва от знака $\models$) истинна, если формулы A, B (слева от знака $\models$ в секвенции
 $\Gamma, A, B \models \Delta$, где запятая означает конъюнкцию) истинны.

В некоторых из последующих утверждений в числителе правила вывода сто-
ят несколько секвенций. Имеется в виду, что они связаны дизъюнкцией.

Утверждение 3. Справедливо обратимое правило введения конъюнкции
справа от знака $\models$:

$$\frac{\Gamma \models \Delta; A \quad \Gamma \models \Delta; B}{\Gamma \models \Delta; A \& B} (\models \&).$$

Доказательство. $\Gamma \vdash \Delta \vee A \quad \Gamma \vdash \Delta \vee B \leftrightarrow$

$\Gamma \vdash \neg \Delta \rightarrow A \quad \Gamma \vdash \neg \Delta \rightarrow B \leftrightarrow$

$\Gamma, \neg \Delta \vdash A \quad \Gamma, \neg \Delta \vdash B \leftrightarrow$

$\Gamma, \neg \Delta \vdash A \& B \leftrightarrow \Gamma \vdash \neg \Delta \rightarrow A \& B \leftrightarrow$

$\Gamma \vdash \Delta \vee A \& B \leftrightarrow \Gamma \mid \Rightarrow \Delta; A \& B.$

Замечание. Содержательно: в секвенции $\Gamma \mid \Rightarrow \Delta; A \& B$ формула $A \& B$ ложна, если ложна формула A (в секвенции $\Gamma \mid \Rightarrow \Delta; A$) или ложна формула B (в секвенции $\Gamma \mid \Rightarrow \Delta; B$).

Утверждение 4. Справедливо обратимое правило введения дизъюнкции слева от знака $\mid \Rightarrow$:

$$\frac{\Gamma, A \mid \Rightarrow \Delta \quad \Gamma, B \mid \Rightarrow \Delta}{\Gamma, A \vee B \mid \Rightarrow \Delta} (\vee \mid \Rightarrow).$$



Доказательство. $\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta \leftrightarrow$

$\Gamma \vdash A \rightarrow \Delta \quad \Gamma \vdash B \rightarrow \Delta \leftrightarrow$

$\Gamma \vdash \neg A \vee \Delta \quad \Gamma \vdash \neg B \vee \Delta \leftrightarrow$

$\Gamma \vdash \neg \Delta \rightarrow \neg A \quad \Gamma \vdash \neg \Delta \rightarrow \neg B \leftrightarrow$

$\Gamma, \neg \Delta \vdash \neg A \quad \Gamma, \neg \Delta \vdash \neg B \leftrightarrow$

$\Gamma, \neg \Delta \vdash \neg A \& \neg B \leftrightarrow$

$\Gamma \vdash \neg \Delta \rightarrow \neg A \& \neg B \leftrightarrow$

$\Gamma \vdash \Delta \vee \neg A \& \neg B \leftrightarrow$

$\Gamma \vdash \neg(\neg A \& \neg B) \rightarrow \Delta \leftrightarrow$

$\Gamma, \neg(\neg A \& \neg B) \vdash \Delta \leftrightarrow$

$\Gamma, A \vee B \vdash \Delta \leftrightarrow$

$\Gamma, A \vee B \mid \Rightarrow \Delta$



Замечание. Содержательно: в секвенции $\Gamma, A \vee B \mid \Rightarrow \Delta$ формула $A \vee B$ истинна, если истинна формула A (в секвенции $\Gamma, A \mid \Rightarrow \Delta$) или истинна формула B (в секвенции $\Gamma, B \mid \Rightarrow \Delta$).

Утверждение 5. Справедливо обратимое правило введения дизъюнкции справа от знака $\mid \Rightarrow$:

$$\frac{\Gamma \mid \Rightarrow \Delta; A; B}{\Gamma \mid \Rightarrow \Delta; A \vee B} (\mid \Rightarrow \vee).$$

Замечание. Содержательно: в секвенции $\Gamma \mid \Rightarrow \Delta; A \vee B$ формула $A \vee B$ ложна, если ложна формула A и ложна формула B (в секвенции $\Gamma \mid \Rightarrow \Delta; A; B$, где точка с запятой означает дизъюнкцию).

Утверждение 6. Справедливо обратимое правило введения импликации справа от знака $\mid \Rightarrow$:

$$\frac{\Gamma, A \mid \Rightarrow \Delta; B}{\Gamma \mid \Rightarrow \Delta; A \rightarrow B} (\mid \Rightarrow \rightarrow).$$

Доказательство.

$\Gamma, A \vdash \Delta; B \leftrightarrow \Gamma, A \vdash \Delta \vee B \leftrightarrow$

$\Gamma, A \vdash \neg \Delta \rightarrow B \leftrightarrow \Gamma, A, \neg \Delta \vdash B \leftrightarrow \Gamma, \neg \Delta \vdash A \rightarrow B \leftrightarrow$

$\Gamma \vdash \neg \Delta \rightarrow (A \rightarrow B) \leftrightarrow \Gamma \vdash \Delta \vee (A \rightarrow B) \leftrightarrow \Gamma \vdash \Delta; (A \rightarrow B).$

Замечание. Содержательно: в секвенции $\Gamma \mid \Rightarrow \Delta; A \rightarrow B$ формула $A \rightarrow B$ ложна, если формула A истинна, а формула B ложна (в секвенции $\Gamma, A \mid \Rightarrow \Delta; B$).

Утверждение 7. Справедливо обратимое правило введения импликации слева от знака $\mid \Rightarrow$:

$$\frac{\Gamma, B \mid \Rightarrow \Delta \quad \Gamma \mid \Rightarrow \Delta; A}{\Gamma, A \rightarrow B \mid \Rightarrow \Delta} (\rightarrow \mid \Rightarrow).$$

Доказательство. $\Gamma, B \vdash \Delta \quad \Gamma \vdash \Delta; A \leftrightarrow$
 $\Gamma \vdash B \rightarrow \Delta \quad \Gamma \vdash \Delta \vee A \leftrightarrow \Gamma \vdash (B \rightarrow \Delta) \& (\Delta \vee A) \leftrightarrow$
 $\Gamma \vdash (\neg B \vee \Delta) \& (\Delta \vee A) \leftrightarrow \Gamma \vdash (A \rightarrow B) \rightarrow \Delta \leftrightarrow \Gamma, A \rightarrow B \vdash \Delta.$

Замечание. Содержательно: в секвенции $\Gamma, A \rightarrow B \mid \Rightarrow \Delta$ формула $A \rightarrow B$ истинна, если формула A (в секвенции $\Gamma \mid \Rightarrow \Delta; A$) или формула B истинна (в секвенции $\Gamma, B \mid \Rightarrow \Delta$).

Утверждение 8. Справедлива секвенция $\Gamma, A \mid \Rightarrow \Delta; A$.

Доказательство.

$$\Gamma, A \mid \Rightarrow \Delta; A \leftrightarrow \Gamma, A \vdash \Delta \vee A \leftrightarrow \Gamma, A \vdash \neg A \rightarrow \Delta \leftrightarrow$$

$$\Gamma \vdash A \rightarrow (\neg A \rightarrow \Delta) \leftrightarrow \Gamma \vdash A \& \neg A \rightarrow \Delta \leftrightarrow \vdash A \& \neg A \rightarrow \Delta.$$

Последнее справедливо. Поэтому справедлива и исходная секвенция.

Замечание. Утверждение 8 устанавливает тот простой факт, что формула A выводится из формулы A при любых совокупностях Γ и Δ . Секвенции $\Gamma, A \mid \Rightarrow \Delta; A$ примем впоследствии за аксиомы секвенциального исчисления высказываний. Рассматриваемые как формулы логики, опровержимыми (это формулы общезначимые) они не являются.

8.2. Секвенциальное исчисление высказываний

Формальные символы СИВ составляют формальные символы ИВ плюс два дополнительных символа: секвенциальный символ $\mid \Rightarrow$ и ; (точка с запятой). Формулы СИВ совпадают с формулами ИВ. Если Γ и Δ есть конечные совокупности формул СИВ, то выражение $\Gamma \mid \Rightarrow \Delta$ есть секвенция.

Схемы аксиом. Если A есть формула СИВ, то секвенция вида $\Gamma, A \mid \Rightarrow \Delta; A$, есть единственная схема аксиом в СИВ.

Замечание. Так как $A \mid \Rightarrow A \leftrightarrow \vdash A \rightarrow A \leftrightarrow \vdash A \vee \neg A$ (последняя формула есть закон исключенного третьего), то в качестве схемы аксиом в СИВ, в сущности, берем закон исключенного третьего.

Правила секвенциального вывода (Янов Ю. И).

Введение отрицания

$$\frac{\Gamma \mid \Rightarrow \Delta; A}{\Gamma, \neg A \mid \Rightarrow \Delta} (\neg \mid \Rightarrow); \quad \frac{\Gamma, A \mid \Rightarrow \Delta}{\Gamma \mid \Rightarrow \Delta; \neg A} (\mid \Rightarrow \neg).$$

Введение конъюнкции

$$\frac{\Gamma, A, B \mid \Rightarrow \Delta}{\Gamma, A \& B \mid \Rightarrow \Delta} (\& \mid \Rightarrow); \quad \frac{\Gamma \mid \Rightarrow \Delta; A \quad \Gamma \mid \Rightarrow \Delta; B}{\Gamma \mid \Rightarrow \Delta; A \& B} (\mid \Rightarrow \&).$$

Введение дизъюнкции

$$\frac{\Gamma, A \models \Delta \quad \Gamma, B \models \Delta}{\Gamma, A \vee B \models \Delta} \quad (\vee \models); \quad \frac{\Gamma \models \Delta; A; B}{\Gamma \models \Delta; A \vee B} \quad (\models \vee).$$

Введение импликации

$$\frac{\Gamma, B \models \Delta \quad \Gamma \models \Delta; A}{\Gamma, A \rightarrow B \models \Delta} \quad (\rightarrow \models); \quad \frac{\Gamma, A \models \Delta; B}{\Gamma \models \Delta; A \rightarrow B} \quad (\models \rightarrow).$$

Выбор этих правил вывода оправдывается ранее доказанными утверждениями 1–7. Приведенные правила вывода обратимы. Эти правила довольно естественны, тем более что Γ и Δ являются множествами формул.

Определение. Вывод в СИВ есть конечная последовательность секвенций, каждая из которых получена из предыдущих секвенций по одному из правил вывода.

Определение. Секвенция S выводима в СИВ, если в СИВ существует вывод, последней секвенцией которого является S .

Определение. Доказательство в СИВ есть конечная последовательность секвенций, каждая из которых есть либо аксиома, либо получена из предыдущих секвенций по одному из правил вывода.

Определение. Секвенция S доказуема в СИВ, если в СИВ существует доказательство, последней секвенцией которого является S .

Вывод, как и доказательство в СИВ, удобно оформлять в виде дерева. Ниже приведем примеры построения деревьев вывода и доказательства в СИВ.

Определение. Формула A доказуема в СИВ (обозначение: $\vdash_G A$), если в СИВ доказуема секвенция $\models A$ (с пустой левой частью).

Теорема. Если A есть формула в СИВ (A есть формула и в ИВ), то $\vdash_G A \leftrightarrow \vdash A$.

Доказательство. Всякая аксиома в СИВ доказуема в ИВ как закон исключенного третьего. Всякое правило вывода в СИВ доказуемо в ИВ (утверждения 1–7). Поэтому доказуемость формулы (или секвенции) в СИВ влечет доказуемость этой формулы (или формулы ИВ, соответствующей секвенции).

Всякая аксиома в ИВ доказуема в СИВ. Покажем это на примере аксиомы $(x \rightarrow z) \rightarrow ((y \rightarrow z) \rightarrow (x \vee y \rightarrow z))$, построив дерево вывода этой аксиомы в СИВ. В листьях дерева помещены аксиомы СИВ, в корне – доказуемая формула. Вывод осуществляется от листьев к корню (справа от горизонталей указаны применяемые секвенциальные правила вывода):

$$\begin{array}{c} x \models z; x; y \quad y \models z; x; y \\ \hline z, x \vee y \models z; x \quad x \vee y \models z; x; y \quad (\vee \models) \\ \hline z, y \rightarrow z, x \vee y \models z \quad y \rightarrow z, x \vee y \models z; x \quad (\rightarrow \models) \\ \hline x \rightarrow z, y \rightarrow z, x \vee y \models z \quad (\rightarrow \models) \\ \hline x \rightarrow z, y \rightarrow z \models x \vee y \rightarrow z \quad (\models \rightarrow) \\ \hline (x \rightarrow z) \models (y \rightarrow z) \rightarrow (x \vee y \rightarrow z) \quad (\models \rightarrow) \\ \hline \models (x \rightarrow z) \rightarrow ((y \rightarrow z) \rightarrow (x \vee y \rightarrow z)) \quad (\models \rightarrow) \end{array}$$

Алгоритм построения дерева вывода в СИВ для формулы $(x \rightarrow z) \rightarrow ((y \rightarrow z) \rightarrow (x \vee y \rightarrow z))$ выглядит следующим образом. Для формул справа от знака $\models$ будем рассуждать о ложности одной из этих формул, а слева – об истинности. Формула $(x \rightarrow z) \rightarrow ((y \rightarrow z) \rightarrow (x \vee y \rightarrow z))$ ложна, если $x \rightarrow z$ истинно, а $(y \rightarrow z) \rightarrow (x \vee y \rightarrow z)$ ложно. Помещаем формулу $x \rightarrow z$ слева, а формулу $(y \rightarrow z) \rightarrow (x \vee y \rightarrow z)$ справа от знака $\models$. Секвенцию $x \rightarrow z \models (y \rightarrow z) \rightarrow (x \vee y \rightarrow z)$ помещаем на следующем уровне дерева вывода (над исходной формулой). Проведя аналогичное рассуждение для формулы $(y \rightarrow z) \rightarrow (x \vee y \rightarrow z)$, получим секвенцию $x \rightarrow z, y \rightarrow z \models x \vee y \rightarrow z$, которую поместим над предыдущей секвенцией. Эти же рассуждения для формулы $x \vee y \rightarrow z$ приведут нас к секвенции $x \rightarrow z, y \rightarrow z, x \vee y \models z$, которую пометим над предыдущей секвенцией. Формула $x \rightarrow z$ истинна, если z истинно или x ложно. Секвенция $x \rightarrow z, y \rightarrow z, x \vee y \models z$ распадается на две секвенции: $z, y \rightarrow z, x \vee y \models z$ и $y \rightarrow z, x \vee y \models z$; x , которые поместим над предыдущей секвенцией. Секвенция $z, y \rightarrow z, x \vee y \models z$ является аксиомой (по z), и разворачивать дерево над ней не нужно. В секвенции $y \rightarrow z, x \vee y \models z$; x формула $y \rightarrow z$ истинна, если z истинно или y ложно. Секвенции $z, x \vee y \models z$; x и $x \vee y \models z$; x помещаем над секвенцией $y \rightarrow z, x \vee y \models z$; x . Секвенция $z, x \vee y \models z$; x является аксиомой. В секвенции $x \vee y \models z$; x ; y формула $x \vee y$ истинна, если x истинно или y истинно. Секвенции $x \models z$; x ; y и $y \models z$; x ; y помещаем над секвенцией $x \vee y \models z$; x ; y . Обе они аксиомы. Построение дерева вывода закончено. В листьях дерева помещаются аксиомы. Следовательно, построенное дерево вывода является доказательством исходной формулы в СИВ.

Если в построенном для некоторой формулы дереве вывода не во всех листьях помещены аксиомы, то исходная формула общезначимой не является. По формулам секвенций, помещенным в таких листьях, можно найти такие значения переменных, при которых исходная формула ложна (т. е. опровержима). Будем, например, исходить из формулы $(x \rightarrow y \vee t) \rightarrow ((y \rightarrow z) \rightarrow (x \vee y \rightarrow z))$. Построим для нее следующее дерево вывода:

$$\begin{array}{c}
 \frac{y, x \models z; y \quad t, x \models z; y}{y \vee t, x \models z; y \quad x \models z; y; x \quad (\vee \models)} \\
 \frac{x \vee y, x \models z; y \quad x \rightarrow y \vee t, y \models z; y \quad (\rightarrow \models)}{y \rightarrow y \vee t, z, x \vee y \models z \quad y \rightarrow y \vee t, x \vee y \models z; y \quad (\vee \models)} \\
 \frac{x \rightarrow y \vee t, y \rightarrow z, x \vee y \models z \quad (\rightarrow \models)}{x \rightarrow y \vee t, y \rightarrow z \models x \vee y \rightarrow z \quad (\models \rightarrow)} \\
 \frac{(x \rightarrow y \vee t) \models (y \rightarrow z) \rightarrow (x \vee y \rightarrow z) \quad (\models \rightarrow)}{\models (x \rightarrow y \vee t) \rightarrow ((y \rightarrow z) \rightarrow (x \vee y \rightarrow z)) \quad (\models \rightarrow)}
 \end{array}$$

В этом дереве листовая секвенция $t, x \models z; y$ дальнейшего упрощения не допускает и (единственная из всех листовых секвенций) не является аксиомой. При $t = 1, x = 1, z = 0, y = 0$ исходная формула будет ложной (опровержимой). На остальных наборах истинна. Корневая секвенция (а потому и исходная формула) в СИВ не доказуема.

В сущности, при построении дерева вывода в СИВ мы пытаемся построить опровержение исходной формулы. При этом либо в листьях окончательного дерева окажутся только аксиомы, и тогда построенное дерево вывода даст доказательство исходной формулы, либо в одном из листьев поместится секвенция, не являющаяся аксиомой, и тогда по этой секвенции можно построить опровержение исходной формулы, которая доказуемой в СИВ не является.

Правило заключения в ИВ эквивалентно выводимости $A, A \rightarrow B \vdash B$, что, в свою очередь, эквивалентно секвенции $A, A \rightarrow B \vdash B$. Выведем эту секвенцию в СИВ. Ниже следует соответствующее дерево вывода:

$$\frac{A, B \vdash B \quad A \vdash B; A}{A, A \rightarrow B \vdash B} (\rightarrow \vdash).$$

Классы формул, доказуемых в СИВ и ИВ, совпадают.

8.3. Секвенциальное исчисление предикатов

Формальные символы секвенционального исчисления предикатов (СИП) совпадают с формальными символами логики предикатов. Формулы СИП совпадают с формулами ИП.

Пусть A, B есть произвольные формулы СИП; Γ, Δ есть произвольные конечные совокупности формул СИП. Секвенция (в СИП) есть выражение вида $\Gamma \vdash \Delta$. Единственной схемой аксиом в СИП является схема аксиом $\Gamma, A \vdash \Delta; A$.

Правила вывода СИП составляют правила вывода СИВ (над формулами из СИП) плюс следующие дополнительные обратимые правила вывода, связанные с обработкой кванторов:

$$\frac{\Gamma \vdash \Delta; A(y)}{\Gamma \vdash \Delta; (\forall x)A(x)} (\vdash \forall); \quad \frac{\Gamma, A(y) \vdash \Delta}{\Gamma, (\exists x)A(x) \vdash \Delta} (\exists \vdash);$$

переменная y не входит в нижнюю секвенцию свободно;

$$\frac{\Gamma, A(t), (\forall x)A(x) \vdash \Delta}{\Gamma, (\forall x)A(x) \vdash \Delta} (\forall \vdash); \quad \frac{\Gamma \vdash \Delta; A(t), (\exists x)A(x)}{\Gamma \vdash \Delta; (\exists x)A(x)} (\vdash \exists);$$

переменная x не входит в верхнюю секвенцию свободно.

Определение. Доказательство в СИП есть конечная последовательность секвенций, каждая из которых является либо аксиомой, либо получена из предыдущих секвенций последовательности по одному из правил вывода.

Определение. Секвенция S доказуема в СИП, если существует доказательство, последней секвенцией которого является S .

Определение. Формула A доказуема в СИП, если в СИП доказуема секвенция $\vdash A$.

Исчисление секвенций предложил Гентцен. Приводим его в той редакции, которую ему придал Кангер. Правила вывода Кангера рассчитаны на то, чтобы, отправляясь от исходной формулы и опираясь на естественный смысл ло-

гических операций и кванторов при построении дерева вывода, попытаться построить опровержение для исходной формулы. Либо это опровержение построим, либо полученное дерево будет доказательством исходной формулы.

Пример. Докажем в СИП формулу

$$(\exists x)(P(a) \rightarrow Q(x)) \rightarrow (P(a) \rightarrow (\exists x)Q(x)).$$

Доказательство этой формулы оформим в виде дерева (рассуждая при этом так же, как это делали в случае вывода формул в СИП):

$$\begin{array}{c}
 Q(b), P(a) \models Q(b); (\exists x)Q(x) \\
 \hline
 Q(b), P(a) \models (\exists x)Q(x) \quad P(a) \models (\exists x)Q(x); P(a) \quad (\models \exists) \\
 \hline
 P(a) \rightarrow Q(b), P(a) \models (\exists x)Q(x) \quad (\rightarrow \models) \\
 \hline
 (\exists x)(P(a) \rightarrow Q(x)), P(a) \models (\exists x)Q(x) \quad (\exists \models) \\
 \hline
 (\exists x)(P(a) \rightarrow Q(x)) \models P(a) \rightarrow (\exists x)Q(x) \quad (\models \rightarrow) \\
 \hline
 \models (\exists x)(P(a) \rightarrow Q(x)) \rightarrow (P(a) \rightarrow (\exists x)Q(x)) \quad (\models \rightarrow)
 \end{array}$$

Прокомментируем этот вывод для правил, связанных с кванторами. В секвенции $(\exists x) (P(a) \rightarrow Q(x)), P(a) \models (\exists x) Q(x)$ формула $(\exists x) (P(a) \rightarrow Q(x))$ истинна, если существует x , например, равный предмету b , отличному от всех ранее встречавшихся предметов, для которого формула $P(a) \rightarrow Q(b)$ истинна. Переходим к секвенции $P(a) \rightarrow Q(b), P(a) \models (\exists x)Q(x)$. В секвенции $Q(b), P(a) \models (\exists x)Q(x)$ формула $(\exists x)Q(x)$ ложна, если формула $\neg(\exists x)Q(x)$ истинна, т. е. для всякого x , в том числе и для x , равного b , формула $Q(b)$ ложна. От секвенции $Q(b), P(a) \models (\exists x)Q(x)$ переходим к секвенции $Q(b), P(a) \models Q(b); (\exists x)Q(x)$, которая является аксиомой. В построенном дереве все листья есть аксиомы, потому исходная формула (являясь общезначимой) доказуема в СИП.

Пример. Попытаемся доказать в СИП формулу

$$(\exists x)(P(a) \rightarrow Q(x)) \rightarrow (P(a) \rightarrow (\forall x)Q(x)).$$

Вывод оформим в виде следующего дерева:

$$\begin{array}{c}
 Q(b), P(a) \models Q(c) \\
 \hline
 Q(b), P(a) \models (\forall x)Q(x) \quad P(a) \models (\forall x)Q(x); P(a) \quad (\models \forall) \\
 \hline
 P(a) \rightarrow Q(b), P(a) \models (\forall x)Q(x) \quad (\rightarrow \models) \\
 \hline
 (\exists x)(P(a) \rightarrow Q(x)), P(a) \models (\forall x)Q(x) \quad (\exists \models) \\
 \hline
 (\exists x)(P(a) \rightarrow Q(x)) \models P(a) \rightarrow (\forall x)Q(x) \quad (\models \rightarrow) \\
 \hline
 \models (\exists x)(P(a) \rightarrow Q(x)) \rightarrow (P(a) \rightarrow (\forall x)Q(x)) \quad (\models \rightarrow)
 \end{array}$$

В этом выводе в секвенции $Q(b), P(a) \models (\forall x)Q(x)$ формула $(\forall x)Q(x)$ ложна, если истинна формула $\neg(\forall x)Q(x)$, т. е. существует x , вообще говоря, отличный от всех ранее встречавшихся предметов, равный, например, предмету c , для которого $\neg Q(c)$ истинно, т. е. $Q(c)$ ложно. От секвенции $Q(b), P(a) \models (\forall x)Q(x)$ переходим к секвенции $Q(b), P(a) \models Q(c)$, дальнейшее упрощение которой невозможно и которая не является аксиомой. По этой секвенции строим опро-

вержение исходной формулы. Полагаем множество $M = \{a, b, c\}$, $P(a) = И$, $Q(b) = И$, $Q(c) = Л$. В остальном предикаты P и Q определяем произвольно. На интерпретации $I = (M, P, Q)$ исходная формула превращается в ложное высказывание. Интерпретация I и есть опровержение исходной формулы. Следовательно, исходная формула общезначимой не является и в СИП не доказуема.

Теорема. Классы формул, доказуемых в СИП и УИП, совпадают.

Доказательство. Покажем, что всякая формула, доказуемая в УИП, доказуема и в СИП. Дерево вывода

$$\frac{\frac{A(t), (\forall x)A(x) \vdash A(t)}{(\forall x)A(x) \vdash A(t)} \quad (\forall \vdash)}{\vdash (\forall x)A(x) \rightarrow A(t)} \quad (\vdash \rightarrow)$$

является доказательством в СИП аксиомы $(\forall x)A(x) \rightarrow A(t)$. Дерево вывода

$$\frac{\frac{A(t) \vdash A(t); (\exists x)A(x)}{A(t) \vdash (\exists x)A(x)} \quad (\vdash \exists)}{\vdash A(t) \rightarrow (\exists x)A(x)} \quad (\vdash \rightarrow)$$

является доказательством в СИП аксиомы $A(t) \rightarrow (\exists x)A(x)$. Правила вывода в УИП являются правилами вывода и в СИП. Покажем теперь, что всякая формула, доказуемая в СИП, доказуема и в УИП. Действительно, всякая аксиома $\Gamma, A \vdash \Delta; A$ в СИП соответствует доказуемой в УИП формуле $\Gamma \& A \rightarrow \Delta \vee A$. Можно показать, что (обратимые) правила вывода в СИП доказуемы и в УИП (если секвенцию $A_1, A_2, \dots, A_n \vdash B_1; B_2; \dots; B_m$ рассматривать как формулу $A_1 \& A_2 \& \dots \& A_n \rightarrow B_1 \vee B_2 \vee \dots \vee B_m$).

Следствие. Класс формул, доказуемых в СИП, совпадает с классом общезначимых формул. Естественный вывод Гентцена довольно удобен при вычислении общезначимости формулы логики предикатов. Его элементы используются в логическом программировании. Но первые языки программирования были построены в логике предикатов с правилом резолюции Дж. Робинсона, компьютерная реализация которого – алгоритмический язык программирования Пролог – оказалась достаточно эффективной.



Метод резолюций в логике предикатов и Пролог



9.1. Метод резолюций в логике высказываний

Построим формально-логическое исчисление, в котором описываются все невыполнимые формулы логики и только они. Понятия алфавита, формулы, подформулы, интерпретации, модели, выполнимости, невыполнимости, общезначимости, опровержимости совпадают с аналогичными понятиями в исчислении высказываний.

Для удобства конъюнкцию нескольких формул будем задавать так же, как множество формул, составляющих эту конъюнкцию, и наоборот, считать множество формул конъюнкцией этих формул, т. е. формулу $A_1 \& A_2 \& \dots \& A_n$ будем задавать как множество $\{A_1, A_2, \dots, A_n\}$.

Пусть M есть конечное множество формул (т. е. их конъюнкция), все переменные которых содержатся среди $p_1, p_2, \dots, p_n$. Как и прежде, пишем в этом случае $M(p_1, p_2, \dots, p_n)$.

Понятие интерпретации множества формул M совпадает с интерпретацией формулы, являющейся конъюнкцией формул из M . Основные свойства множества формул определяются следующим образом.

1. Множество формул общезначимо, если общезначимы все формулы из M .
2. Множество формул выполнимо, если существует интерпретация, на которой выполнимы все формулы из M .
3. Множество формул невыполнимо, если для любой интерпретации найдется формула из M , ложная на этой интерпретации.
4. Множество формул опровержимо, если существует интерпретация, на которой хотя бы одна из формул в M ложна.

Формула B есть логическое следствие формул $A_1, A_2, \dots, A_n$, если формула $A_1 \& A_2 \& \dots \& A_n \rightarrow B$ общезначима.

Формула B есть логическое следствие формул $A_1, A_2, \dots, A_n$, если и только если формула $A_1 \& A_2 \& \dots \& A_n \& \neg B$ (эквивалентная отрицанию формулы $A_1 \& A_2 \& \dots \& A_n \rightarrow B$) невыполнима.

Литера есть элементарная формула (атом) или ее отрицание. Литеры p и $\neg p$ называются *контрарной парой*. *Дизъюнкт* есть дизъюнкция конечного числа литер. Для удобства дизъюнкт иногда задают как множество литер. Например, дизъюнкт $p \vee q \vee \neg r$ можно задать как $\{p, q, \neg r\}$. Обратно, конечное множество дизъюнктов можно задавать как их дизъюнкцию. При этом дизъюнкцию нескольких дизъюнктов предполагаем заданной без повторов и не зависящей от порядка ее дизъюнктивных слагаемых.

Примеры дизъюнктов. $p \vee \neg q$; $p \vee \neg q \vee r$; $\{p, \neg q, r\}$; $\neg p \vee q \vee \neg r \vee s$; $\{\neg p, q, \neg r, s\}$.

n -литерный дизъюнкт содержит n литер. Однолитерный дизъюнкт называется *единичным дизъюнктом*. *Пустой дизъюнкт* $\square$ по определению тождественно ложен. Его можно понимать как невыполнимую формулу. Множество дизъюнктов невыполнимо, если на каждой интерпретации ложен хотя бы один дизъюнкт.

Пример. Множество дизъюнктов $S = \{p \vee \neg q \vee \neg r, \neg p \vee q \vee \neg r\}$ выполнимо, ибо конъюнкция $(p \vee \neg q \vee \neg r) \& (\neg p \vee q \vee \neg r)$ истинна при интерпретации $I = (p = И, q = И, r = Л)$. Множество дизъюнктов $S = \{p \vee q, p \vee \neg q, \neg p \vee q, \neg p \vee \neg q\}$ невыполнимо, ибо их конъюнкция тождественно ложна.

Дизъюнкт, содержащий контрарную пару, есть тавтология, ибо он истинен при любых интерпретациях.

9.1.1. Семантическое дерево

Пусть $S = S(p_1, p_2, \dots, p_n)$ есть множество дизъюнктов. Задать интерпретацию для S значит задать истинностные значения пропозициональных переменных в S . Для переменной p положим p^1 равным p , а p^0 – равным $\neg p$. Пусть набору литер $(p_1^c, \dots, p_n^c)$ соответствует интерпретация $I = (c_1, \dots, c_n)$; при этом 1 отождествляется с истиной (И), а 0 – с ложью.

Определение. Пусть $S = S(p_1, \dots, p_n)$ есть множество дизъюнктов. *Полное семантическое дерево* T для S есть конечное бинарное дерево, в i -м ярусе которого из каждого узла N исходят два ребра, одно из которых помечено литерой p_i , а другое – литерой $\neg p_i$.

Всякой ветви дерева (ее длина равна n) можно поставить в соответствие некоторую интерпретацию для S следующим образом. Если ребра некоторой ветви в T помечены литерами $(p_1^c, \dots, p_n^c)$, то эта ветвь задает интерпретацию $I = \{c_1, \dots, c_n\}$. Семантическое дерево T исчерпывает все 2^n интерпретаций.

Определение. Узел N в семантическом дереве T для S называется *опровергающим*, если путь в T от корня дерева до N задает такие истинностные значения переменных p_i , при которых опровергается некоторый дизъюнкт C из S , но при этом никакой другой узел от корня до N свойством опровержимости не обладает.

Множество дизъюнктов S невыполнимо, если и только если любая ветвь в семантическом дереве T для S содержит опровергающий узел.

Обрежем семантическое дерево T для S , удалив из T все поддеревья с корнями в опровергающих узлах, оставляя опровергающие узлы в T . Полученное дерево называется *замкнутым семантическим деревом* для S .

Множество дизъюнктов S невыполнимо тогда и только тогда, когда в замкнутом семантическом дереве для S все его концевые узлы (листья) являются опровергающими.

Пример 1. Пусть дано множество дизъюнктов $S = \{p, \neg p \vee q, \neg q \vee r, \neg q \vee \neg r\}$. Полное семантическое дерево для S приведено на рис. 6.1а. Его узлы перенумерованы. Узлы с 7 по 14 концевые. Узел 0 есть корень дерева. Путь от корня 0 до концевого узла 10, помеченный литерами $p, \neg q, \neg r$, дает интерпретацию $I = (И, Л, Л)$; путь от корня до узла 12 – интерпретацию $I = (Л, И, Л)$.

Замкнутое семантическое дерево для S приведено на рис. 6.1б. В концевом узле 4 при значениях переменных $p = И, q = Л$ опровержим дизъюнкт $\neg p \vee q$. В узле 6 опровержим дизъюнкт $\neg q \vee r$. Все концевые узлы (2, 4, 7, 8) в T являются опровергающими (они помечены крестами). Поэтому множество дизъюнктов S невыполнимо.

Пример 2. Множество дизъюнктов $S = \{p, \neg p \vee q, \neg q \vee \neg r\}$. Полное семантическое дерево для S (см. рис. 9.1а) совпадает с семантическим деревом для S из предыдущего примера. На рис. 9.1б приведено замкнутое семантическое дерево T для S . Не все его концевые узлы являются опровергающими, а именно в узле 8 при $p = И, q = И, r = Л$ все дизъюнкты из S истинны. Поэтому множество дизъюнктов S невыполнимым не является (т. е. S выполнимо).

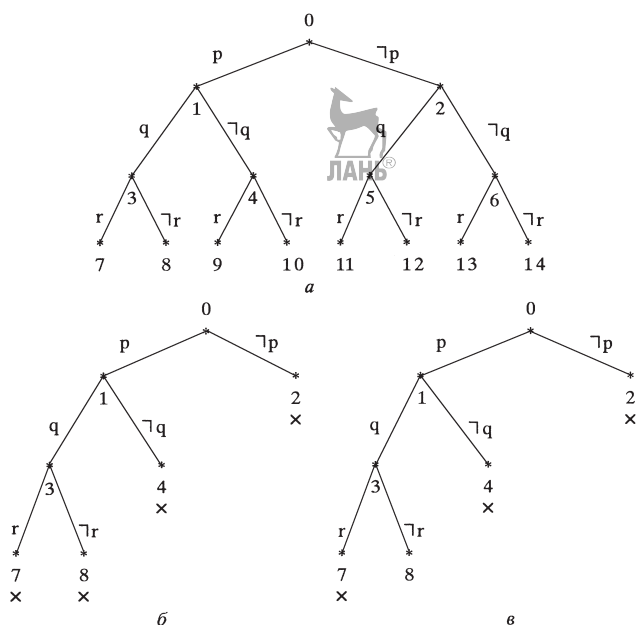


Рис. 9.1

9.1.2. Правило резолюции

Определение. Пусть C_1 и C_2 есть дизъюнкты (рассматриваемые как множества литер); p есть пропозициональная переменная, причем $p \in C_1$, $\neg p \in C_2$. Тогда дизъюнкт $C = (C_1 - p) \cup (C_2 - \neg p)$ называется *резольвентой* дизъюнктов C_1 и C_2 . Правило получения резольвенты дизъюнктов C_1 и C_2 называется *правилом резолюции*. Литеры p и $\neg p$ называются *отрезаемыми*.

Не будем в дальнейшем указывать, рассматриваются ли дизъюнкты C_1 и C_2 как формулы $C' \vee p$ и $C'' \vee \neg p$ соответственно или как множество литер. Это будет видно из контекста.

Пусть равенство $PR(C_1, C_2)$ означает, что дизъюнкт C получен из дизъюнктов C_1 и C_2 по правилу резолюции.

Пример 1. Пусть даны следующие дизъюнкты. $C_1: \neg p \vee q$. $C_2: \neg p \vee r$. Так как в C_1 нет литеры, контрарной какой-либо литере в C_2 , то не существует никакой резольвенты C_1 и C_2 .

Пример 2. Пусть даны следующие дизъюнкты. $C_1: p \vee r$. $C_2: \neg p \vee q$.

Дизъюнкт C_1 имеет литеру p , которая контрарна литере $\neg p$ в C_2 . Отрезая литеры p и $\neg p$ из C_1 и C_2 соответственно, получаем новый дизъюнкт $r \vee q$, являющийся резольвентой C_1 и C_2 .

В дальнейшем мы будем рассматривать резольвенту либо как формулу, либо как множество литер. Во втором случае мы записали бы пример следующим образом: $C_1: \{p, r\}$. $C_2: \{\neg p, q\}$. Резольвентой C_1 и C_2 является множество $\{r, q\}$.

Замечание. Правило резолюции Робинсона обобщает правило заключения (modus ponens). Правило заключения $\frac{p, p \rightarrow q}{q}$, эквивалентное правилу

$\frac{p, \neg p \vee q}{q}$, может рассматриваться как частный случай правила резолюции, где дизъюнкты $C_1 = \neg p \vee q$ и $C_2 = p$, а резольвентой C_1 и C_2 является дизъюнкт q .

Определение. Резолютивный вывод дизъюнкта C из множества дизъюнктов S есть конечная последовательность дизъюнктов, каждый из которых есть либо дизъюнкт из S , либо получен из предыдущих дизъюнктов последовательности по правилу резолюции. Дизъюнкт C выводим из множества дизъюнктов S (обозначение $S \vdash C$), если существует вывод из S , последний дизъюнкт которого есть C .

Теорема. Если дизъюнкт C выводится из множества дизъюнктов S , то формула $S \rightarrow C$ общезначима.

Доказательство. Индукция по числу k применений правила резолюции.

Базис. $k = 1$. Резольвента C дизъюнктов C_1 и C_2 есть логическое следствие из C_1 и C_2 , ибо если $C_1 = C' \vee p$, $C_2 = C'' \vee \neg p$ (и тогда формула $C' \vee C''$ есть резольвента для C_1 и C_2), то формула

$$(C' \vee p) \& (C'' \vee \neg p) \rightarrow (C' \vee C'')$$

общезначима.

В самом деле, предположим, что C_1 и C_2 истинны в интерпретации I . Покажем, что C истинна в I . Для любой интерпретации I либо p , либо $\neg p$ ложно в I .

Пусть для определенности p ложно в I . Тогда C_1 не должен быть единичным дизъюнктом, ибо иначе C_1 был бы ложен в I . Следовательно, C' истинен в I , а потому и формула $C' \vee C''$ истинна в интерпретации I . Аналогично, если $\neg p$ ложно в интерпретации I , то C'' истинен в I , а потому формула $C' \vee C''$ истинна в интерпретации I . Тогда формула $C' \vee C''$ истинна в интерпретации I , а следовательно, формула $C_1 \& C_2 \rightarrow C$ общезначима.

Предположение индукции. Предположим, что теорема справедлива для всякого вывода, число применений правила резолюций в котором меньше k .

Шаг индукции. Покажем, что теорема справедлива для всякого вывода, число применений правила резолюций в котором равно k . Пусть $C_1, C_2, \dots, C_i, \dots, C_j, \dots, C_n$ есть вывод длины k , где $C_k = C$, причем $C_k = \text{ПР}(C_i, C_j)$. Так как длины выводов $C_1, C_2, \dots, C_i$ и $C_1, C_2, \dots, C_j, \dots, C_j$ меньше k , то по предположению индукции формулы $S \rightarrow C_i$ и $S \rightarrow C_j$ общезначимы. Тогда общезначима и формула $S \rightarrow C_i \& C_j$. Так как $C_k = \text{ПР}(C_i, C_j)$, то по базису формула $C_i \& C_j \rightarrow C_k$ общезначима. Следовательно, по правилу силлогизма общезначима формула $S \rightarrow C_k$.

Шаг индукции установлен. Теорема доказана.

Теорема (о полноте). Непустое множество дизъюнктов S невыполнимо тогда и только тогда, когда из S выводится пустой дизъюнкт.

Доказательство. Необходимость. Пусть множество дизъюнктов S невыполнимо. Пусть T есть полное семантическое дерево для S . Так как S невыполнимо, то в замкнутом семантическом дереве T' для S все концевые узлы опровергающие.

Утверждение. Пустой дизъюнкт выводится из S .

Доказательство. Индукции по числу k всех узлов в T' при любом S .

Базис. $k = 1$, т. е. T' имеет единственный (опровергающий) узел. Так как множество S не пусто и невыполнимо, то S содержит пустой дизъюнкт, который выводится из S .

Предположение индукции. Допустим, что доказываемое утверждение справедливо для любого S , для которого замкнутое семантическое дерево T' имеет число узлов меньше k .

Шаг индукции. Покажем, что доказываемое утверждение справедливо для любого S , а потому и для любого семантического дерева T , для которого замкнутое семантическое дерево T' имеет k узлов.

Пусть N есть узел, предшествующий двум опровергающим (концевым) узлам N_1 и N_2 в T' . Пусть исходящим из N ребрам приписаны литеры L_k и $\neg L_k$. Пусть пути от корня до N_1 и N_2 помечены литерами

$$L_1, L_2, \dots, L_{k-1}, L_k;$$

$$L_1, L_2, \dots, L_{k-1}, \neg L_k$$

соответственно. Пусть C_1 и C_2 есть дизъюнкты из S , опровержимые в N_1 и N_2 соответственно. В узле N эти дизъюнкты не опровергаются. Поэтому L_k входит в C_1 , а $\neg L_k$ в C_2 . Отрезая контрарную пару L_k и $\neg L_k$, получим резольвенту $C = (C_1 - L_k) \cup (C_2 - \neg L_k)$.

Так как

$$C_1 = L_1^1 \vee L_2^1 \vee L_{r_1}^1 \vee \neg L_k \text{ ложно в } N_1,$$

$$C_2 = L_1^2 \vee L_2^2 \vee L_{r_2}^2 \vee L_k \text{ ложно в } N_2,$$

то резольвента

$$C = L_1^1 \vee L_2^1 \vee \dots \vee L_{r_1}^1 \vee L_1^2 \vee L_2^2 \vee \dots \vee L_{r_2}^2,$$

не содержащая L_k и $\neg L_k$, ложна в N . Пусть $S_1 = S \cup \{C\}$. Множество S_1 невыполнимо, ибо невыполнимо его подмножество S . Полное семантическое дерево для S_1 совпадает с полным семантическим деревом для S . Дерево T_1' , полученное из T' удалением узлов N_1 и N_2 , содержит в себе замкнутое семантическое дерево T_1 для S_1 , все концевые узлы которого опровергающие. Так как число узлов в T_1 меньше k , то по предположению индукции из множества дизъюнктов S_1 выводится пустой дизъюнкт. Следовательно, пустой дизъюнкт выводится также из S , так как S_1 получается из S однократным применением правила резолюции. Утверждение, а вместе с ним и необходимость теоремы доказаны.

Достаточность. Пусть из множества дизъюнктов S выводится пустой (ложный) дизъюнкт, т. е. $S \vdash \square$. Тогда формула $S \rightarrow \square$ общезначима, а потому множество дизъюнктов S невыполнимо.

Теорема доказана.

Пример 1. Множество дизъюнктов

$$S = \{p \vee q \vee r, p \vee q \vee \neg r, p \vee \neg q, \neg p \vee \neg q, \neg p \vee q\}.$$

Строим для S замкнутое семантическое дерево T (рис. 9.2а).

Все его концевые узлы опровергающие. В узле 3 опровергается дизъюнкт $\neg p \vee \neg q$, в узле 4 – дизъюнкт $\neg p \vee q$, в узле 5 – дизъюнкт $p \vee \neg q$, в узле 13 – дизъюнкт $p \vee q \vee \neg r$, а в узле 14 – дизъюнкт $p \vee q \vee r$. Поэтому множество дизъюнктов S невыполнимо. Вывод пустого дизъюнкта из S имеет следующий вид.

- (1) $p \vee q \vee r$, из S , опровергается в узле 8,
- (2) $p \vee q \vee \neg r$, из S , опровергается в узле 7,
- (3) $p \vee q$, ПР(1,2), опровергается в узле 6,
- (4) $p \vee \neg q$, из S , опровергается в узле 5,
- (5) p , ПР(3,4), опровергается в узле 2,
- (6) $\neg p \vee \neg q$, из S , опровергается в узле 3,
- (7) $\neg p \vee q$, из S , опровергается в узле 4,
- (8) $\neg p$, ПР(6,7), опровергается в узле 1,
- (9) $\square$, ПР(5,8), опровергается в узле 0.

Пример 2. Множество дизъюнктов $S = \{\neg p \vee \neg q, \neg p \vee q \vee \neg r, p \vee \neg q\}$. Строим для S замкнутое семантическое дерево T (рис. 9.2б).

Не все его концевые узлы опровергающие (узлы 8, 9, 10). Поэтому множество дизъюнктов S невыполнимым не является, и потому пустой дизъюнкт из S не выводим.

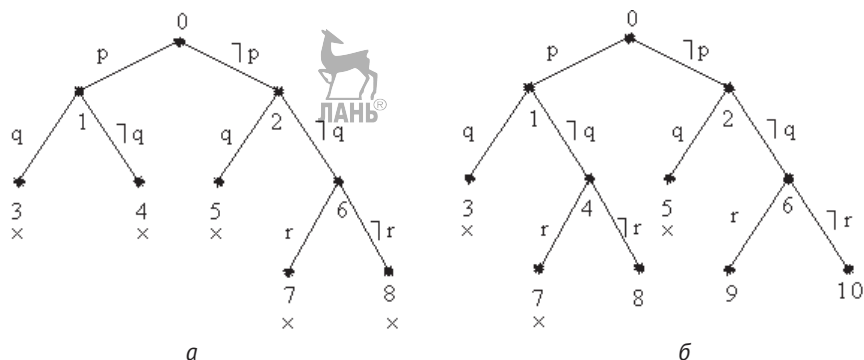


Рис. 9.2

Теорема. Пусть S есть некоторое множество дизъюнктов. Следующие утверждения эквивалентны.

1. Множество дизъюнктов S невыполнимо.
2. Пустой дизъюнкт выводим из S .
3. Формула $S \rightarrow \square$ общезначима.

9.2. Эрбрановы универсум, базис, интерпретация

Переходим к изложению метода резолюций в логике предикатов. Понятия алфавита, терма, формулы, подформулы, интерпретации, общезначимости, выполнимости, невыполнимости, опровержимости совпадают с аналогичными понятиями логики предикатов. Атомарная формула (или атом) строится из предикатного символа и одного или нескольких термов.

Как и в исчислении высказываний, формула B логики предикатов есть логическое следствие формул $A_1, A_2, \dots, A_k$, если формула $A_1 \& A_2 \& \dots \& A_k \rightarrow B$ общезначима.

Определение. Литера есть атом или его отрицание. Дизъюнкт есть дизъюнкция конечного числа литер.

Определение. Пусть S есть некоторое множество дизъюнктов и H_0 есть множество констант в S . Если констант в множестве S нет, то $H_0 = \{a\}$, где a есть символ некоторой константы, не содержащейся в S . Далее полагаем $H_{i+1} = H_i \cup \{f(t_1, \dots, t_k) : f \text{ есть функциональный символ из } S, a \ t_1, t_2, \dots, t_k \in H_i, i = 0, 1, 2, \dots\}$. Множество $H = \bigcup_{i=0}^{\infty} H_i$ называется эрбрановым универсумом множества дизъюнктов S .

Замечание. Эрбранов универсум есть множество всех термов сигнатуры для S , не имеющих предметных переменных.

Примеры.

1. $S = \{P(a), \neg P(x) \vee P(f(x))\}$. Множество $\{a, f\}$ составляет формальные символы (сигнатуру) для построения эрбранова универсума:

$H_0 = \{a\};$
 $H_1 = \{a, f(a)\};$
 $H_2 = \{a, f(a), f(f(a))\};$
 $\dots$
 $H = \{a, f(a), f(f(a)), f(f(f(a))), \dots\}.$

2. $S = \{P(x) \vee Q(x), R(z), T(y) \vee W(y)\}.$

$H_0 = \{a\} = H_1 = H_2 = \dots = H.$

3. $S = \{P(f(x), a, g(y), b) \vee Q(x)\}.$ Множество $\{a, b, f, g\}$ составляет формальные символы (сигнатуру) для построения эрбранова универсума:

$H_0 = \{a, b\};$
 $H_1 = \{a, b, f(a), f(b), g(a), g(b)\};$
 $\dots$
 $H = \{a, b, f(a), f(b), g(a), g(b), f(f(a)), f(f(b)), f(g(a)), f(g(b)), g(f(a)), g(f(b)), g(g(a)), g(g(b)), f(f(f(a))), \dots\}.$

Определение. Эрбранов базис для множества дизъюнктов S есть множество всех атомов $B = \{P(t_1, t_2, \dots, t_k) : P \text{ есть предикатный символ из } S, a, t_1, t_2, \dots, t_k \text{ есть элементы эрбранова универсума}\}.$

Замечание. Эрбранов базис B множества S составляют элементарные формулы (атомы) сигнатуры для S , в которых отсутствуют предметные переменные.

Пример. $S = \{P(f(x), a, g(y), b) \vee Q(x)\}.$ Множество $\{a, b, f, g, P, Q\}$ составляет формальные символы (сигнатуру) для построения эрбранова базиса, который является множеством $B = \{P(t_1, t_2, t_3, t_4), Q(t) : t_1, t_2, t_3, t_4, t \text{ пробегают элементы эрбранова универсума}\}.$

Определение. Основной пример дизъюнкта C из множества дизъюнктов S есть дизъюнкт, полученный заменой всех предметных переменных в C на элементы эрбранова универсума.

Пример. Множество дизъюнктов $S = \{P(x), Q(f(y)) \vee R(y)\}.$

Эрбранов универсум $H = \{a, f(a), f(f(a)), f(f(f(a))), \dots\}.$

Пусть дизъюнкт C есть $P(x)$. Тогда $P(a), P(f(a)), P(f(f(a))), \dots$ есть основные примеры дизъюнкта C .

Пусть теперь в качестве C взят дизъюнкт $Q(f(y)) \vee R(y)$. Тогда

$Q(f(a)) \vee R(a), Q(f(f(a))) \vee R(f(a)), Q(f(f(f(a)))) \vee R(f(f(a))), \dots$

есть основные примеры дизъюнкта C .

Замечание. Основной пример дизъюнкта C не имеет предметных переменных. Он получается замещением предметных переменных в C некоторыми термами из эрбранова универсума.

В дальнейшем в сигнатуру формулы (множества формул) не будем включать ее свободные предметные переменные. Предполагаем, что формула их не содержит. Свободные переменные, если они все же имеются, рассматриваются как предметные константы.

Определение. Эрбранова интерпретация (Н-интерпретация) множества дизъюнктов S есть всякая интерпретация вида

$$IH = (H, P_1, \dots, P_n, f_1, \dots, f_m, a_1, \dots, a_r),$$

где

- 1) H есть эрбранов универсум для S ;
- 2) $(P_1, \dots, P_n, f_1, \dots, f_m, a_1, \dots, a_r)$ есть сигнатура для S (т. е. множество всех предикатных, функциональных, константных букв из S ; множество свободных предикатных переменных в сигнатуру для S не входит);
- 3) для всяких термов $t_1, \dots, t_k$ из H функция $f(t_1, \dots, t_k) = f(t_1, \dots, t_k)$, т. е. значение $f(t_1, \dots, t_k)$ есть слово в алфавите, из которого строятся термы эрбранова универсума H ;
- 4) значения предикатов P_i на термах из H не ограничиваются ничем;
- 5) всякое $a_i = a_i$.

Замечание. Пусть A есть замкнутая формула логики предикатов; IH есть эрбранова интерпретация для A ; $(\forall x_1) \dots (\forall x_k) B$ есть стандартная форма Скулема для A ; $C_1 \& \dots \& C_s$ есть конъюнктивная нормальная форма для B . Тогда следующие утверждения эквивалентны.

1. Формула A не выполнима в IH .
2. Формула $(\forall x_1) \dots (\forall x_k) B$ не выполнима в IH .
3. Формула $(\forall x_1) \dots (\forall x_k) (C_1 \& \dots \& C_s)$ не выполнима в IH .
4. Формула $(\forall x_1) \dots (\forall x_k) C_1 \& \dots \& (\forall x_1) \dots (\forall x_k) C_s$ не выполнима в IH .
5. Некоторый сомножитель $(\forall x_1) \dots (\forall x_k) C_i$ не выполним в IH .
6. Формула $\neg (\forall x_1) \dots (\forall x_k) C_i$ выполнима в IH .
7. Формула $(\exists x_1) \dots (\exists x_k) \neg C_i(x_1, \dots, x_k)$ выполнима в IH .
8. Существуют $x_1 = t_1 \in H, \dots, x_k = t_k \in H$, для которых основной пример $C_i(t_1, \dots, t_k)$ (обозначим его через C') дизъюнкта $C_i(x_1, \dots, x_k)$ (обозначим его через C) ложен в IH .
9. Существует основной пример C' некоторого дизъюнкта C , ложный в IH . При этом дизъюнкт C есть некоторая дизъюнкция литер $L_1(x_1, \dots, x_k) \vee \dots \vee L_u(x_1, \dots, x_k)$, основной пример C' есть дизъюнкция элементов эрбранова базиса $L_1(t_1, \dots, t_k) \vee \dots \vee L_u(t_1, \dots, t_k)$.

Теорема. Множество дизъюнктов S невыполнимо тогда и только тогда, когда S невыполнимо во всех Н-интерпретациях.

Доказательство. Если S невыполнимо, то S ложно во всех интерпретациях, в том числе и во всех Н-интерпретациях. Пусть теперь S ложно во всех Н-интерпретациях. Пусть

- $P_1, P_2, \dots$, все предикатные буквы, встречающиеся в S ;
- $f_1, f_2, \dots$, все функциональные буквы, встречающиеся в S ;
- $a_1, a_2, \dots$, все константные буквы, встречающиеся в S .

Допустим противное: множество дизъюнктов S выполнимо, и пусть $I = (M, P_1, \dots, f_1, \dots, a_1, \dots)$ есть интерпретация, в которой S истинно. Построим отображение $\varphi: H \rightarrow M$, положив $\varphi(a_i) = a_i = a_i$; $\varphi(f_i(t_1, \dots, t_k)) = f_i(\varphi(t_1), \dots, \varphi(t_k))$, т. е. заменив

в терме из H все функциональные и константные символы из H на функции и константы из I . Построим H -интерпретацию

$$IH = (H, Q_1, \dots, g_1, \dots, b_1, \dots), \text{ положив} \\ Q_i(t_1, \dots, t_{k_i}) = I \leftrightarrow P_i(\varphi(t_1), \dots, \varphi(t_{k_i})) = I.$$

В построенной H -интерпретации IH множество S ложно, т. е. существует основной пример S' некоторого дизъюнкта S из S , ложный в IH . Пусть дизъюнкт $S = L_1 \vee \dots \vee L_u$ и литера L_i в нем равна $P_i(t(x_1, \dots, x_k), \dots)$. Тогда

$$Q_i(t(t_1, \dots, t_k), \dots) = I \leftrightarrow P_i(\varphi(t(t_1, \dots, t_k)), \dots) = I.$$

Поэтому литера L_i принимает значение I . Аналогично поступаем с остальными литерами из S . Тогда в интерпретации I все литеры в S ложны, и потому S ложно в I , откуда получаем, что S ложно в I . Противоречие с истинностью S в I . Следовательно, S ложно во всех интерпретациях, т. е. множество дизъюнктов S невыполнимо.

Теорема (Левенгейм–Сколем). Если формула логики предикатов имеет какую-либо модель, то она имеет и не более чем счетную модель (например, H -интерпретацию).

Теорема Левенгейма–Сколема утверждает, что с помощью понятия выполнимости формулы логики предикатов нельзя определить мощность ее модели.

9.3. Семантические деревья. Теорема Эрбрана

Пусть множество дизъюнктов S имеет сигнатуру $(P_1, \dots, f_1, \dots, a_1, \dots)$ и эрбрановский базис $A_1, A_2, \dots$, состоящий из атомарных формул для S без предметных переменных. Задать эрбрановскую интерпретацию значит задать истинностные значения для атомов эрбрановского базиса. Пусть выражение A^c , где $c \in \{0, 1\}$, означает, что $A^1 = A$, $A^0 = \neg A$. Для последовательности формул $A_1^c, A_2^c, \dots$ (без предметных переменных) зададим эрбранову интерпретацию как последовательность $c_1, c_2, \dots$, считая, что 1 символизирует истину, а 0 – ложь. Например, двум последовательностям формул

$$A_1^1, A_2^0, A_3^0, A_4^1, A_5^0, \dots; \\ A_1^0, A_2^0, A_3^0, A_4^1, A_5^1, \dots$$

соответствуют две эрбрановы интерпретации

$$I, I, I, I, I, \dots; \\ I, I, I, I, I, \dots$$

Определение. Пусть S есть множество дизъюнктов и $\{A_1, A_2, \dots\}$ есть эрбранов базис для S . Семантическое дерево T для S есть бинарное дерево, в каждом ярусе i которого из каждого узла N исходят два ребра, одно из которых помечено буквой A_i , а другое $\neg A_i$ (рис. 9.3).

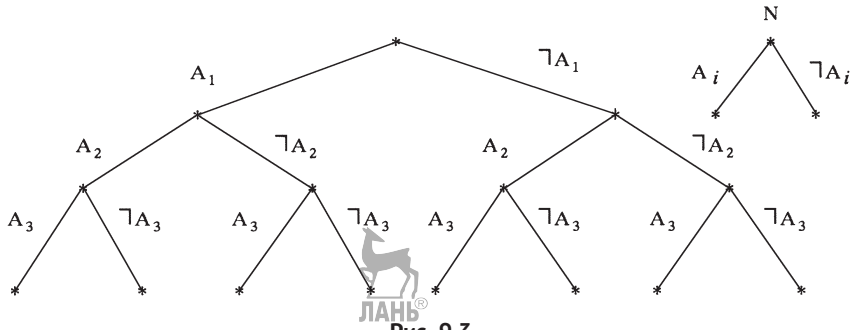


Рис. 9.3

Отметим, что ребра помечаются литерами, обозначающими истинностные значения атомов (предикатов вида $P(t_1, \dots, t_n)$, где $t_1, \dots, t_n$ есть элементы эрбранова универсума), которые есть элементы эрбранова базиса B для S .

Всякая ветвь дерева T задает некоторую N -интерпретацию. Семантическое дерево T исчерпывает все N -интерпретации для S .

Определение. Узел N в семантическом дереве T называется *опровергающим*, если путь в T от корня N содержит истинностные значения атомов из эрбранова базиса B , при которых опровергается некоторый основной пример C' некоторого дизъюнкта C из S . При этом никакой узел от корня до N указанным свойством опровержимости не обладает.

Пример. $S = \{P(x), \neg P(x) \vee Q(f(x)), \neg Q(f(a))\}$.

Эрбранов универсум $H = \{a, f(a), f(f(a)), \dots\}$.

Эрбранов базис $B = \{P(a), Q(a), P(f(a)), Q(f(a)), P(f(f(a))), Q(f(f(a))), \dots\}$.

Построим для S семантическое дерево T (рис. 9.4).

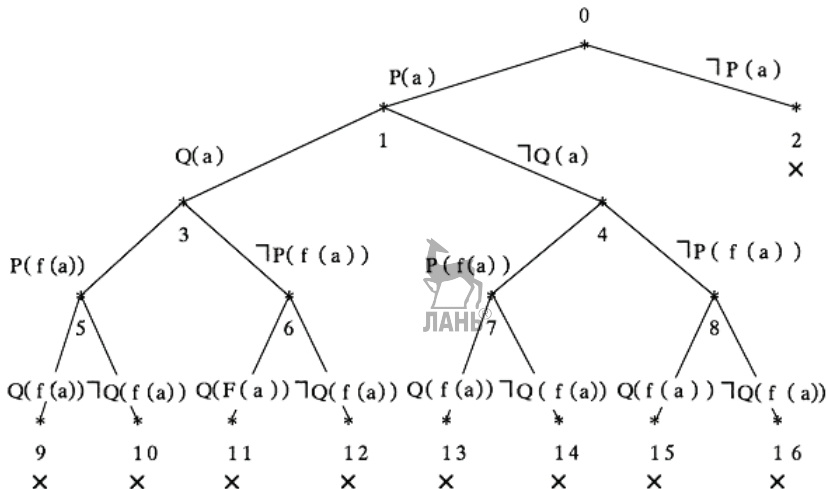


Рис. 9.4

Опроверяющие узлы на рис. 9.4 помечены крестами. Укажем соответствующие этим узлам проверяющие Н-интерпретации (табл. 9.1).

Таблица 9.1

Узел	Интерпретации	$P(a)$	$Q(a)$	$P(f(a))$	$Q(f(a))$	...
узел 2	H_1	Л	...			
узел 9	H_2	И	И	И	И	...
узел 10	H_3	И	И	И	Л	...
узел 11	H_4	И	И	Л	И	...
узел 12	H_5	И	И	Л	Л	...
узел 13	H_6	И	Л	И	И	...
узел 14	H_7	И	Л	И	Л	...
узел 15	H_8	И	Л	Л	И	...
узел 16	H_9	И	Л	Л	Л	...

В узле 2 опровержим дизъюнкт $C = P(x)$, ибо его основной пример $C' = P(a)$ ложен, т. е. формула $P(x)$ на интерпретации H_1 ложна.

В узле 9 опровержим дизъюнкт $C = \neg Q(f(a))$, ибо он в интерпретации H_2 ложен.

В узле 10 опровержим дизъюнкт $C = \neg P(x) \vee Q(f(x))$, ибо его основной пример $C' = \neg P(a) \vee Q(f(a))$ ложен в H_3 , т. е. формула $\neg P(x) \vee Q(f(x))$ на интерпретации I_3 ложна.

В узле 11 опровержим дизъюнкт $C = \neg Q(f(a))$, ибо он в интерпретации H_4 ложен.

В узле 12 опровержим дизъюнкт $C = \neg P(x) \vee Q(f(x))$, ибо его основной пример $C' = \neg P(a) \vee Q(f(a))$ ложен в H_5 .

В узле 13 опровержим дизъюнкт $C = \neg Q(f(a))$, ибо он в интерпретации H_6 ложен.

В узле 14 опровержим дизъюнкт $C = \neg P(x) \vee Q(f(x))$, ибо его основной пример $C' = \neg P(a) \vee Q(f(a))$ в H_7 ложен.

В узле 15 опровержим дизъюнкт $C = \neg Q(f(a))$, ибо он в интерпретации H_8 ложен.

В узле 16 опровержим дизъюнкт $C = \neg P(x) \vee Q(f(x))$, ибо его основной пример $C' = \neg P(a) \vee Q(f(a))$ в H_9 ложен.

Множество дизъюнктов S невыполнимо, ибо S опровержимо на каждой Н-интерпретации, т. е. всякая Н-интерпретация опровергает один из дизъюнктов во множестве S (делает ложным один из конъюнктивных множителей в конъюнкции дизъюнктов множества S).

Заметим, что в каждом проверяющем узле невыполним один из основных примеров одного из дизъюнктов в S . Конечное множество узлов дерева $\{2, 9, 10, 11, 12, 13, 14, 15, 16\}$ порождает конечное множество (которое понимается как конъюнкция) $\{P(a), \neg Q(f(a)), \neg P(a) \vee Q(f(a))\}$ основных примеров, невыполнимых на всякой Н-интерпретации (а потому и на произвольной интерпретации).

Лемма. Пусть S есть невыполнимое множество дизъюнктов. Тогда:

- 1) каждая ветвь семантического дерева T для S содержит опровергающий узел;
- 2) множество всех опровергающих узлов в T конечно.

Доказательство. Пусть V есть ветвь в T , IVN есть соответствующая ветви V Эрбрановская N -интерпретация. Множество S ложно в IVN , т. е. в S существует дизъюнкт C , существует основной пример C' для C , ложный в N -интерпретации IVN ветви V . Основной пример C' имеет конечное число атомов. Поэтому на V существует узел N , на пути к корню от которого определены все истинностные значения атомов из C' . Так как C' ложно в IVN , то C' опровержимо в N . В качестве опровергающего узла N возьмем на пути V ближайший к корню узел с указанным свойством. Итак, каждая ветвь в дереве T имеет опровергающий узел.

Покажем, что число опровергающих узлов в T конечно.

Обрежем дерево T , отбросив в T каждое поддереву с корнем в опровергающем узле (оставив сам опровергающий узел). Получим дерево T' . Покажем, что дерево T' конечно (имеет конечное число узлов). Допустим, что T' бесконечно. Среди узлов яруса 1 в T' выберем поддерево T_x с корнем x , которое содержит бесконечно много узлов (в T'). Узел x опровергающим не является. В T_x выберем то поддерево T_y с корнем y в ярусе 2, которое содержит бесконечно много узлов из T' . Узел y опровергающим не является. И так далее. В результате получим бесконечную ветвь, проходящую через узлы $x, y, \dots$, без опровергающего узла. Противоречие с наличием опровергающего узла на каждой ветви в T . Следовательно, множество опровергающих узлов в T конечно.

Теорема (Эрбран). Конечное множество дизъюнктов S невыполнимо (ни в какой интерпретации) тогда и только тогда, когда существует конечное невыполнимое (ни в какой интерпретации) подмножество S' множества всех основных примеров дизъюнктов из S .

Доказательство. Пусть множество дизъюнктов S невыполнимо. Возьмем семантическое дерево T для S и отметим в T все его опровергающие узлы (их конечное число). Для каждого опровергающего узла возьмем один основной пример того дизъюнкта из S , который опровергается в этом узле. Образуется конечное невыполнимое на всякой N -интерпретации (а потому и на произвольной интерпретации) множество S_0 основных примеров дизъюнктов из S (рассматриваемое как их конъюнкция), ибо для всякой N -интерпретации ложен один из основных примеров в S_0 того дизъюнкта из S , который опровергается в опровергающем узле на ветви семантического дерева T , соответствующей этой N -интерпретации.

Пусть теперь некоторое множество S_0 основных примеров для S невыполнимо (при любой интерпретации). Покажем, что множество дизъюнктов S невыполнимо. Допустим противное: множество S выполнимо. Тогда существует интерпретация, а следовательно, и некоторая N -интерпретация IN , выполняющая множество S . Тогда любой дизъюнкт $C = (\forall x_1) \dots (\forall x_k) D(x_1, \dots, x_k)$ выпол-

ним на ИН. По смыслу кванторов общности на ИН выполнима и любая формула $D(t_1, \dots, t_k)$ при всяких $t_1, \dots, t_k$ из эрбрановского универсума H для S . Но тогда выполнимо и множество основных примеров S_0 , ибо всякий основной пример C' из S_0 получается из некоторого дизъюнкта C из S подходящей заменой предметных переменных $x_1, \dots, x_k$ в D на некоторые элементы $t_1, \dots, t_k$ эрбрановского универсума H . Противоречие с невыполнимостью S_0 . Следовательно, множество S невыполнимо.

Теорема доказана.

На основе теоремы Эрбрана строится формально-логическое исчисление, а именно метод резолюций, для установления невыполнимости (а потому и общезначимости) формул логики предикатов. А именно формула A представляется в стандартной форме Сколема B . Бескванторная часть формулы B представляется в конъюнктивной нормальной форме и рассматривается как множество дизъюнктов S . Строим эрбрановский универсум и эрбрановский базис для S .

Строим семантическое дерево T для S , поярусно проверяя каждый построенный узел на опровержимость. Если построенный узел оказывается опровержимым, то дальнейшее построение дерева от этого узла прекращается. Если на некотором шаге дальнейшее построение дерева T невозможно (все листья конструируемого дерева оказались опровергающими узлами), то множество дизъюнктов S (а потому и исходная формула A) невыполнимо. Если же построение дерева T продолжается бесконечно, то множество дизъюнктов S (а потому и исходная формула A) невыполнимым не является. Заметим, что не существует алгоритма, который по любой формуле логики предикатов A устанавливал бы, закончится процедура построения семантического дерева T для S или не закончится.

9.4. Унификация

Определение. Подстановка (t_i вместо x_i) есть множество $\theta = \{t_1/x_1, \dots, t_k/x_k\}$, где $x_1, \dots, x_k$ есть попарно различные предметные переменные, а все t_i есть термы, отличные от x_i , $i = 1, 2, \dots, k$.

Пусть ε есть пустая подстановка.

Определение. Пусть C есть дизъюнкт, а $\theta = \{t_1/x_1, \dots, t_k/x_k\}$ есть подстановка. Обозначим через $C\theta$ (точнее через $C \cdot \theta$) дизъюнкт, полученный одновременной заменой переменных $x_1, \dots, x_k$ в C на термы $t_1, \dots, t_k$ соответственно. Дизъюнкт $C\theta$ называется *примером* дизъюнкта C .

Аналогично определяется пример терма.

Замечание. Основным примером дизъюнкта не имеет предметных переменных. Пример дизъюнкта может иметь предметные переменные.

Определение. Подстановка θ есть *унификатор* множества литер $W = \{L_1, \dots, L_m\}$, если $L_1\theta = L_2\theta = \dots = L_m\theta$. (Обозначение: $Y(W) = \theta$.) Литеры $L_1, \dots, L_m$ называются унифицируемыми.

Пример. Множество литер $\{P(x), P(f(y))\}$ унифицируемо подстановкой $\theta = \{f(y)|x\}$. Множество литер $\{P(x), P(y), \neg P(z)\}$ не унифицируемо, ибо никакая подстановка не сделает равной формулу и ее отрицание.

Определение. Пусть $\theta = \{t_1|x_1, \dots, t_k|x_k\}$, $\lambda = \{u_1|y_1, \dots, u_m|y_m\}$ есть две подстановки. Композиция $\theta \circ \lambda$ подстановок θ и λ есть подстановка, полученная из множества

$$\{t_1\lambda|x_1, \dots, t_k\lambda|x_k, u_1|y_1, \dots, u_m|y_m\}$$

удалением всех элементов вида $t_i\lambda \mid x_i$, для которых $t_i\lambda = x_i$, и удалением всех элементов $u_i|y_i$, в которых $y_i \in \{x_1, \dots, x_k\}$.

Пример

$$\begin{aligned}\theta &= \{t_1|x_1, t_2|x_2\} = \{f(y)|x, z|y\}; \\ \lambda &= \{u_1|y_1, u_2|y_2, u_3|y_3\} = \{a|x, b|y, y|z\}.\end{aligned}$$

Множество

$$\begin{aligned}\{t_1\lambda|x_1, t_2\lambda|x_2, u_1|y_1, u_2|y_2, u_3|y_3\} = \\ \{f(y)\lambda|x, z\lambda|y, a|x, b|y, y|z\} = \{f(b)|x, y|y, a|x, b|y, y|z\}.\end{aligned}$$

Из этого множества удаляем элемент $y|y$ и элементы $a|x$ и $b|y$, ибо $y_1 = x \in \{x, y\} = \{x_1, x_2\}$ и $y_2 = y \in \{x, y\} = \{x_1, x_2\}$. Композиция $\theta\lambda = \theta \circ \lambda = \{f(b)|x, y|z\}$.

Замечание

1. $(\theta \circ \lambda) \circ \mu = \theta \circ (\lambda \circ \mu)$.
2. $\varepsilon \circ \theta = \theta \circ \varepsilon = \theta$.
3. $C(\theta \circ \lambda) = (C\theta)\lambda$.

Пример. $C = P(x, f(y), g(x, z))$. Подстановки θ и λ взяты из предыдущего примера.

$$\begin{aligned}C(\theta \circ \lambda) &= (P(x, f(y), g(x, z))) \circ (\theta \circ \lambda) = P(f(b), f(y), g(f(b), y)); \\ (C\theta)\lambda &= ((P(x, f(y), g(x, z))\theta)\lambda = P(f(y), f(z), g(f(y), z))\lambda = P(f(b), f(y), g(f(b), y)).\end{aligned}$$

Определение. Пусть W есть множество литер. Унификатор σ есть наиболее общий унификатор для W (обозначение $\sigma = \text{НОУ}(W)$), если для любого другого унификатора θ для W существует подстановка λ , для которой $\theta = \sigma \circ \lambda$.

Замечание. Из НОУ для W можно получить любой другой унификатор для W .

Пример. $W = \{P(x), P(f(y))\}$. Подстановки $\sigma = \{f(a)|x, a|y\}$; $\theta = \{f(y)|x\}$, являющиеся унификаторами для W и θ , есть НОУ.

Определение. Множество рассогласований некоторого множества литер W получается выписыванием из W всех термов, начинающихся с первой слева позиции в литере, на которой стоят буквы, не все одинаковые.

Пример. $W = \{P(x, f(y, z)); P(x, a)\}$. Множество рассогласований $D = \{f(y, z), a\}$.

Алгоритм унификации

Пусть $W = \{L_1, \dots, L_r\}$ есть множество литер с совпадающими предикатными буквами, которые либо все положительны (без отрицаний), либо все отрицательны (с отрицаниями). В противном случае множество литер W не унифицируемо. Пусть $|W|$ означает число литер в W .

Выполним следующую последовательность шагов.

1. $W_0 = W, \sigma_0 = \varepsilon$.

2. Если W_k содержит единственную литеру, то останов: σ_k есть НОУ для W . В противном случае найдем множество D_k рассогласований для W_k . Если множество D_k содержит переменную v_k и терм t_k , причем v_k не содержится в t_k , то перейдем к следующему шагу. В противном случае останов: множество W не унифицируемо.

3. $\sigma_{k+1} = \sigma_k \{t_k | v_k\}$; $W_{k+1} = W_k \{t_k | v_k\}$ (при этом $W_{k+1} = W \sigma_{k+1}$). Переходим к пункту 2.

Пример. $W = \{P(a, x, f(g(y))), P(z, f(z), f(u))\}$.

0. $\sigma_0 = \varepsilon$; $W_0 = W$; $|W_0| > 1$; $D_0 = \{a, z\}$; $v_0 = z$, $t_0 = a$, $v_0 \notin t_0$; $\omega_0 = \{a | z\}$.

1. $\sigma_1 = \sigma_0 \circ \{t_0 | v_0\} = \varepsilon \circ \{a | z\} = \sigma_0 \circ \omega_0$;

$W_1 = W_0 \omega_0 = W \omega_0 = \{P(a, x, f(g(y))), P(a, f(a), f(u))\}$; $|W_1| > 1$; $D_1 = \{x, f(a)\}$;

$v_1 = x$, $t_1 = f(a)$, $v_1 \notin t_1$; $\omega_1 = \{f(a) | x\}$.

2. $\sigma_2 = \sigma_1 \circ \omega_1 = \{a | z\} \circ \{f(a) | x\} = \{a | z, f(a) | x\}$;

$W_2 = W_1 \omega_1 = (W_0 \sigma_0 \omega_0) \omega_1 = W_0 \sigma_1 \omega_1 = W \sigma_2 = \{P(a, f(a), f(g(y))), P(a, f(a), f(u))\}$; $|W_2| > 1$;

$D_2 = \{g(y), u\}$; $v_2 = u$, $t_2 = g(y)$, $v_2 \notin t_2$; $\omega_2 = \{g(y) | u\}$.

3. $\sigma_3 = \sigma_2 \circ \omega_2 = \{a | z, f(a) | x\} \circ \{g(y) | u\} = \{a | z, f(a) | x, g(y) | u\}$;

$W_3 = W_2 \omega_2 = W_0 \sigma_2 \omega_2 = W \sigma_3 = \{P(a, f(a), f(g(y))), P(a, f(a), f(g(y)))\} = \{P(a, f(a), f(g(y)))\}$;

$|W_3| = 1$. Останов. Унификатор $\theta = \sigma_3$, при этом $\theta = \text{НОУ}(W)$.

Пример. $W = \{Q(f(a, g(x))), Q(y, y)\}$.

0. $\sigma_0 = \varepsilon$; $W_0 = W$; $|W_0| > 1$; $D_0 = \{f(a), y\}$; $v_0 = y$, $t_0 = f(a)$, $v_0 \notin t_0$; $\omega_0 = \{f(a) | y\}$.

1. $\sigma_1 = \sigma_0 \circ \omega_0 = \omega_0 = \{f(a) | y\}$; $W_1 = W_0 \omega_0 = \{Q(f(a), g(x)), Q(f(a), f(a))\}$;

$|W_1| > 1$; $D_1 = \{g(x), f(a)\}$. В D_1 нет переменной v_1 , не входящей в какой-либо терм из D_1 . Множество W не унифицируемо.

Замечание. Алгоритм унификации всегда заканчивает свою работу, указывая при этом унификатор или сообщая об его отсутствии. В языке программирования Пролог унификация есть одна из его основных операций.

Теорема (об унификации). Пусть W есть конечное унифицируемое множество литер. Тогда алгоритм унификации всегда заканчивает свою работу через конечное число шагов и выдает для W наиболее общий унификатор.

Доказательство. Пусть θ есть произвольный унификатор для W , т. е. $\theta = \text{У}(W)$. Алгоритм унификации останавливается всегда. Пусть, работая над W , алгоритм останавливается на шаге r . Покажем, что алгоритм указал на унифицируемость W и выдал НОУ для W .

Утверждение. Для всякого шага k , $0 \leq k \leq r$, работы алгоритма унификации существует подстановка λ_k , для которой $\theta = \sigma_k \circ \lambda_k$.

Доказательство. Индукция по k .

Базис. $k = 0$. Так как $\theta = \varepsilon \circ \theta$ и $\sigma_0 = \varepsilon$, то при $\lambda_0 = \theta$ получим $\theta = \sigma_0 \circ \lambda_0$.

Предположение индукции. Пусть для шага k , $0 \leq k < r$, существует подстановка λ_k , для которой $\theta = \sigma_k \circ \lambda_k$.

Шаг индукции. Покажем, что на шаге $k + 1$ существует подстановка λ_{k+1} , для которой $\theta = \sigma_{k+1} \circ \lambda_{k+1}$. На шаге k работы алгоритма унификации возможны следующие два случая.

1. $|W| = 1$. Тогда $k = r$, алгоритм унификации закончил свою работу, выдал унификатор σ_r , причем для произвольного унификатора θ по предположению индукции существует подстановка λ_r , для которой $\theta = \sigma_r \circ \lambda_r$. Следовательно, $\sigma_r = \text{НОУ}(W)$.

2. $|W| > 1$. Тогда $k < r$, и возможен следующий шаг. На шаге k алгоритм выдал множество рассогласований D_k для множества $W_k = W \circ \sigma_k$. Так как подстановка $\theta = \sigma_k \circ \lambda_k$ есть унификатор для W , то подстановка λ_k должна унифицировать D_k (как часть W). Так как D_k есть множество рассогласований, то множество D_k имеет в качестве элемента не переменную v_k (иначе множество D_k имеет одни константы и не унифицируется).

Пусть t_k есть любой другой терм в D_k , отличный от v_k . Так как λ_k унифицирует D_k , то $v_k \lambda_k = t_k \lambda_k$. Переменной v_k нет в t_k , ибо иначе ввиду $v_k \neq t_k$ получим $v_k \lambda_k \neq t_k \lambda_k$, что противоречит полученному выше равенству $v_k \lambda_k = t_k \lambda_k$. Так как v_k нет в t_k , то алгоритм унификации выдает подстановку $\omega_k = \{t_k \mid v_k\}$ и строит подстановку $\sigma_{k+1} = \sigma_k \omega_k = \sigma_k \circ \{t_k \mid v_k\}$.

Положим $\lambda_{k+1} = \lambda_k - \{t_k \lambda_k \mid v_k\}$. Так как v_k нет в t_k , то $t_k \lambda_{k+1} = t_k (\lambda_k - \{t_k \lambda_k \mid v_k\}) = t_k \lambda_k$. Тогда

$$\omega_k \circ \lambda_{k+1} = \{t_k \mid v_k\} \circ \lambda_{k+1} = \{t_k \lambda_{k+1} \mid v_k\} \cup \lambda_{k+1} = \{t_k \lambda_k \mid v_k\} \cup \lambda_{k+1} = \\ \{t_k \lambda_k \mid v_k\} \cup \{\lambda_k \mid \{t_k \lambda_k \mid v_k\}\} = \lambda_k, \text{ т. е. } \lambda_k = \omega_k \circ \lambda_{k+1}.$$

Следовательно, $\theta = \sigma_k \circ \lambda_k = \sigma_k \circ (\omega_k \circ \lambda_{k+1}) = (\sigma_k \circ \omega_k) \circ \lambda_{k+1} = \sigma_{k+1} \circ \lambda_{k+1}$.

Шаг индукции установлен. Утверждение доказано.

Продолжим доказательство теоремы. При $k = r$ получим $\theta = \sigma_r \circ \lambda_r$, где $\sigma_r = \text{НОУ}(W)$. Итак, для произвольного унификатора θ и унификатора σ_r существует подстановка λ_r , для которой $\theta = \sigma_r \circ \lambda_r$, т. е. $\sigma_r = \text{НОУ}(W)$.

Теорема доказана.

9.5. Метод резолюций в логике предикатов

Опишем формально-логическое исчисление – метод резолюций, который был принят за основу при разработке языка программирования Пролог.

Определение. Если две или более литер (одной и той же положительности) в дизъюнкте C , рассматриваемом как множество литер, имеют наиболее общий унификатор σ , то множество литер $C\sigma$ называется *склежкой* для C . Если $C\sigma$ есть одноэлементный дизъюнкт, то склейка называется *единичной склейкой*.

Пример. Дизъюнкт $C = P(x) \vee P(f(y)) \vee \neg Q(x)$.

$$\sigma = \text{НОУ}(P(x) \vee P(f(y)) \vee \neg Q(x)) = \{f(y) \mid x\}.$$

Следовательно, дизъюнкт

$$C\sigma = P(f(y)) \vee P(f(y)) \vee \neg Q(x) = P(f(y)) \vee \neg Q(x)$$

есть склейка для C .

Определение. Пусть C_1 и C_2 есть два дизъюнкта (называемых посылками), которые не имеют никаких общих переменных, а дизъюнкт C' есть либо C_1 ,

либо его склейка. Аналогично для C'' . Пусть L_1 есть литера из C' , а $\neg L_2$ есть литера из C'' . Если L_1 и L_2 имеют наиболее общий унификатор σ , то дизъюнкт $C = (C'\sigma - L_1\sigma) \cup (C''\sigma - \neg L_2\sigma)$ называется *резольвентой* дизъюнктов C_1 и C_2 . Дизъюнкт C называется также заключением (выведенным из посылок C_1 и C_2). Литеры L_1 и L_2 называются *отрезаемыми литерами*.

Пример. Дизъюнкты $C_1 = P(x) \vee Q(x)$ и $C_2 = \neg P(a) \vee R(y)$; $\sigma = \{a/x\}$ НОУ($P(x), P(a)$); литеры $L_1 = P(x)$, $L_2 = P(a)$.

Резольвента C дизъюнктов C_1 и C_2 есть дизъюнкт

$$C = (C_1\sigma - L_1\sigma) \cup (C_2\sigma - \neg L_2\sigma) = \\ (\{P(a), Q(a)\} - \{P(a)\}) \cup (\{\neg P(a), R(y)\} - \{\neg P(a)\}) = \\ \{Q(a)\}, \{R(y)\} = Q(a) \vee R(y).$$

Определение. Дизъюнкт C получен из дизъюнктов C_1 и C_2 по *правилу резолюции*, если C есть резольвента дизъюнктов C_1 и C_2 .

Пусть запись $C = \text{ПР}(C_1, C_2)$ означает, что дизъюнкт C получен из дизъюнктов C_1 и C_2 по правилу резолюции.

Замечание. В основе правила резолюции лежат классические правило подстановки и правило заключения:

$$\frac{P(x)}{P(t)}, \frac{A, A \rightarrow B}{B} \text{ эквивалентное правилу } \frac{A, \neg A \rightarrow B}{B}.$$

Последнее можно обобщить до правила вывода

$$\frac{A \vee B_1 \vee \dots \vee B_m, \neg A \vee C_1 \vee \dots \vee C_n}{B_1 \vee \dots \vee B_m \vee C_1 \vee \dots \vee C_n}.$$

Формулы A и $\neg A$ могут стоять в любом месте дизъюнкции. Включающее в себя подстановку правило резолюции

$$\frac{L_1 \vee L_{11} \vee \dots \vee L_{1m}, \neg L_2 \vee L_{21} \vee \dots \vee L_{2n}}{(L_1 \vee L_{11} \vee \dots \vee L_{1m}, \neg L_2 \vee L_{21} \vee \dots \vee L_{2n}) \circ \theta}$$

получается из последнего правила вывода, если в нем вместо

$$A \vee B_1 \vee \dots \vee B_m, \neg A \vee C_1 \vee \dots \vee C_n$$

взять дизъюнкты

$$L_1 \vee L_{11} \vee \dots \vee L_{1m}, \neg L_2 \vee L_{21} \vee \dots \vee L_{2n},$$

составленные из элементарных формул (предикатных атомов), имеющих в своем составе функциональные термы. С помощью алгоритма унификации разыскивается та подстановка σ , которая атомы L_1 и L_2 делает идентичными: $L_1 \circ \sigma = L_2 \circ \sigma$. Результат

$$(L_{11} \vee \dots \vee L_{1m} \vee L_{21} \vee \dots \vee L_{2n}) \circ \sigma$$

подстановки σ вместо переменных в дизъюнкции

$$L_{11} \vee \dots \vee L_{1m} \vee L_{21} \vee \dots \vee L_{2n}$$

ставится в знаменатель правила резолюции.

Определение. Резолютивный вывод из множества дизъюнктов S есть последовательность дизъюнктов, каждый из которых есть либо дизъюнкт из S , либо получен из предыдущих дизъюнктов последовательности по правилу резолюции.

Определение. Дизъюнкт C *выводится* из множества дизъюнктов S (обозначение $S \vdash C$), если существует конечный резолютивный вывод из множества S , последним дизъюнктом которого является C .

Теорема. Если из множества дизъюнктов S выводится дизъюнкт C , то формула $S \rightarrow C$ общезначима.

Доказательство. Индукция по числу k применений правила резолюций.

Базис. $k = 1$. Покажем, что резольвента C двух дизъюнктов C_1 и C_2 есть логическое следствие из C_1 и C_2 . Так как $C = \text{ПР}(C_1, C_2)$, то существуют дизъюнкты C'_1, C'_2 , литеры L_1 и L_2 и подстановка σ , для которых $C_1 = C'_1 \vee L_1$, $C_2 = C'_2 \vee \neg L_2$, $L_1\sigma = L_2\sigma$, $C = (C'_1\sigma - L_1\sigma) \cup (C'_2\sigma - \neg L_2\sigma)$.

Пусть x есть список всех предметных переменных для C_1 и C_2 , а y есть список всех предметных переменных для $C'_1\sigma \vee L_1\sigma$, $C'_2\sigma \vee \neg L_2\sigma$. Подстановку σ можно выбрать так, что списки переменных x и y не пересекаются. Дизъюнкты C_1, C_2 , записанные без кванторов в виде множества литер, представляют собой другую запись формулы $(\forall x)(C_1 \ \& \ C_2)$, т. е. формулы $(\forall x)((C'_1 \vee L_1) \ \& \ (C'_2 \vee \neg L_2))$. Тогда формула $(\forall x)((C'_1 \vee L_1) \ \& \ (C'_2 \vee \neg L_2)) \rightarrow (C'_1\sigma \vee L_1\sigma) \ \& \ (C'_2\sigma \vee \neg L_2\sigma)$ общезначима, ибо ее заключение есть подстановка некоторых термов вместо некоторых переменных в посылке $(C'_1 \vee L_1) \ \& \ (C'_2 \vee \neg L_2)$. Формула $(C'_1\sigma \vee L_1\sigma) \ \& \ (C'_2\sigma \vee \neg L_2\sigma) \rightarrow (C'_1\sigma \vee C'_2\sigma)$ общезначима из алгебраических соображений. Тогда общезначима формула

$$(\forall x)((C'_1 \vee L_1) \ \& \ (C'_2 \vee \neg L_2)) \rightarrow (C'_1\sigma \vee C'_2\sigma).$$

Следовательно, общезначима формула

$$(\forall y)(\forall x)((C'_1 \vee L_1) \ \& \ (C'_2 \vee \neg L_2) \rightarrow (C'_1\sigma \vee C'_2\sigma)).$$

Поэтому общезначима формула

$$(C'_1 \vee L_1) \ \& \ (C'_2 \vee \neg L_2) \rightarrow (C'_1\sigma \vee C'_2\sigma),$$

которая совпадает с формулой $C_1 \ \& \ C_2 \rightarrow (C'_1 \vee C'_2)\sigma$, являющейся тоже общезначимой. Следовательно, резольвента $(C'_1 \vee C'_2)\sigma$ есть логическое следствие дизъюнктов C_1 и C_2 .

Предположение индукции. Допустим, что теорема справедлива для всех выводов, в которых число применений правила резолюции меньше k .

Шаг индукции устанавливается точно так же, как и в случае аналогичной теоремы для логики высказываний.

Лемма (о подъеме). Пусть C'_1, C'_2 есть примеры дизъюнктов C_1 и C_2 соответственно. Пусть C' есть резольвента для C'_1, C'_2 . Тогда существует резольвента C для C_1 и C_2 , для которой C' есть основной пример.

Доказательство. Можно считать, что C_1 и C_2 (так же как и C'_1, C'_2) не имеют общих переменных (это достигается переименованием переменных). По условию для C'_1 и C'_2 резольвента $C' = (C'_1\sigma - L'_1\sigma) \cup (C'_2\sigma - L'_2\sigma)$, где $\sigma = \text{НОУ}(L'_1, \neg L'_2)$, а L'_1 и L'_2 есть отрезаемые литеры, причем $L'_1 \in C'_1, L'_2 \in C'_2$. Так как C'_1 и C'_2 не имеют общих переменных, то подстановку σ можно представить в виде объединения двух подстановок $\sigma = \sigma_1 \cup \sigma_2$, где σ_1 содержит только переменные из C'_1 , а σ_2 из C'_2 . Так как C'_1 и C'_2 есть примеры для C_1 и C_2 соответственно, то для некоторых подстановок θ_1 и θ_2 имеем $C'_1 = C_1\theta_1, C'_2 = C_2\theta_2$. Тогда резольвента $C' = (C_1\theta_1\sigma_1 - L'_1\sigma_1) \cup (C_2\theta_2\sigma_2 - L'_2\sigma_2)$. Выделим в C_i все те литеры, которые подстановка θ_i склеивает в одну литеру. Тогда

$$L_i^1\theta_i = \dots = L_i^i\theta_i = L'_i\sigma_i, i = 1, 2.$$

Так как множество $\{L_i^1, \dots, L_i^i\}, i = 1, 2$, унифицируемо, то существует наиболее общий унификатор λ_i для этого множества литер, тогда как $\theta_i\sigma_i$ есть какой-то унификатор. Следовательно, существует подстановка ρ_i , что $\theta_i\sigma_i = \lambda_i\rho_i$. Поэтому резольвента

$$C' = (C_1\lambda_1\rho_1 - \{L_1^1, \dots, L_1^i\}\lambda_1\rho_1) \cup (C_2\lambda_2\rho_2 - \{L_2^1, \dots, L_2^i\}\lambda_2\rho_2).$$

Пусть $L_i = L_i^1\lambda_i, i = 1, 2$ (в силу того, что λ_i есть наиболее общий унификатор для множества $\{L_i^1, \dots, L_i^i\}$, все литеры $L_i^1\lambda_i, \dots, L_i^i\lambda_i$ совпадают). Тогда L_i есть литеры в склейке $C_i\lambda_i$ для C_i . Поскольку переменные в C_1 и C_2 не пересекаются, объединим соответствующие им подстановки следующим образом: $\lambda = \lambda_1 \cup \lambda_2$ и $\rho = \rho_1 \cup \rho_2$. Тогда $C' = (C_1\lambda\rho - L_1\rho) \cup (C_2\lambda\rho - L_2\rho)$, где ρ есть некоторая подстановка, унифицирующая L_1 и L_2 .

Пусть $\tau = \text{НОУ}(L_1, L_2)$. По определению НОУ найдется подстановка v , для которой $\rho = \tau v$. Тогда

$$C' = ((C_1\lambda)\tau v - L_1\tau v) \cup ((C_2\lambda)\tau v - L_2\tau v) = (C_1(\lambda \circ \tau \circ v) - L_1^1(\lambda \circ \tau \circ v)) \cup (C_2(\lambda \circ \tau \circ v) - L_2^1(\lambda \circ \tau \circ v)).$$

С другой стороны, построим резольвенту C дизъюнктов C_1 и C_2 следующим образом:

$$C = ((C_1\lambda)\tau - (L_1^1\lambda)\tau) \cup ((C_2\lambda)\tau - (L_2^1\lambda)\tau) = (C_1(\lambda \circ \tau) - L_1^1(\lambda \circ \tau)) \cup (C_2(\lambda \circ \tau) - L_2^1(\lambda \circ \tau)).$$

Очевидно, что C' является примером резольвенты C , поскольку $(\lambda \circ \tau)$ есть более общий унификатор, чем $(\lambda \circ \tau \circ v)$. Лемма доказана.

Теорема (о полноте метода резолюций). Непустое множество дизъюнктов S невыполнимо тогда и только тогда, когда из S выводится пустой дизъюнкт.

Доказательство. Пусть множество дизъюнктов S невыполнимо. Пусть T есть семантическое дерево для S . Так как S невыполнимо, то множество опровергающих узлов в T конечно, и каждый путь в T имеет опровергающий узел. Пусть T' есть конечное поддереве дерева T с (конечными) путями от корня до опровергающих узлов. Докажем, что пустой дизъюнкт выводится из S . Доказательство проведем методом индукции по числу k узлов в T' .

Базис. $k = 1$, т. е. дерево T' имеет единственный опровергающий корневой узел. Так как множество S не пусто и не выполнимо, то S содержит пустой дизъюнкт, выводимый из S .

Предположение индукции. Допустим, что доказываемое утверждение справедливо для любого S и потому для любого семантического дерева T , для которого T' имеет число узлов меньше k .

Шаг индукции. Покажем, что доказываемое утверждение справедливо для любого S и потому для любого семантического дерева T , для которого T' имеет число узлов k . Пусть N есть узел, предшествующий двум конечным опровергающим узлам N_1 и N_2 в T' . Пусть ребрам, исходящим из N , приписаны литеры L_k и $\neg L_k$. Пусть пути от корня до N_1 и N_2 проходят через литеры

$L_1, \dots, L_{k-1}, L_k$ до узла N_1 ;
 $L_1, \dots, L_{k-1}, \neg L_k$ до узла N_2 .

Пусть C'_1 есть основной пример дизъюнкта C_1 из множества дизъюнктов S , который опровергается (ложен) в N_1 , а C'_2 есть основной пример дизъюнкта C_2 из S , который опровергается в N_2 . В узле N дизъюнкты C_1 и C_2 не опровергаются. Поэтому в C'_1 входит $\neg L_k$, а в C'_2 входит L_k . Для C'_1 и C'_2 , отрезая литеру L_k , получим резольвенту $C' = (C'_1 - \neg L_k) \cup (C'_2 - L_k)$. Так как

$C'_1 = L_1^1 \vee \dots \vee L_{1r_1}^1 \vee \neg L_k$ ложно в N_1 ,
 $C'_2 = L_1^2 \vee \dots \vee L_{2r_2}^2 \vee L_k$ ложно в N_2 ,

то резольвента $C' = L_1^1 \vee \dots \vee L_{1r_1}^1 \vee L_1^2 \vee \dots \vee L_{2r_2}^2$, не содержащая L_k и $\neg L_k$, ложна в N . По лемме о подъеме для C_1 и C_2 существует резольвента C , для которой C' является основным примером. Пусть $S_1 = S \cup \{C\}$. Множество S_1 не выполнимо, ибо не выполнимо его подмножество S .

Семантическое дерево для S_1 совпадает с семантическим деревом для S . Дерево T_1 , полученное из T удалением поддеревьев с корнями в опровергающих узлах для S_1 , входит в дерево, полученное удалением из T' узлов N_1 и N_2 с принадлежащими им ребрами. Так как число узлов в T_1 меньше k , то по предположению индукции из S_1 выводится пустой дизъюнкт. Следовательно, пустой дизъюнкт выводится и из S , ибо S_1 получается из S однократным применением правила резолюций.

Достаточность очевидна, ибо если из S выводится пустой дизъюнкт, то формула $S \rightarrow \square$ общезначима, а потому множество S не выполнимо. Теорема доказана.

Замечание. Пролог имеет дело с хорновскими дизъюнктами. Хорновский дизъюнкт есть дизъюнкт, содержащий не более одной положительной литеры; оставшиеся литеры дизъюнкта отрицательны. Хорновский дизъюнкт имеет вид $A \vee \neg B \vee \neg C \vee \dots \vee \neg D$ (что равносильно формуле $B \& C \& \dots \& D \rightarrow A$, записываемой в Прологе как $A : - B, C, \dots, D$, где $A, B, C, \dots, D$ есть некоторые атомы). Отрицательных дизъюнктивных слагаемых в хорновских дизъюнктах может и не быть. Пролог-программу составляет множество $\{D_1, D_2, \dots, D_n\}$ (рас-

смаатриваемое как конъюнкция $D1 \& D2 \& \dots \& Dn$) хорновских дизъюнктов $D1, D2, \dots, Dn$, замкнутых кванторами общности. В этом смысле программирование в Прологе есть программирование в хорновских дизъюнктах. Пролог-программе можно ставить запросы. Запрос есть конъюнкция $Q(X)$ положительных атомов $Q1, Q2, \dots, Qk$, где X есть список всех их переменных. Нужно выяснить, следует ли логически запрос $Q(X)$ из программы $D1 \& D2 \& \dots \& Dn$ и при каких значениях переменных X это возможно, т. е. является ли формула $D1 \& D2 \& \dots \& Dn \rightarrow (EX)Q(X)$ общезначимой, или ее отрицание $D1 \& D2 \& \dots \& Dn \& (AX) \neg Q(X)$ не выполнимо.

Другими словами, выводится ли из множества хорновых дизъюнктов $D1, D2, \dots, Dn$, $\neg Q(X)$ пустой дизъюнкт и при каких значениях переменных X это возможно. Вычисление этого обстоятельства проводится с помощью правила резолюции, причем в Прологе используется один его частный вид, называемый линейной резолюцией. А именно в запросе $\neg Q$, являющемся дизъюнкцией $\neg Q1 \vee \neg Q2 \vee \dots \vee \neg Qk$, правило резолюции применяется к первому дизъюнктивному слагаемому $\neg Q1$ и к первому (хорновому) дизъюнкту F , равному $A \vee \neg B \vee \neg C \vee \dots \vee \neg D$ в Пролог-программе, такому, что к $\neg Q1$ и F применение правила резолюции возможно. Это правило имеет вид для того $HOY q$, для которого атомы Aq и $Q1Wq$ одинаковы. Формируется новый запрос $(\neg B \vee \neg C \vee \dots \vee \neg D \vee \neg Q2 \vee \dots \vee \neg Qk)q$, в котором правило линейной резолюции, в свою очередь, будет применяться к первому дизъюнктивному слагаемому $(\neg B)q$. Вывод в Прологе есть последовательное получение подобных запросов с помощью линейной резолюции. Применение указанного выше правила линейной резолюции эквивалентно применению двух обычных правил логики предикатов – правила подстановки и правила заключения: для линейной резолюции справедлива теорема о полноте. Для хорновых дизъюнктов справедливо следующее утверждение.

Утверждение. Пролог является универсальным языком программирования: любой алгоритм может быть записан в формализме метода резолюций с хорновыми дизъюнктами.

В свете этого утверждения любая задача, записываемая на языке дизъюнктов, может быть записана и на языке хорновых дизъюнктов.

9.6. Основы Пролога

Универсальный язык программирования Пролог (акроним от PROgramming in LOGic) был разработан в 1970-х г. При разработке программ будем ориентироваться на версию языка SWI-Prolog.

Алфавит

1. $A B C \dots X Y Z a b c \dots x y z$ есть большие и малые буквы латинского алфавита. В компьютерах, имеющих клавиатуру с кириллицей, допускаются большие и малые буквы русского алфавита.

2. $0 1 2 3 4 5 6 7 8 9$ есть цифры.

3. $+ - * / < > = : \dots$ есть специальные знаки (или спецзнаки). В качестве спецзнаков могут выступать любые печатаемые символы клавиатуры компьютера, не являющиеся буквой или цифрой.

4. $() []$, есть различного вида скобки и запятая. Эти символы можно было бы отнести к спецзнакам; их отметили особо ввиду частого использования в стандартных конструкциях языка.

Приступим к описанию типов данных в Прологе.

Атомы

1. Конечная последовательность, состоящая из букв, цифр и символов подчеркивания `_`, начинающаяся с малой буквы, есть (буквенный) атом.

Пример. `xу, x1, x34, abc, x25AB, x, procedure, beata, alexey_morozov, май_1_1989, a_25, x__y_7, b_, anna, nil.`

2. Конечная последовательность спецзнаков есть (специальный) атом.

Пример. `<-->, <--->, =>, ==>, ..., *, +, /, :, :: = .`

При использовании атомов этого вида следует сохранять осторожность, ибо некоторые атомы, например такие, как `:-, =, =..=, =\=` и другие, имеют специальный смысл.

3. Любое слово в алфавите формальных символов, заключенное в апострофы, есть атом.

Пример. `'1 мая 1989 года', 'Алексей Морозов', 'Адрес неизвестен', 'зов предков', 'звезда в ночи'.`

Длина атома не может превышать 255 символов.

Целые числа. Целые числа в Прологе задаются обычным образом. Например, `1, 135, 0, -97`. Целые числа заключены между -2^{15} и $2^{15} - 1$. Положительные числа знаком «плюс» перед ними не сопровождаются.

Символ ASCII со стоящим перед ним обратным апострофом ``` тоже является целым числом (кодом этого символа). Например, вместо ``a` можно писать `97`, а вместо `97` – ``a`.

Вещественные числа. Вещественные числа (числа с плавающей запятой) могут иметь до 15 десятичных знаков после десятичной точки. Например, `0.6, 135.238, -17.259, -10.0`. Целая часть числа, десятичная точка и хотя бы одна цифра после десятичной точки обязательны. Возможна экспоненциальная запись вещественного числа. Порядок числа заключен между `-99` и `99`. Например, `-71.0E21, 212.34E -5, 0.35E -12`.

Пролог есть язык для символьных вычислений, поэтому числа, особенно вещественные, ему, вообще говоря, не свойственны.

Стринги. *Стринг* есть произвольная последовательность печатаемых символов алфавита, включая пробел (такую последовательность иногда называют текстовой константой), заключенная между знаками `$`. Например, `$Он сегодня позвонит мне$; $ты ничего не говорил об этом$`. Стринги используются при работе с текстами на естественных (или искусственных) языках. `$$` есть пустой стринг. Знак `$` внутри стринга пишется как `$$`. Например, `$Он должен мне $15$`.

Атомы, числа (целые и вещественные), строки являются константами.

Переменные. *Переменная* есть слово в алфавите из букв, цифр и символов подчеркивания, начинающееся с большой буквы или с символа подчеркивания.

Пример. X, Y21, AB5C, Input, Xyz, X_1_2_3, Head, Tail, List, Object.

Переменная, которой в процессе выполнения программы на компьютере не присваивалось никакого значения, называется *незамещенной переменной*. В противном случае переменная называется *замещенной*.

Термы

1. Всякий атом есть *терм*. Всякое число есть терм. Всякий строка есть терм.

2. Всякая переменная есть терм.

3. Если a есть буквенный атом, а каждое из $t_1, t_2, \dots, t_k$ есть терм, то $a(t_1, t_2, \dots, t_k)$ есть терм, в котором атом a называется *главным функтором*. Всякий функтор внутри терма главным уже не является. Терм $a(t_1, t_2, \dots, t_k)$ называется также *структурой* или *предикатом*. Буквенный атом в структуре в качестве главного функтора называется *именем предиката*. Два предиката с разным числом аргументов считаются разными предикатами, даже если они имеют одно и то же имя.

Термы без переменных являются *константами*. Структура без аргументов, т. е. нульместная структура, является буквенным атомом.

Пример. дата(1, май, 1989), июль, кафедра (Преподаватели, Инженеры, Лаборанты), $*(+(a, b), -(c, 5))$.

Говоря о конкретном терме, можно сказать о нем: терм-атом, терм-число, терм-переменная, терм-структура (или просто структура), предикатный терм (или предикатная структура), арифметический терм (с ним познакомимся позже).

Списки

1. Выражение $[]$ есть пустой список.

2. Если каждое из $L_1, L_2, \dots, L_k$ есть терм или список, то выражение $[L_1, L_2, \dots, L_k]$ есть список.

Первый элемент списка есть *голова* списка. Список без своего первого элемента называется *хвостом* списка. В списке голову от хвоста можно отделять разделителем $|$ явно (этот разделитель называют также конструктором).

Пример

Список	Голова списка	Хвост списка
$[]$	—	—
$[a, b, c]$	a	$[b, c]$
$[[], [], b, c]$	$[]$	$[[], b, c]$
$[[a, b], [X, c], d, Y]$	$[a, b]$	$[[X, c], d, Y]$
$[a]$	a	$[]$
$[[a, [b, c]], X, [d]]$	$[a, [b, c]]$	$[X, [d]]$
$[a[[b, c], a, [b, c]]]$	a	$[[b, c], a, [b, c]]$
$[X, b, [c, d] [X, b, [c]]] X$	$[b, [c, d], X, b, [c]]$	

Знаком | можно отделять любое число начальных элементов списка от остальной его части. Например, если список $L = [a,b,c,d,e]$ и $L = [X,Y|Z]$, то $X := a$, $Y := b$, $Z := [c,d,e]$. Здесь знак $:=$ означает присваивание (знак $:=$ Прологу не принадлежит).

В компьютере списки представляются с помощью бинарного функционального символа . (точка), взятого в апострофы.

Например, следующие записи представляют один и тот же список:

```
[a,b]           '(a,'(b,[]))
[a,b,c]         '(a,'(b,'(c,[])))
[a,b,c],a,d]    '(a,'(,'(b,'(c,[])),,'(a,'(d,[]))) [[a,b],c,d]
'('(a,'(b,[])),,'(c,'(d,[])))
```

При определении терма в структуре $a(t_1, \dots, t_k)$ каждое из $t_1, \dots, t_k$ может быть еще и списком.

Список кодов символов ASCII можно задавать как последовательность символов этого списка, заключенную в двойные апострофы. Например, следующие записи определяют один и тот же список:

```
"cfk"          [99,102,107]      '(99,'(102,'(107,[]))
```

Список без переменных есть константа.

В Прологе приняты следующие типы данных: атомы, числа (целые и вещественные), строки, переменные, структуры, списки.

Пролог-программа. Факт есть структура, заканчивающаяся точкой. В качестве фактов чаще всего используются предикатные структуры.

Если $A, B, C, \dots, D$ являются структурами (чаще всего предикатными), то выражение

$A : - B, C, \dots, D$

называется *правилом*; A есть голова правила; $B, C, \dots, D$ есть тело правила. Правило называется также *условным правилом*. Знак $:$ – читается как «если»; запятая в теле правила означает конъюнкцию. Точка с запятой в теле правила (если она есть) означает дизъюнкцию. *Клауза* есть факт или условное предложение.

С точки зрения логики условное предложение представляет собой формулу логики предикатов $B \ \& \ C \ \& \ \dots \ \& \ D \rightarrow A$, равносильную формуле $\neg B \vee \neg C \vee \dots \vee \neg D \vee A$; последняя формула является дизъюнктом, все дизъюнктивные слагаемые которого отрицательны, за исключением одного положительного; подобные дизъюнкты называются *хорновыми*. Так что всякое предложение Пролога является хорновым дизъюнктом, и программирование на Прологе есть программирование в хорновых дизъюнктах.

Если $F, G, \dots, H$ являются структурами, то выражение

$? - F, G, \dots, H$

называется *запросом*, а элементы $F, G, \dots, H$ – *целями*, или целевыми утверждениями запроса.

Предложение: в Прологе есть один из трех объектов: факт, правило, запрос. Всякое предложение Пролога состоит из головы и тела. Факт имеет только голову (и пустое тело). Правило имеет и голову, и тело. Запрос имеет только тело (и пустую голову). Точка в конце запроса и предложения обязательна.

Программа: в Прологе есть конечная последовательность фактов и правил. Иногда программу называют базой данных; иногда базой данных называют множество фактов в программе, а самую программу – базой знаний (БЗ). Программу в рабочей области (т. е. оттранслированную программу) тоже будем называть БД.

Процедура есть конечная последовательность фактов и правил, имеющих в голове один и тот же главный функтор.

Если переменная входит в предложение только один раз, то ее можно заменять символом подчеркивания; тогда говорят об *анонимной переменной*. Например, вместо предложения

имеет_ребенка(X) : – родитель(X,Y).

можно записать предложение

имеет_ребенка(X) : – родитель(X,_).

Каждое вхождение символа подчеркивания в предложении Пролог-программы символизирует новую переменную. Например, предложение

кто-то_имеет_ребенка : – родитель(,_).

означает то же самое, что и предложение

кто-то_имеет_ребенка : – родитель(X,Y).

Если анонимная переменная входит в запрос, то на нее Пролог сведений не выдает.

Область действия всякой переменной (в том числе и анонимной) есть то предложение, в которое эта переменная входит. Две одинаковые переменные в разных предложениях означают две разные переменные.

Описанные конструкции составляют так называемый чистый (или стандартный) Пролог, входящий в любую реализацию языка. Типы данных в различных реализациях Пролога могут быть разными.

Пролог-программу P составляет множество (конъюнкция) фактов и условий предложений. Программе можно ставить запросы. Пролог пытается установить, выводится ли запрос $G(x)$ из множества предложений программы: $P \vdash (\exists x)G(x)$. Если да, то попутно вычисляются значения вектора x предметных переменных, содержащихся в запросе, и в конце вычислений Пролог выдает эти значения (т. е. попутно вычисляется подстановка θ , для которой $P \supset G \circ \theta$). Если нет, то Пролог вычисления заканчивает и сообщает об этом.

Пример простейшей Пролог-программы. Приведем пример Пролог-программы, характеризующей семейные отношения (рис. 9.5):

родитель(мария, петр).
 родитель(иван, петр).
 родитель(иван, елена).
 родитель(петр, анна).
 родитель(петр, нина).
 родитель(нина, андрей).

Запись родитель(мария,петр) означает, что Мария есть родитель Петра. Приведенные предложения (они являются фактами) составляют Пролог-программу. К этой программе можно обращаться с запросами. Компьютер строит соответствующее дерево вывода и выдает ответ согласно результатам вычислений. Заметим, что знак ? – и стрелку → выдает компьютер; точку с запятой набирает на клавиатуре пользователь. Точка с запятой означает поиск альтернатив. При отсутствии очередной альтернативы компьютер выдает слово false. Приведем несколько запросов к компьютеру и ответов компьютера на них.

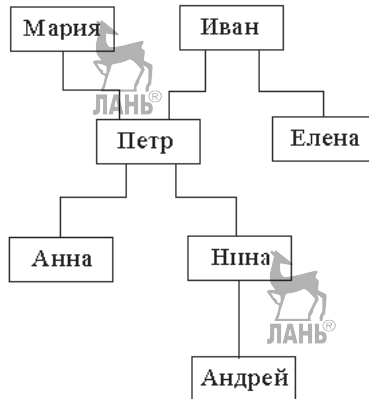


Рис. 9.5

? – родитель(мария,петр).
 true
 ? – родитель(иван,анна).
 false
 ? – родитель(нина,андрей).
 true
 ? – родитель(петр,Y).
 Y=анна ;
 Y=нина ;
 false
 ? – родитель(X,петр).
 X=мария ;
 X=иван ;
 false

Через запятую, которая в данном случае означает конъюнкцию, запрос можно составить из нескольких целей:

? – родитель(иван,X),родитель(X,Y).

X=петр

Y=анна ;

X=петр

Y=нина ;

false

Расширим программу о родственниках фактами об их поле.

женский(мария).

женский(елена).

женский(анна).

женский(нина).

мужской(иван).

мужской(петр).

мужской(андрей).



Определим предикат с именем потомок, имея в виду непосредственного потомка, с помощью следующего предложения:

потомок(X,Y) : – родитель(Y,X).

Знак : – соответствует логическому «если». В логике предикатов это предложение записали бы в виде следующей формулы:

$(\forall X)(\forall Y) (\text{родитель}(Y,X) \rightarrow \text{потомок}(X,Y))$, или

$(\forall X)(\forall Y) (\text{потомок}(X,Y) \leftarrow \text{родитель}(Y,X))$.

Введем также предикаты мать(X,Y), отец(X,Y), сестра(X,Y), брат(X,Y), бабушка(X,Y), дедушка(X,Y), бабдед(X,Y) с помощью следующих предложений.

мать(X,Y) : – родитель(X,Y),женский(X).

отец(X,Y) : – родитель(X,Y),мужской(X).

сестра(X,Y) : – родитель(Z,X),родитель(Z,Y),женский(X).

брат(X,Y) : – родитель(Z,X),родитель(Z,Y),мужской(X).

бабушка(X,Z) : – родитель(X,Y),родитель(Y,Z),женский(X).

дедушка(X,Z) : – родитель(X,Y),родитель(Y,Z),мужской(X).

бабдед(X,Z) : – родитель(X,Y),родитель(Y,Z).

Приведем несколько примеров запросов к программе и ответов компьютера на них.

? – потомок(петр,мария).

true

? – потомок(петр,X).

X=анна -> ;


```
X=нина -> ;
false
? – сестра(анна,анна).
true
? – брат(петр,елена).
true
? – брат(елена,петр).
false
```

Видим, что Анна может быть сестрой себе. Чтобы этого избежать, следует ввести предикат различия. Это мы сделаем позже.

Добавим к программе семейных отношений предикат предок(X,Y) с помощью следующей процедуры.

```
предок(X,Z) : – родитель(X,Z).
предок(X,Z) : – родитель(X,Y), предок(Y,Z).
```

Второе предложение соответствует формуле логики предикатов

$(\forall X)(\forall Y)(\forall Z) (\text{родитель}(X,Z), \text{предок}(Y,Z) \rightarrow \text{предок}(X,Z)),$

эквивалентной формуле

$(\forall X)(\forall Z)((\exists Y)(\text{родитель}(X,Z), \text{предок}(Y,Z)) \rightarrow \text{предок}(X,Z)),$

или же формуле

$(\forall X)(\forall Z) (\text{предок}(X,Z) \leftarrow (\exists Y)(\text{родитель}(X,Z), \text{предок}(Y,Z))).$

Поэтому если переменная в теле предложения не содержится среди переменных головы этого предложения, то эта переменная находится под знаком квантора существования.

При вычислении значения предиката предок произойдет обращение к этому же предикату, но уже с другими значениями переменных. Такое задание предиката называется рекурсивным.

9.6.1. Унификация в Прологе

Унификация есть основной инструмент передачи параметров из одного предложения в другое предложение при исполнении программы. Унификация в Прологе есть некоторое ограничение теоретической унификации, и осуществляется она с помощью операций согласования и присваивания. Если в процессе выполнения программы переменной в процессе унификации присваивается какое-либо значение, то говорят также, что переменная конкретизируется (этим значением), переменная замещается (на это значение). Такая переменная называется тогда *конкретизированной*, или *замещенной*.

Если же переменной в процессе вычислений не присваивалось никаких значений, то переменная называется *неконкретизированной*, или *незамещенной*.

Атомы согласуются, только если они одинаковы. Целые числа согласуются только с равными им целыми числами. Вещественные числа согласуются только с равными им вещественными числами. Строинги согласуются только с равными им строингами. Ссылочные числа базы данных согласуются только с равными им ссылочными числами. Незамещенная переменная согласуется с любой другой незамещенной переменной.

Две незамещенные переменные после согласования называются связанными. В процессе выполнения программы может оказаться несколько связанных в результате согласования переменных. Если одна из связанных переменных принимает какое-либо значение (т. е. замещается на какое-то значение), то и все другие с ней связанные переменные принимают это же значение (т. е. замещаются на такое же значение).

Две замещенные переменные согласуются, если согласуются значения, которыми эти переменные замещены.

Два терма согласуются в следующих случаях.

1. Термы идентичны.
2. Незамещенным переменным в обоих термах могут быть присвоены такие объекты, после подстановки которых вместо соответствующих переменных термы становятся идентичными.

При проверке согласуемости двух термов возможны следующие случаи.

Термы не согласуются. Тогда согласование безуспешно.

Термы согласуются. Тогда согласование успешно, и переменным в обоих термах производится присваивание таких значений, или, что есть то же самое, переменные замещаются на такие значения, определяемые наиболее общим унификатором этих термов, при которых термы становятся идентичными. Согласуемость и последующее присваивание переменным осуществляются с помощью алгоритма унификации.

В Прологе действуют следующие правила согласования и присваивания (т. е. правила передачи параметров в результате унификации).

1. Если термы S и T есть константы, то они согласуемы, если они одинаковы.
2. Если S есть незамещенная переменная, а T есть какой-либо терм, не являющийся переменной, то S и T согласуемы и S присваивается T (т. е. S замещается на T). Обратно, если T есть незамещенная переменная, а S есть терм, не являющийся переменной, то S и T согласуемы и T присваивается S . Если термы S и T являются незамещенными переменными, то эти переменные являются связанными, а их значения будут одинаковыми, если одна из них станет конкретизированной.

3. Если термы S и T являются структурами, то они согласуемы при выполнении следующих двух условий.

- S и T имеют одинаковые главные функторы;
- соответствующие аргументы в S и T согласуемы.

Конечное присваивание определяется результатами согласования и присваивания соответствующих аргументов в S и T .

4. Списки согласуются, если согласуются их соответствующие элементы.

5. Два правила (два условных предложения) согласуются, если согласуются их головы и тела, т. е. согласуются их соответствующие цели. В частности, два факта согласуются, если они согласуются как две предикатные структуры (по п. 3).

9.6.2. Декларативный и операторный смысл Пролог-программы

Пролог-программа лишь декларирует характер тех утверждений, которые подлежат вычислению. С первого взгляда не ясно, как и в каком порядке будут проводиться вычисления. Организация вычислений передается транслятору. Пользователю не надо заботиться о написании порядка, в котором будут проводиться вычисления; это сделает за него компьютер согласно заложенному в транслятор предписанию. Это является достоинством Пролога. Впрочем, это может оказаться и его недостатком, ибо иногда для повышения эффективности вычислений следовало бы предусмотреть средства организации вычислений самим пользователем (и Пролог имеет такие средства).

Декларативный смысл программы определяет, следует ли логически данный запрос из предложений программы, и если да, то при каких значениях переменных запроса. Если да, то говорят также, что запрос выполним, запрос истинен, запрос успешен, запрос удовлетворяется, запрос согласуется с программой, запрос согласуется с базой данных.

Операторный смысл программы состоит в том, как Пролог вычисляет ответ на запрос к программе. Запрос задается в виде предложения $? - G_1, G_2, \dots, G_m$. Термы $G_1, G_2, \dots, G_m$ есть цели запроса. Ответ на этот запрос состоит в том, что Пролог отслеживает первую успешную ветвь дерева вывода целей запроса из программы и выдает соответствующие значения переменных запроса. Дерево вывода строится индуктивно в ширину следующим образом. В корне дерева вывода помещается исходный запрос $(G_1, G_2, \dots, G_m)\sigma_0$. Подстановка $\sigma_0 = \varepsilon$, т. е. σ_0 есть пустая подстановка (никакой подстановки). Запись $(G_1, G_2, \dots, G_m)\sigma_0$ означает, что подстановка σ_0 действует на все цели запроса $G_1, G_2, \dots, G_m$.

Предположим, что на некотором шаге построен фрагмент T дерева вывода. Построим следующий фрагмент. Пусть очередной обрабатываемый узел дерева вывода помечен запросом $(A_1, \dots, A_k)\sigma_i$, где $\sigma_i = \theta_0 \dots \theta_i$ ($\theta_0 = \sigma_0$) есть композиция подстановок, примененных к запросу до текущего шага. Чтобы построить очередной фрагмент дерева вывода, выбирается цель $A_1\sigma_i$ – первая в запросе $(A_1, \dots, A_k)\sigma_i$. В программе берется предложение D вида $E :- F_1, \dots, F_l$, его переменные переименовываются так, чтобы они отличались от всех ранее встречающихся переменных, и голова E унифицируется с $A_1\sigma_i$.

Если унификация успешна, то вычисляется подстановка $\theta_{i+1} = \text{НОУ}(A_1\sigma_i, E)$ и формируется следующий запрос $(F_1, \dots, F_l, A_2, \dots, A_k)\sigma_{i+1}$, где $\sigma_{i+1} = \sigma_i \circ \theta_{i+1}$ (из-за переименования переменных $F_j\sigma_i = F_j$, $F_j\sigma_{i+1} = F_j\theta_{i+1}$). Если подцель $A_1\sigma_i$ может унифицироваться с p предложениями программы, то от обрабатываемого узла фрагмента T вниз отходит p ребер с концевыми узлами, помеченными соот-

ветствующими запросами. Если унификация с некоторым предложением программы безуспешна, то соответствующий концевой узел помечается буквой f (fail). Если унификация дает пустой дизъюнкт $\square$, то соответствующий концевой узел помечается знаком $\square$.

Узлы, помеченные знаками f и $\square$, окончательные, и от них дальнейшее построение дерева вывода не проводится. Очередной фрагмент дерева вывода построен. Шаг индукции выполнен. Дерево вывода построено. Оно может оказаться конечным, и тогда все его концевые вершины помечены знаками f и $\square$. Некоторые ветви построенного дерева могут оказаться и бесконечными. В последующем дадим примеры построения конкретных деревьев вывода в ответ на конкретные запросы к конкретным программам.

Пролог осуществляет обход дерева вывода в глубину. При этом осуществляется бэктрекинг: при попадании в узел дерева, помеченный буквой f (безуспешная ветвь), Пролог автоматически осуществляет отход в ближайший верхний узел, от которого можно перейти на следующую ветвь дерева вывода. Вычисление оканчивается успешно, если Пролог попадает в концевой узел дерева, помеченный знаком $\square$. Пролог выдает в качестве решения все полученные значения переменных запроса (последнюю подстановку θ) и предлагает пользователю осуществить бэктрекинг для поиска других возможных решений.

Проследим, как Пролог, обходя дерево вывода в глубину сверху вниз слева направо, отслеживает (строит) успешную его ветвь, чтобы вычислить и выдать ответ на исходный запрос $? - G_1, \dots, G_m$. Чтобы вычислить ответ на этот запрос, выполняются следующие операции.

1. Если список запросов пуст, то вычисление успешно (выведен пустой дизъюнкт).

2. Если список запросов не пуст, то выполняется следующая последовательность операций.

Просматривается список предложений программы сверху вниз, пока не встретится предложение C , голова которого согласуется с целью G_1 . Цель G_1 выступает как образец, а предложение C программы сравнивается с этим образцом. Если такого предложения C нет, то попытка вычисления безуспешна; Пролог сообщает об этом, и работа транслятора прекращается. Если такое предложение C есть, то пусть оно имеет вид:

$$H : - B_1, B_2, \dots, B_n.$$

Переменные в C переименовываются так, что получившееся предложение C_1 и цели $G_1, G_2, \dots, G_m$ не имеют общих переменных. Пусть клауза C_1 имеет вид:

$$H_1 : - B_{11}, B_{21}, \dots, B_{n1}.$$

Унифицируем G_1 и H_1 (т. е. с помощью алгоритма унификации найдем их НОУ) и осуществим присваивания переменным в G_1 и H_1 тех значений, которые найдены в соответствии с результатами согласования. Если эти перемен-

ные встречаются в $C_1, G_1, G_2, \dots, G_m$, то им присваиваются те же значения. В результате предложение C_1 примет вид клаузы C_2 :

$$H_2 : - B_{12}, B_{22}, \dots, B_{n2}.$$

Исходный запрос превратится в запрос

$$? - B_{12}, B_{22}, \dots, B_{n2}, G_{22}, G_{32}, \dots, G_{m2}.$$



Далее следует возврат к п. 2, но уже со вновь сформированным запросом.

Описанный алгоритм может закончить работу успешно только п. 1. Тогда Пролог выдает вычисленные значения переменных начального запроса и предлагает пользователю продолжить вычисления для нахождения других возможных значений переменных запроса, если таковые значения имеются. Если же их нет, то Пролог сообщает об этом и прекращает работу. Процесс может попасть на бесконечную вычислительную ветвь дерева вывода; тогда Пролог прекращает вычисления из-за исчерпания объема стека, или вычисления будут продолжаться циклически неограниченно долго, пока пользователь не прекратит это. В следующем параграфе на конкретных примерах покажем, как организуется в Прологе вычислительный процесс.

9.6.3. Бэктрекинг и оператор отсечения

Пролог, вычисляя ответ на запрос, строит вывод, пытаясь установить, следует ли логически запрос из предложений программы. При этом Пролог осуществляет бэктрекинг: при получении безуспешного вывода по одной из ветвей дерева вывода Пролог автоматически переходит к построению другой его ветви. При получении успешного вывода по одной из ветвей дерева вывода Пролог для поиска других возможных решений предлагает осуществить бэктрекинг пользователю. Если последнее предложение принимает, нажимая клавишу ; (точка с запятой), то Пролог продолжает вычисления. Если нет, то пользователь нажимает любую другую клавишу, например клавишу ввода, и тогда Пролог вычисления прекращает. Таким способом можно обойти все дерево вывода запроса из предложений программы сверху вниз и слева направо и получить все решения, которые возможны.

Осуществляя бэктрекинг, т. е. возврат из некоторого узла дерева вывода в другой, вышерасположенный узел (ближе к корню), Пролог при этом отходе отменяет все присваивания переменным, осуществленным на этом пути ранее при движении от корня дерева вывода между этими двумя узлами. Так что Пролог, обходя дерево вывода (на прямом, активном участке пути), удаляясь от корня, выполняет вычисления согласно программе, а осуществляя отход (на обратном, пассивном участке пути), отменяет сделанные вычисления, чтобы испытать альтернативные варианты согласования цели с другими возможными предложениями программы, имеющими тот же главный функтор, что и вычисляемая цель.

Находясь в каком-либо промежуточном узле дерева вывода и обрабатывая в этом узле очередную цель G , Пролог получает некоторое решение (т. е. некоторые значения переменных), удовлетворяющее этой цели. Возвратившись при бэктрекинге снова в узел исполнения цели G , Пролог при каждом таком возврате автоматически начинает испытывать возможные согласования цели G с другими предложениями программы, пытаясь найти альтернативные решения, удовлетворяющие цели G , даже если этого и не требуется, ибо по смыслу решаемой задачи пользователю, быть может, достаточно получить только одно решение, удовлетворяющее цели G . Поэтому бэктрекинг может снижать эффективность программы. Избежать этой неэффективности позволяет оператор отсечения (он обозначается восклицательным знаком: $!$), принудительно отсекающий поиск альтернативных решений для цели G после первого же успешного исполнения G .

Следует отметить, что в чистом Прологе предложения программы часто перестановочны. Использование оператора отсечения может имеющуюся перестановочность нарушить.

Пусть очередная исполняемая цель запроса унифицируется (согласуется) с головой предложения, содержащего оператор отсечения. Результат унификации вместе с оператором отсечения перейдет в новый запрос. Когда при последовательном выполнении целей запроса оператор отсечения встречается, наконец, как цель, то эта цель всегда совместима с БД. При этом консервируются все те значения, которые присвоены переменным в целях запроса, расположенных в унифицируемом предложении программы между головой предложения и знаком $!$ оператора отсечения (голова включается). Все другие альтернативы присваивания не рассматриваются (т. е. отсекаются). Поясним сказанное на примере предложения C :

$H : - B_1, B_2, \dots, B_m, !, \dots, B_n.$

Пусть очередная исполняемая цель G запроса согласуется с H . Термы G и H имеют один и тот же главный функтор. Цели $B_1, B_2, \dots, B_m, !, \dots, B_n$ переходят в новый запрос и исполняются последовательно друг за другом, в том числе цели от B_1 до B_m ; при этом определяются решения этих целей, т. е. определяются значения переменных, удовлетворяющих этим целям. Наконец, исполняется цель $!$. При исполнении оператора отсечения $!$ система уже нашла решения целей $H, B_1, B_2, \dots, B_m$. После исполнения оператора $!$ эти текущие решения замораживаются, и все другие возможные решения целей $H, B_1, B_2, \dots, B_m$ при бэктрекинге, т. е. при возвращении от некоторого узла дерева вывода в узлы исполнения целей $H, B_1, B_2, \dots, B_m$, не рассматриваются (отсекаются), хотя в программе возможны другие предложения, согласующиеся с целями $H, B_1, B_2, \dots, B_m$. Эти предложения могли бы дать альтернативные решения данных целей.

9.6.4. Объявление операторов

Пролог допускает привычную инфиксную запись арифметических выражений (термов). Внутри компьютера та запись переводится в префиксную форму запи-

си, в которой знак операции предшествует операндам. Например, термы $a+b$, $a-b$, $a*b$, a/b транслятор переводит в $+(a,b)$, $-(a,b)$, $*(a,b)$, $/(a,b)$ соответственно.

В Прологе принят общепринятый приоритет арифметических операций, при котором умножение и деление связывают сильнее сложения и вычитания. Например, арифметический терм $2*a+b*c$ Пролог воспринимает как $(2*a)+(b*c)$. Внутри компьютера транслятор переведет этот терм в запись $+(*(2,a),*(b,c))$. Впрочем, Пролог в состоянии принять вводимую префиксную запись арифметических термов сразу с клавиатуры.

Пролог предоставляет программисту возможность в пределах своей программы самому определять нужные ему одноместные и двуместные операции с заданным приоритетом. Это делается с помощью предложения, которое имеет следующий вид:

`ор(<приоритет>,<спецификация>,<имя оператора>).`

Приоритет определяемого оператора есть натуральное число от нуля до 1200. Чем больше число, тем выше приоритет. При отсутствии скобок в терме операции с более высоким приоритетом выполняются раньше, чем операции с меньшим приоритетом.

Приоритет аргумента равен нулю, если аргумент заключен в скобки или если аргумент не является структурой. Если аргумент есть структура (не заключенная в скобки), то приоритет этого аргумента равен приоритету его главного функтора.

Спецификация определяемого оператора строится из букв f , x , y и имеет один из следующих видов:

fx , fy , xf , yf , xfx , xfy , yfx .

Буква x при f означает, что аргумент оператора f должен иметь приоритет строго меньше, чем приоритет самого оператора f .

Буква y при f означает, что аргумент оператора f должен иметь приоритет, равный или меньший, чем приоритет самого оператора f .

Запись fx означает, что одноместный оператор f префиксный, т. е. f стоит перед своим аргументом, при этом приоритет аргумента должен быть строго меньше, чем приоритет оператора f . Наличие нескольких рядом стоящих операторов f не допускается.

Запись fy означает, что одноместный оператор f префиксный; при этом приоритет аргумента должен быть равен или меньше, чем приоритет оператора f . Оператор f правоассоциативный, т. е. при наличии нескольких рядом стоящих операторов f скобки группируются справа.

Запись xf означает, что одноместный оператор f постфиксный, т. е. f стоит после своего аргумента, при этом приоритет аргумента должен быть строго меньше, чем приоритет оператора f . Наличие нескольких рядом стоящих операторов f не допускается.

Запись yf означает, что одноместный оператор f постфиксный, при этом приоритет аргумента должен быть равен или меньше, чем приоритет операто-

ра f . Оператор f левоассоциативный, т. е. при наличии нескольких рядом стоящих операторов f скобки группируются слева.

Запись xfx означает, что двуместный оператор f инфиксный, т. е. стоит между двумя своими аргументами. Слева и справа от f должны стоять аргументы, приоритеты которых должны быть строго меньше, чем приоритет оператора f . Наличие нескольких рядом стоящих операторов f не допускается.

Запись xfy означает, что двуместный оператор f инфиксный. Слева от f должен стоять аргумент, приоритет которого строго меньше, чем приоритет оператора f . Справа от f должен стоять аргумент, приоритет которого должен быть равен или меньше, чем приоритет оператора f . Оператор f левоассоциативный, т. е. при наличии нескольких рядом стоящих операторов f , перемежаемых аргументами, скобки (при их отсутствии) группируются слева.

Запись yfx означает, что двуместный оператор f инфиксный. Справа от f должен стоять аргумент, приоритет которого строго меньше, чем приоритет оператора f . Слева от f должен стоять аргумент, приоритет которого должен быть равен или меньше, чем приоритет оператора f . Оператор f правоассоциативный, т. е. при наличии нескольких рядом стоящих операторов f , перемежаемых аргументами, скобки (при их отсутствии) группируются справа.

Пример. Определим отрицание, конъюнкцию, дизъюнкцию, импликацию, эквивалентность с помощью следующих предложений.

```

or(200, fy, ^).
or(180, yfx, &).
or(160, yfx, \).
or(140, yfx, - ->).
or(120, yfx, < - ->).
```

Для подробных сведений о работе с языком следует обратиться к документации, поставляемой фирмой вместе с языком.

9.7. Примеры программ и вычислений в Прологе

9.7.1. Принадлежность элемента списку

Предикат `member(X,Y)` истинен, если X есть элемент списка Y . Например:

```

member(a,[b,c,a,d]) истинно;
member(a,[b,c,d]) ложно;
member(b,[]) ложно;
member([a,b],[c,[a,b],e,d]) истинно;
member([a,b],[c,[b,a],e,d]) ложно.
```

Множества в Прологе представляются списками. Списки могут иметь повторы элементов. Программисту следует заботиться о том, чтобы используемые для представления множеств списки этих повторов не имели.

Предикат $\text{member}(X,Y)$ задается следующей процедурой:

```
member(X,[X|Y]).
member(X,[Y|Z]) :- member(X,Z).
```

Рассмотрим ход вычислений в Прологе в ответ на запрос $\text{member}(c,[b,a,c,d])$. Построим соответствующее дерево вывода (рис. 6.6).

Процедура вычисления предиката member состоит из следующих предложений:

```
C1. member(X,[X|Y]).
C2. member(X,Z) :- member(X,[Y|Z]).
```

Запрос:

```
C3. member(c,[b,a,c,d]).
```

Пусть знак $:=$ означает назначение (т. е. присваивание) переменным.

Узел 0. Запрос C3 помещается в узел 0 (т. е. в корень) дерева вывода (рис. 9.6).

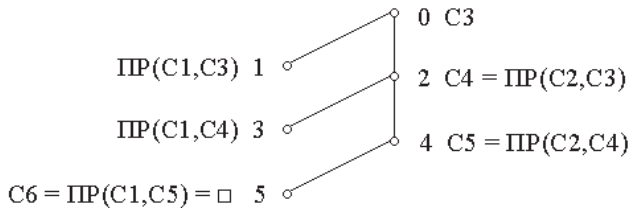


Рис. 9.6

Узел 1. Попытка унификации дизъюнктов C3 и C1. Присваивание

$X := c, X := b, Y := [a,c,d]$

противоречиво. Попытка унификации не удалась. В узле 1 дерева вывода вычисление безуспешно. Осуществляется бэктрекинг, т. е. отход в узел 0. Сделанные ранее на пути от узла 0 к узлу 1 присваивания переменным отменяются. Испытывается возможность согласования запроса C3 с предложением программы.

Узел 2. Попытка унификации дизъюнктов C3 и C2. Присваивание

$X := c, Y := b, Z := [b,c,d]$

непротиворечиво. Формируется новый запрос, который есть правая часть C2 при найденных значениях переменных.

$C4 = \text{member}(c,[a,c,d]),$

В узле 2 помещается цель C4.

Узел 3. Проводится попытка унификации C4 и C1. Присваивание

$$X := c, X := a, Z := [c,d]$$

противоречиво. Попытка унификации не удалась. В узле 3 вычисление безуспешно. Осуществляется отход в узел 2; сделанные ранее на пути от узла 2 к узлу 3 присваивания переменным отменяются.

Узел 4. Проводится попытка унификации C4 и C2. Присваивание

$$X := c, Y := a, Z := [c,d]$$

непротиворечиво. Формируется новый запрос, который есть правая часть C2 при найденных значениях переменных.

$$C5 = \text{member}(c, [c,d]),$$

В узле 4, где вычисление успешно, помещается цель C5.

Узел 5. Проводится попытка унификации C5 и C1.

Присваивание $X := c, X := c, Y := [d]$ непротиворечиво. Формируется новый запрос, который есть правая пустая часть C1 при найденных значениях переменных.

$$C6 = \square$$

Выведен пустой дизъюнкт. В узле 5 вычисление закончилось успешно. Ответ на исходный запрос об истинности $\text{member}(c, [b,a,c,d])$ утвердительный, о чем компьютер сообщает пользователю. Транслятор прекращает работу.

Проследим ход вычислений в Прологе в ответ на запрос $\text{member}(X, [b,a])$. Построим соответствующее дерево вывода (рис. 9.7). Предложения программы:

C1. $\text{member}(X, [X|Y]).$

C2. $\text{member}(X, Z) :- \text{member}(X, [Y|Z]).$

Запрос:

C3. $\text{member}(X, [b,a]).$

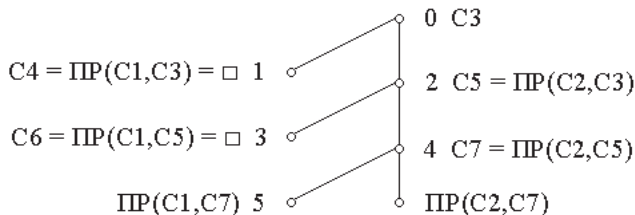


Рис. 9.7

Узел 0. Запрос C3 помещается в узел 0 дерева вывода.

Узел 1. Унификации C3 и C1. Присваивание $X := b, Y := [a]$ непротиворечиво. Формируется новый запрос, который есть правая пустая часть C1 при найденных значениях переменных.

$$C4 = \square$$

Вычисление в узле 1 закончилось успешно. Система выдает решение $X = b$ и предлагает (с помощью бэктрекинга) найти другие возможные решения. Вычисления продолжаются нажатием клавиши ; (точка с запятой). Осуществляется отход в узел 0 дерева вывода. Сделанные ранее на пути от узла 0 к узлу 1 присваивания переменным отменяются.

Узел 2. Унификации C3 и C2. Присваивание

$Y := b, Z := [a]$

непротиворечиво. Формируется новый запрос, который есть правая часть C1 при найденных значениях переменных.

$C5 = \text{member}(X, [a]).$

Узел 3. Унификации C5 и C1. Присваивание $X := a, Y := []$ непротиворечиво. Формируется новый запрос, который есть правая пустая часть C1 при найденных значениях переменных.

$C6 = []$

Выведен пустой дизъюнкт. Система выдает новое решение $X = a$ и предлагает отыскать другие решения.

Вычисления продолжаются нажатием клавиши точка с запятой. Осуществляется отход в узел 2. Все сделанные ранее на этом пути назначения переменным отменяются.

Узел 4. Унификации C5 и C2. Присваивание $Y := a, Z := []$ непротиворечиво. Формируется новый запрос, который есть правая часть C2 при найденных значениях переменных.

$C7 = \text{member}(X, []).$

Узлы 5, 6. Так как пустой список не имеет ни головы, ни хвоста, то попытка унификации C1 и C7 безуспешна. По этой же причине безуспешна попытка унификации C2 и C7. Вычисления в узлах 5 и 6 дерева вывода безуспешны. Решений больше нет. Система сообщает об этом пользователю и прекращает работу.

9.7.2. Первый элемент в списке

Предикат $\text{first}(X, Y)$ истинен, если X есть первый элемент списка Y . Например:

$\text{first}(a, [a, b, c])$ истинно;
 $\text{first}(a, [b, a, c])$ ложно;
 $\text{first}([a, b], [[a, b], c, d])$ истинно;
 $\text{first}([a, b], [[b, a], c, d])$ ложно.

Предикат $\text{first}(X, Y)$ задается следующей процедурой, состоящей из единственного предложения.

$\text{first}(X, [X|Y]).$

Рассмотрим ход вычисления в Прологе в ответ на запрос $\text{first}(X, [a, a, b])$.

Процедура вычисления предиката first состоит из формулы логики предикатов:

C1. $\text{first}(X, [X|Y])$

Запрос:

C2. $\text{first}(X, [a, a, b])$.

Проводится попытка унификации C1 и C2. Присваивание

$X := a, Y := [a, b]$

непротиворечиво. Дизъюнкт

$C3 = \text{PP}(C1, C2) = \square$

Получено решение $X = a$. Других возможностей унификации нет. Система сообщает об отсутствии других решений и прекращает работу.

9.7.3. Последний элемент в списке

Предикат $\text{last}(X, Y)$ истинен, если X есть последний элемент списка Y . Например:

$\text{last}(c, [a, b, c])$ истинно;

$\text{last}(c, [a, c, b])$ ложно.

Предикат $\text{last}(X, Y)$ задается следующей процедурой.

$\text{last}(X, [X])$.

$\text{last}(X, [Z|Y]) :- \text{last}(X, Y)$.

На рис. 9.8 приведено дерево вывода в ответ на помещенный в узел 0 дерева вывода запрос $\text{last}(X, [a, s, d])$. Ниже следуют комментарии.

C1. $\text{last}(X, [X])$.

C2. $\text{last}(X, [Z|Y]) :- \text{last}(X, Y)$.

C3. $\text{last}(X, [a, s, d])$.

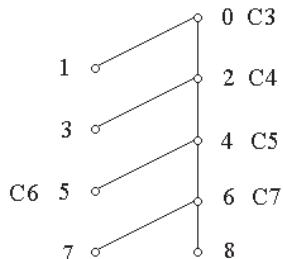


Рис. 9.8

В узле 1 попытка провести унификацию C1 и C3 безуспешна. Отход от узла 1 в узел 0. Попытка провести унификацию C2 и C3 успешна.

В узле 2 формируется запрос

$$C4 = \text{ПР}(C2, C3) = \text{last}(X, [s, d]).$$

Попытка провести унификацию C1 и C4 в узле 3 безуспешна. Отход от узла 3 в узел 2. Попытка провести унификацию C2 и C4 успешна. В узле 4 формируется запрос

$$C5 = \text{ПР}(C2, C4) = \text{last}(X, [d]).$$

Унификация C1 и C5 успешна. В узле 5 формируется запрос

$$C6 = \text{ПР}(C1, C5) = \square$$

Решение $X = d$. При бэктрекинге осуществляется отход от узла 5 в узел 4 и проводится успешная попытка унификации C2 и C5. В узле 6 формируется запрос

$$C7 = \text{ПР}(C2, C5) = \text{last}(X, []).$$

Попытки провести унификацию для C1 и C7 в узле 7, равно как и для C2 и C7 в узле 8, безуспешны. Других решений нет. Система сообщает об этом и прекращает работу.

9.7.4. Следующий элемент в списке

Предикат $\text{next}(X, Y, L)$ истинен, если элемент Y непосредственно следует за элементом X в списке L. Например:

$\text{next}(a, b, [s, h, a, b, c, d])$ истинно; $\text{next}(a, b, [s, h, a, c])$ ложно.

Предикат $\text{next}(X, Y, L)$ задается следующей процедурой:

$\text{next}(X, Y, [X, Y|Z]).$

$\text{next}(X, Y, [U|Z]) \text{ :- } \text{next}(X, Y, Z).$

Рассмотрим ход вычислений в Прологе в ответ на запрос $\text{next}(2, V, [1, 2, 3, 2, 5])$; построим соответствующее дерево вывода (рис. 9.9). Процедура вычисления предиката next состоит из следующих двух предложений

C1. $\text{next}(X, Y, [X, Y|Z]).$

C2. $\text{next}(X, Y, [U|Z]) \text{ :- } \text{next}(X, Y, Z).$

и запроса

C3. $\text{next}(2, V, [1, 2, 3, 2, 5]).$



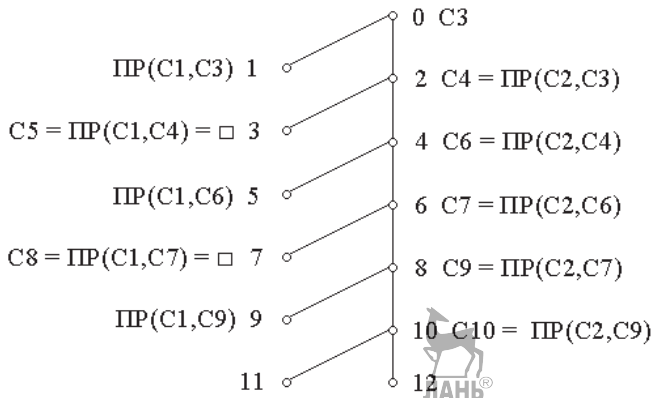


Рис. 9.9

Транслятор работает следующим образом.

Узел 0. Запрос $C3$ помещается в узел 0 дерева вывода.

Узел 1. Проводится попытка унификации дизъюнктов $C1$ и $C3$. Возникающее из сравнения $C1$ и $C3$ присваивание

$$X := 2, Y := V, X := 1, Y := 2, Z := [3, 2, 5]$$

противоречиво. В узле 1 попытка унификации не удалась. В узле 1 дерева вывода вычисление безуспешно. Осуществляется отход в узел 0. Сделанные ранее на пути от узла 0 к узлу 1 присваивания переменным отменяются. Испытывается возможность согласования запроса $C3$ со следующим предложением программы.

Узел 2. Проводится попытка унификации дизъюнктов $C2$ и $C3$. Присваивание

$$X := 2, Y := V, U := 1, Z := [2, 3, 2, 5]$$

непротиворечиво. Формируется новый запрос, который есть правая часть $C2$ при найденных значениях переменных.

$C4.\text{next}(2, V, [2, 3, 2, 5]).$

Узел 3. Проводится попытка унификации $C1$ и $C4$.

Присваивание $X := 2, Y := V, X := 2, Y := 3, Z := [2, 5]$ непротиворечиво. Формируется новый запрос, который есть правая пустая часть $C1$ при найденных значениях переменных.

$$C5 = \text{PR}(C1, C4) = \text{PR}(C12, C4) = \square.$$

Элемент $V = 3$ следует в данном списке за элементом 2. Решение $V = 3$ сообщается пользователю.

Транслятор (по желанию пользователя) работает дальше, пытаясь найти в исходном списке другие элементы, следующие за двойкой. Осуществляется отход от узла 3 в узел 2.

Узел 4. Проводится попытка унификации C2 и C4. Присваивание

$X := 2, Y := V, U := 2, Z := [3, 2, 5]$

непротиворечиво. Формируется новый запрос, который есть правая часть C2 при найденных значениях переменных:

$C6 = \text{ПР}(C2, C4) = \text{next}(2, V, [3, 2, 5]),$

являющийся новым запросом.

Узел 5. Проводится попытка унификации C1 и C6. Присваивание

$X := 2, Y := V, X := 3, Y := 2, Z := [5]$

противоречиво. Отход в узел 4.

Узел 6. Проводится попытка унификации C2 и C6. Присваивание

$X := 2, Y := V, Z := [2, 5]$

непротиворечиво. Формируется новый запрос, который есть правая часть C2 при найденных значениях переменных.

$C7 = \text{ПР}(C2, C6) = \text{next}(2, V, [2, 5]).$

Узел 7. Проводится попытка унификации C1 и C7. Присваивание

$X := 2, Y := V, X := 2, Z := []$

непротиворечиво. Формируется новый запрос, который есть правая пустая часть C1 при найденных значениях переменных.

$C8 = \text{ПР}(C1, C7) = \square$

Найден еще один элемент $V = 5$, следующий в исходном списке за элементом 2, о чем тоже сообщается пользователю.

Транслятор продолжает работу. Отход от узла 7 в узел 6.

Узел 8. Проводится попытка унификации C2 и C7. Присваивание

$X := 2, Y := V, U := 2, Z := [5]$

непротиворечиво. Формируется новый запрос, который есть правая часть C2 при найденных значениях переменных.

$C9 = \text{ПР}(C2, C7) = \text{next}(2, V, [5]).$

Узлы 9, 10, 11, 12. Попытка унификации C9 и C1 безуспешна. Попытка унификации C9 и C2 успешна. Формируется новый запрос, который есть правая часть C2 при найденных значениях переменных.

$C10. \text{next}(2, V, []).$

Так как пустой список в C10 не имеет ни головы, ни хвоста, то попытка унификации C10 и C1, а также C10 и C2 безуспешны. Компьютер сообщает, что решений больше нет. Вычисления закончены.

9.7.5. Соединение списков

Предикат $\text{append}(L1, L2, L3)$ истинен, если список $L3$ получается приписыванием списка $L2$ к списку $L1$. Например:

$\text{append}([a,b],[c,d,e],[a,b,c,d,e])$ истинно;
 $\text{append}([a],[b,c],[b,a,c])$ ложно.



Если $X = [a,[b,c]]$, $Y = [g,h]$, то $\text{append}(X,Y,Z)$ истинно при $Z = [a,[b,c],g,h]$.
 Предикат $\text{append}(L1, L2, L3)$ задается следующей процедурой.

$\text{append}([], L, L).$
 $\text{append}([X|L1], L2, [X|L3]) :- \text{append}(L1, L2, L3).$

Обозначим предложения программы через $C1$, $C2$ соответственно и проследим ход вычислений в ответ на запрос

$C3. \text{append}([a,b,c],[d,e],L).$

Сопроводим вычисления краткими пояснениями. Пусть $Y(C1, C2)$ означает «унификация $C1$ и $C2$ ».

$Y(C3, C1)$ безуспешна по первому аргументу.

$Y(C3, C2)$ успешна.

$X := a, L1 := [b,c], L2 := [d,e], L := [a|L3].$

Новый запрос:

$C4. \text{append}([b,c],[d,e],L3).$

$Y(C4, C1)$ безуспешна по первому аргументу.

$Y(C4, C2)$ успешна.

$X := b, L1 := [c], L2 := [d,e], L3 := [b|L31].$

Новый запрос:

$C5. \text{append}([c],[d,e],L31).$

$Y(C5, C1)$ безуспешна по первому аргументу.

$Y(C5, C2)$ успешна.

$X := c, L1 := [], L2 := [d,e], L31 := [c|L32].$

Новый запрос:

$C6. \text{append}([], [d,e], L32).$



$Y(C6, C1)$ успешна. $L32 := [d,e].$

Далее последовательно получаем: $L31 := [c|L32] = [c,d,e].$

$L3 := [b|L31] = [b,c,d,e]. L := [c|L3] = [a,b,c,d,e].$

Транслятор выдает ответ $L = [a,b,c,d,e]$ и прекращает работу.

Предикат `appendk(S,L)` истинен, если список `L` есть объединение элементов в списках из `S`. Например, если `S = [[a,b,c, d],[e,f],[q, r,t]]`, то `L=[a,b,c,d,e,f,q,r,t]`. Предикат `appendk(S,L)` задается следующей процедурой.

```

appendk(S,L) :- length(S,K),K>1,append1k(S,L).
appendk([H|[]],[H|[]]).
appendk([],[]).
append1k([H1,H2|T],L) :-
    append(H1,H2,L1),append2k(L1,T,L).
append2(L1,[H|T],L) :- append(L1,H,L2),append2k(L2,T,L).
append2k(L,[],L).

```

9.7.6. Обращение списка

Предикат `reverse(L,S)` истинен, если список `S` есть обращение списка `L`. Например:

```

reverse([a,b,c],[c,b,a]) истинно;
reverse([a,b,c],[c,d,a]) ложно;
reverse([a,[b,c]],[[b,c],a]) истинно;
reverse([a,[b,c]],[[c,b],a]) ложно.

```

Если `X = [a,[b,c],d]`, то предикат `reverse(X,Y)` истинен при `Y = [d,[b,c],a]`. Предикат `reverse(L,S)` задается следующей процедурой.

```

reverse([],[]).
reverse([H|T],L) :- reverse(T,Z),append(Z,[H],L).

```

Обозначим предложения программы через `C1`, `C2` соответственно и проследим ход вычислений в ответ на запрос

`C3. reverse([a,b,c],L).`

Пусть `Y(C1,C2)` по-прежнему означает «унификация `C1` и `C2`».

`Y(C3,C1)` безуспешна по первому аргументу.

`Y(C3,C2)` успешна. `H := a`, `T := [b,c]`. Новый запрос `reverse([b,c],Z),append(Z,[a],L).`

Пусть `C4` есть очередная цель `reverse([b,c],Z)` в этом новом сложном запросе, состоящем из двух целей.

`Y(C4,C1)` безуспешна по первому аргументу.

`Y(C4,C2)` успешна. `H := b`, `T := [c]`.

Новый запрос:

```
reverse([c],Z1),append(Z1,[b],Z),append(Z,[a],L).
```

Пусть `C5` есть очередная цель `reverse([c],Z1)` в этом запросе.

`Y(C5,C1)` безуспешна по первому аргументу.

`Y(C5,C2)` успешна. `H := c`, `T := []`. Новый запрос:

```
reverse([],Z2),append(Z2,[c],Z1),append(Z1,[b],Z),append(Z,[a],L).
```

Пусть C6 есть очередная цель $\text{reverse}([], Z2)$ в этом запросе.
 $Y(C6, C1)$ успешна. $Z2 := []$. Новый запрос:

$\text{append}([], [c], Z1), \text{append}(Z1, [b], Z), \text{append}(Z, [a], L)$.

Обработка трех целей этого запроса последовательно дает:

$Z1 := [c], Z := [c, b], L := [c, b, a]$.

Транслятор в качестве ответа выдает $L = [c, b, a]$ и прекращает работу.
 Предикат $\text{reverse}(L, S)$ можно задать также следующей процедурой:

$\text{reverse1}(L1, L2) :- \text{rev}(L1, [], L2)$.

$\text{rev}([], L, L)$.

$\text{rev}([X|L], L2, L3) :- \text{rev}(L, [X|L2], L3)$.

9.7.7. Выравнивание списка

Предикат $\text{flatten}(S, L)$ истинен, если L есть список атомов списка S . Например:

- если $S = [a, b]$, то $L = [a, b]$;
- если $S = [a, [b, c], d]$, то $L = [a, b, c, d]$;
- если $S = [a, [b, [c, d]], [e]]$, то $L = [a, b, c, d, e]$.

Предикат $\text{flatten}(S, L)$ задается следующей процедурой.

$\text{flatten}([H|T], L) :- \text{flatten}(H, LH), \text{flatten}(T, LT), \text{append}(LH, LT, L)$.

$\text{flatten}([], [])$.

$\text{flatten}(X, [X])$.

9.7.8. Добавление элемента в начало списка

Предикат $\text{add}(X, L, L1)$ истинен, если список $L1$ получается добавлением элемента X в начало списка L . Например:

$\text{add}(a, [b, c], [a, b, c])$ истинно;

$\text{add}(a, [b, c], [b, a, c])$ ложно.

Если $X = a$, $Y = [[a], b, [d]]$, то предикат $\text{add}(X, Y, Z)$ истинен при $Z = [a, [a], b, [d]]$.

Предикат $\text{add}(X, L, L1)$ задается процедурой

$\text{add}(X, L, [X|L])$.

Добавление элемента в начало списка без его удвоения можно задать следующей процедурой.

$\text{add1}(X, L, L) :- \text{member}(X, L), !$.

$\text{add1}(X, L, [X|L])$.

В этой процедуре используется оператор отсечения. Если список L содержит элемент X , то цель member исполнится успешно, затем последует оператор отсечения, после исполнения которого ни add1 , ни member при бэктрекинге вычисляться уже не будут. Транслятор закончит вычисления и выдаст ответ. Если

список L элемента X не содержит, то цель `member` окажется безуспешной (ложной), согласование запроса с первым предложением программы будет тоже безуспешным, осуществится бэктрекинг. После согласования запроса со вторым предложением программы элемент X будет поставлен первым элементом в списке L и выдан ответ.

9.7.9. Удаление первого вхождения данного элемента из списка

Предикат `delete(X,L,M)` истинен, если список M есть результат удаления из списка L первого вхождения элемента X. Например:

```
delete(a,[s,f,a,h,a,g],[s,f,h,a,g]) истинно;
delete(a,[s,f,a,h,a,g],[s,f,a,h,g]) ложно.
```

Если $X = a$, $Y = [s,[d,s],a,h,a,d,a]$, то предикат `delete(X, Y,Z)` истинен при $Z = [s,[d,s],h,a,d,a]$. Если в списке L элемента X нет, то запрос `?- delete(X,L,M)` безуспешен.

Предикат `delete(X,L,M)` задается следующей процедурой:

```
delete(A,[A|B],B) :- !.
delete(A,[B|L],[B|M]) :- delete(A,L,M).
```

Оператор отсечения предотвращает альтернативу для предиката `delete`. Если оператор отсечения убрать, то с помощью бэктрекинга из списка L можно удалять последовательные вхождения элемента X.

9.7.10. Удаление всех вхождений данного элемента из списка

Предикат `delall(X,L,M)` истинен, если список M получается удалением из списка L всех вхождений элемента X. Например, если $X = a$, $Y = [a,s,d,a,f,a,h]$, то предикат `delall(X,Y,Z)` истинен при $Z = [s,d,f,h]$.

Предикат `delall(X,L,M)` задается следующей процедурой.

```
delall(_,[],[]).
delall(X,[X|L],M) :- !,delall(X,L,M).
delall(X,[Y|L1],[Y|L2]) :- delall(X,L1,L2).
```

Обозначим предложения программы через C1,C2,C3 соответственно; проследим ход вычислений и построим дерево вывода (рис. 9.10) в ответ на помещенный в корневой узел 0 дерева запрос

C4. `delall(a,[b,a,c,a,d],S)`.

Пусть $Y(C4,C1)$ означает «унификация C4 и C1».

Узел 1. $Y(C4,C1)$ безуспешна. Отход от узла 1 в узел 0.

Узел 2. $Y(C4,C2)$ безуспешна. Отход от узла 2 в узел 0.

Узел 3. $Y(C4,C3)$ успешна.

$X:=a$, $Y:=b$, $L1:=a,c,a,d$, $S:=b|L2$.

В узле 3 формируется запрос

$C5. \text{delall}(a, [a, c, a, d], L2).$

Запрос $C5$ получается заменой запроса $C4$ на правую часть предложения $C3$ программы при найденных значениях переменных.

Узел 4. $Y(C5, C1)$ безуспешна. Отход от узла 4 в узел 3.

Узел 5. $Y(C5, C2)$ успешна. $X:=a, L:=[c, a, d], L2:=M.$

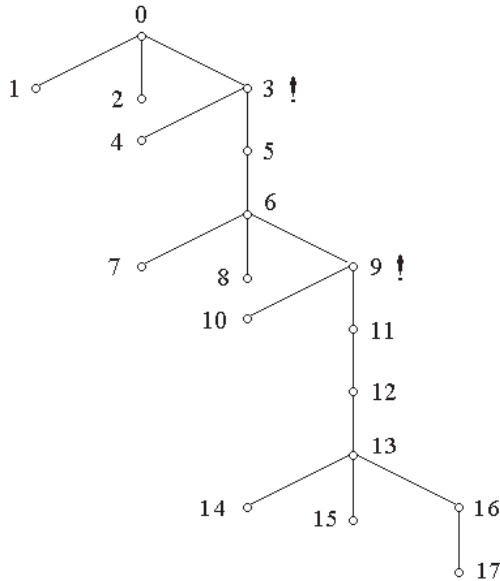


Рис. 9.10

В узле 5 формируется запрос

$C6. \text{!,delall}(a, [c, a, d], M).$

Запрос $C6$ получается заменой запроса $C5$ на правую часть предложения $C2$ программы при найденных значениях переменных.

Узел 6. В узле 6 выполняется оператор отсекаения; в узле 3 (он помечен знаком !) альтернатива отсекается. В узле 6 формируется запрос

$C7. \text{delall}(a, [c, a, d], M).$

Узел 7. $Y(C7, C1)$ безуспешна. Отход от узла 7 в узел 6.

Узел 8. $Y(C7, C2)$ безуспешна. Отход от узла 8 в узел 6.

Узел 9. $Y(C7, C3)$ успешна. $X:=a, Y:=c, L1:=[a, d], M := [c|L21]$. В узле 9 формируется запрос

$C8. \text{delall}(a, [a, d], L21).$

Запрос C8 получается заменой запроса C7 на правую часть предложения C3 программы при найденных значениях переменных.

Узел 10. $Y(C8, C1)$ безуспешна. Отход от узла 10 в узел 9.

Узел 11. $Y(C8, C2)$ успешна. $X:=a$, $L:=[d]$, $L21:=M1$. В узле 11 формируется запрос

C9. $!, delall(a, [d], M1)$.



Запрос C9 получается заменой запроса C8 на правую часть предложения C2 программы при найденных значениях переменных.

Узел 12. В узле 12 выполняется оператор отсеечения; в узле 11 (он помечен знаком !) альтернатива отсекается.

Узел 13. В узле 13 формируется запрос

C10. $delall(a, [d], M)$.

Узел 14. $Y(C10, C1)$ безуспешна. Отход от узла 14 в узел 13.

Узел 15. $Y(C10, C2)$ безуспешна. Отход от узла 15 в узел 13.

Узел 16. $Y(C10, C3)$ успешна. $X:=a$, $Y:=d$, $L1:=[]$, $M1 := [d|L22]$. В узле 16 формируется запрос

C11. $delall(a, [], L22)$.

Запрос C11 получается заменой запроса C10 на правую часть предложения C3 программы при найденных значениях переменных.

Узел 17. $Y(C11, C1)$ успешна. $L22:=[]$.

В узле 17 формируется запрос C12. □ Запрос C12 получается заменой запроса C11 на правую часть предложения C1 программы, которая пуста. Выведен пустой дизъюнкт. В узле 17 вычислительная ветвь закончилась успешно. Далее последуют следующие вычисления:

$M1 = [d|L22] = [d|[]] = [d]$; $L21 = M1 = [d]$;

$M = [c|L21] = [c, d]$; $L2 = M = [c, d]$; $S = [b|L2] = [b, c, d]$.

Выдается ответ $S = [b, c, d]$, и система предлагает бэктрекинг. В узлах 16, 13, 12, 11, 6, 5, 0 успешных альтернативных согласований с предложениями программы нет. В узлах 9 и 3 есть альтернатива успешного согласования помещенных в этих узлах запросов с третьим предложением программы. Но в узлах 9 и 3 альтернатива отсечена. Поэтому система возвращается в корневой узел 0 и сообщает об отсутствии альтернативных решений.

Заметим, что первая построенная успешная ветвь вычислений дает правильный ответ и без оператора отсеечения. Оператор отсеечения позволяет отследить только эту одну первую успешную ветвь в дереве вывода. Если оператор отсеечения убрать, то альтернативное согласование запроса с двумя последними предложениями программы при бэктрекинге даст неверные варианты ответов: некоторые входящие элемента X из списка L удалены не будут. Если процедура $delall$ будет составной частью другой программы, то при ее исполнении

может быть осуществлен отход в узлы исполнения целей процедуры `delall`. Тогда отсутствие оператора отсечения приведет к неверному результату для `delall` и потому к неверному ответу для всей программы.

9.7.11. Замена элемента в списке

Предикат `substitute(X,L,A,M)` истинен, если список `M` получается заменой в списке `L` всех вхождений элемента `X` на элемент `A`. Например, если `X = a`, `L = [b,a,c,a]`, `A = [b,d]`, то `M = [b,[b,d],c,[b,d]]`.

Предикат `substitute(X,L,A,M)` задается следующей процедурой:

```
substitute(_,[],_,[]).
substitute(X,[X|L],A,[A|M]) :- !,substitute(X,L,A,M).
substitute(X,[Y|L],A,[Y|M]) :- substitute(X,L,A,M).
```

Оператор отсечения в процедуре `substitute` работает аналогично тому, как он работает в процедуре `delall`.

9.7.12. Быть подписанием в списке

Предикат `sublist(X,Y)` истинен, если список `X` есть подсписок списка `Y` (в смысле вхождения слова `X` в слово `Y`). Например:

```
sublist([a,s],[g,d,a,s,h]) истинно;
sublist([a,s],[g,a,d,s,h]) ложно.
```

Предикат `sublist(X,Y)` задается следующей процедурой:

```
sublist([X|L],[X|M]) :- coincide(L,M),!.
sublist(L,[_|M]) :- sublist(L,M).
coincide([],_).
coincide([X|L],[X|M]) :- coincide(L,M).
```

Обозначим предложения программы через `C1,C2,C3,C4` соответственно. Проследим ход вычислений и построим дерево вывода (рис. 9.11) в ответ на помещенный в корневой узел 0 дерева запрос `C5. sublist([a,s],[g,d,a,s,h])`.

Пусть `U(C5,C1)` означает «унификация `C5` и `C1`».

Узел 1. `U(C5,C1)` безуспешна. Отход от узла 1 в узел 0.

Узел 2. `U(C5,C2)` успешна. В узле 2 формируется запрос

`C6. sublist([a,s],[d,a,s,h])`.

Узел 3. `U(C6,C1)` безуспешна. Отход от узла 3 в узел 2.

Узел 4. `U(C5,C2)` успешна. В узле 4 формируется запрос

`C7. sublist([a,s],[a,a,s,h])`.

Узел 5. `U(C7,C1)` успешна. В узле 5 формируется запрос

`C8. coincide([s],[a,s,h]),!`.

В запросе C8 обрабатывается цель

C9. `coincide([s],[a,s,h]).`

Узел 6. $Y(C9, C1)$ безуспешна. Отход от узла 6 в узел 5.

Узел 7. $Y(C9, C2)$ безуспешна. Отход от узла 7 в узел 5.

Узел 8. $Y(C9, C3)$ безуспешна. Отход от узла 8 в узел 5.

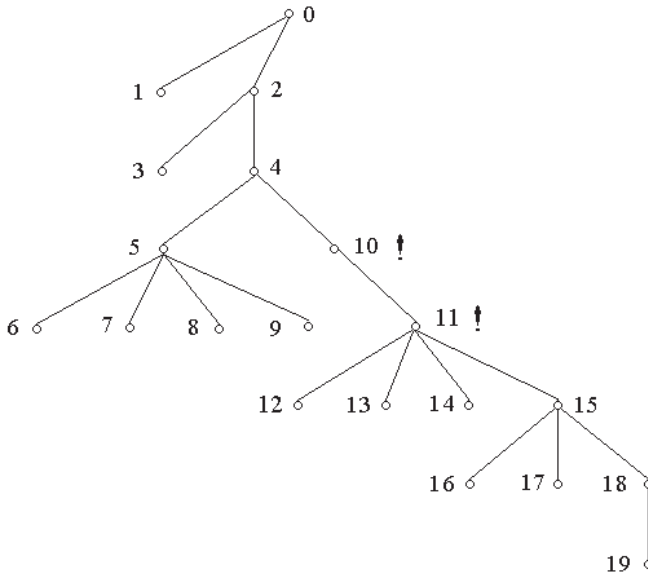


Рис. 9.11

Узел 9. $Y(C9, C4)$ безуспешна. Отход от узла 9 в узел 5.

Альтернатив по согласованию запроса C9 в узле 5 с предложениями программы больше нет. Осуществляется отход в узел 4, где для помещенного в этот узел запроса

C7 `sublist([a,s],[a,a,s,h])`

есть альтернатива согласования со вторым предложением программы.

Узел 10. $Y(C7, C2)$ успешна. В узле 10 формируется запрос

C10. `sublist([a,s],[a,s,h]).`

Узел 11. $Y(C10, C1)$ успешна. В узле 11 формируется запрос

C11. `coincide([s],[s,h]),!`.

Узел 12. $Y(C11, C1)$ безуспешна. Отход от узла 12 в узел 11.

Узел 13. $Y(C11, C2)$ безуспешна. Отход от узла 13 в узел 11.

Узел 14. $Y(C11, C3)$ безуспешна. Отход от узла 14 в узел 11.

Узел 15. $\mathcal{U}(C11, C4)$ успешна. В узле 15 формируется запрос
 $C12. \text{coincide}([], [h]), !.$

Запрос $C12$ получается заменой цели $\text{coincide}([s], [s, h])$ в запросе $C11$ на правую часть предложения $C4$ программы при найденных значениях переменных.

Узел 16. $\mathcal{U}(C12, C1)$ безуспешна. Отход от узла 16 в узел 15.

Узел 17. $\mathcal{U}(C12, C2)$ безуспешна. Отход от узла 17 в узел 15.

Узел 18. $\mathcal{U}(C12, C3)$ успешна. В узле 18 формируется запрос

$C13. !.$

Запрос $C13$ получается заменой цели $\text{coincide}([], [h])$ в запросе $C12$ на правую пустую часть предложения $C3$ программы. Далее исполняется оператор отсеечения. В узлах 10 и 11 альтернатива отсекается. В узле 19 формируется запрос $C14. \square$

В узле 19 вычисления закончились успешно. Система дает утвердительный ответ на исходный запрос и останавливается. При бэктрекинге получаем, что в узле 18 альтернатив нет. В узле 15 попытка $\mathcal{U}(C12, C4)$ безуспешна. Отход по дереву вывода продолжается. В узлах 11, 10 альтернатива отсечена. Осуществляется отход в узел 4, где запрос $\text{sublist}([a, s], [a, a, s, h])$ имеет две безуспешные попытки согласования с предложениями $C3$ и $C4$ программы. Следует отход в узел 2, где запрос имеет две безуспешные попытки согласования с предложениями $C3$ и $C4$ программы. Система отходит в корневой узел 0, и после двух безуспешных попыток согласования помещенного в этот узел запроса с предложениями $C3$ и $C4$ программа при отсутствии других решений прекращает работу.

9.7.13. Включение множеств

Предикат $\text{subset}(X, Y)$ истинен, если множество X есть подмножество множества Y . Например:

$\text{subset}([a, s, d], [h, d, s, h, a])$ истинно;
 $\text{subset}([a, s, d], [h, s, h, a])$ ложно;
 $\text{subset}([a, [s, d]], [a, [d, s], s, h, a])$ ложно.

Предикат $\text{subset}(X, Y)$ задается следующей процедурой:

$\text{subset}([], Y).$
 $\text{subset}([A|X], Y) :- \text{member}(A, Y), \text{subset}(X, Y).$

Процедура $\text{subset}(X, Y)$ работает правильно, если обе ее переменные на момент выполнения процедуры замещены списками без переменных. Процедуру $\text{subset}(X, Y)$ нельзя использовать для порождения всех подмножеств множества Y . Для этого следует воспользоваться следующей процедурой.

$\text{subset1}([], []).$
 $\text{subset1}([H|S], [H|T]) :- \text{subset1}(S, T).$
 $\text{subset1}(S, [H|T]) :- \text{subset1}(S, T).$

Процедура `subset1(X,Y)` работает правильно, если на момент ее выполнения X есть переменная, а Y заменено списком без переменных.

9.7.14. Равенство множеств

Предикат `equalset(X,Y)` истинен, если множества X и Y совпадают. Множества X и Y задаются списками атомов.

Предикат `equalset(X,Y)` задается процедурой

`equalset(X,Y) :- subset(X,Y),subset(Y,X).`

Процедуру `equalset(X,Y)` нельзя использовать для порождения множества X . Для этого служит процедура

`equalset1(X,Y) :- subset1(X,Y),subset1(Y,X).`

9.7.15. Объединение множеств

Предикат `unionset(X,Y,Z)` истинен, если множество Z есть объединение множеств X и Y . Например, если $X = [a,s]$, $Y = [f, h,j]$, то $Z = [a,s,f,h,j]$. Предикат `unionset(X,Y,Z)` задается следующей процедурой:

`unionset([X|R],Y,Z) :- member(X,Y),!,unionset(R,Y,Z).`

`unionset([X|R],Y,[X|Z]) :- unionset(R,Y,Z).`

`unionset([],X,X).`

Обозначим предложения программы через $C1$, $C2$, $C3$ соответственно. Проследим ход вычислений и построим дерево вывода (рис. 9.12) в ответ на помещенный в корневой узел 0 дерева запрос

$C4. \text{unionset}([c,b,a],[d,a,b,f],U).$

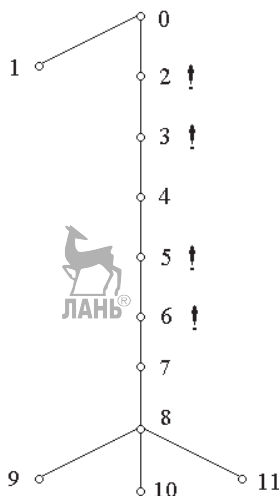


Рис. 9.12

Пусть $U(C4, C1)$ означает «унификация $C4$ и $C1$ ».

Узел 1. $U(C4, C1)$ успешна. В узле 1 формируется запрос

$C5. \text{member}(c, [d, a, b, f]), !, \text{unionset}([c, b, a], [d, a, b, f], U).$

Цель $\text{member}(c, [d, a, b, f])$ ложна. Отход от узла 1 в узел 0.

Узел 2. $U(C4, C2)$ успешна. $X:=c, R:=[b, a], U:=[c|Z]$. В узле 2 формируется запрос

$C6. \text{unionset}([b, a], [d, a, b, f], Z).$

Узел 3. $U(C6, C1)$ успешна. В узле 3 формируется запрос

$C7. \text{member}(b, [d, a, b, f]), !, \text{unionset}([a], [d, a, b, f], Z).$

Узел 4. Цель $\text{member}(b, [d, a, b, f])$ успешна. В узле 4 формируется запрос

$C8. !, \text{unionset}([a], [d, a, b, f], Z).$

Узел 5. Цель $!$ успешна. В узле 5 формируется запрос

$C9. \text{unionset}([a], [d, a, b, f], Z).$

В узлах 2 и 3 (они помечены знаком $!$) альтернатива для помещенных в них целей отсекается.

Узел 6. $U(C9, C1)$ успешна. В узле 6 формируется запрос

$C10. \text{member}(a, [d, a, b, f]), !, \text{unionset}([], [d, a, b, f], Z).$

Узел 7. Цель $\text{member}(a, [d, a, b, f])$ успешна. В узле 7 формируется запрос

$C11. !, \text{unionset}([], [d, a, b, f], Z).$

Узел 8. Цель $!$ успешна. В узле 8 формируется запрос

$C12. \text{unionset}([], [d, a, b, f], Z).$

В узлах 5 и 6 (они помечены знаком $!$) альтернатива для помещенных в них целей отсекается.

Узел 9. $U(C12, C1)$ безуспешна. Отход от узла 9 в узел 8.

Узел 10. $U(C12, C2)$ безуспешна. Отход от узла 10 в узел 8. Узел 11. $U(C12, C3)$ успешна. $Z:=[d, a, b, f]$. В узле 11 формируется запрос

$C13. \square$

В узле 11 вычислительная ветвь закончилась успешно. Далее будет вычислено $U = [c|Z] = [c, d, a, b, f]$. Система выдаст ответ $U = [c, d, a, b, f]$ и предложит бэктрекинг для поиска альтернативных решений. От узла 11 осуществится отход в узел 8, где альтернатив согласования помещенного в этом узле запроса с предложениями программы нет. В узлах 7 и 4 альтернатив нет. В узлах 6, 5, 3, 2 альтернатива (поиск альтернативных решений) отсечена. В узле 0 альтернатива согласования помещенного в него запроса с третьим предложением программы безуспешна. Система указывает на отсутствие других решений и прекращает работу.

9.7.16. Пересечение множеств

Предикат `intersect(X,Y,Z)` истинен, если множество Z есть пересечение множеств X и Y . Например, если $X = [a,b,c,d]$, $Y = [s,c,b,f]$, то $Z = [b,c]$.

Предикат `intersect(X,Y,Z)` задается следующей процедурой:

```
intersect([],X,[]).
intersect([X|R],Y,[X|Z]) :- member(X,Y),!,intersect(R,Y,Z).
intersect([X|R],Y,Z) :- intersect(R,Y,Z).
```

Использование оператора отсеечения в процедуре `intersect(X,Y,Z)` аналогично его использованию в процедуре `unionset(X,Y,Z)`.

9.7.17. Разность множеств

Предикат `difset(X,Y,Z)` истинен, если множество Z есть разность множеств X и Y . Например, если $X = [a,d,g,f]$, $Y = [u,a,p,g,e]$, то $Z = [d,f]$.

Предикат `difset(X,Y,Z)` задается следующей процедурой:

```
difset(X,Y,T) :- dif1(X,Y,X,T).
dif1([R|X],Y,[R|Z],T) :- not(member(R,Y)), append(Z,[R],Z1),dif1(X,Y,Z1,T).
dif1([R|X],Y,[R|Z],T) :- member(R,Y),dif1(X,Y,Z,T).
dif1([],Y,Z,Z).
```

9.7.18. Декартово произведение множеств

Предикат `dek2(X,Y,Z)` истинен, если множество Z есть декартово произведение множеств X и Y .

Предикат `dek2(X,Y,Z)` задается процедурой:

```
dek2(X,Y,Z) :- findall([A,B], (member(A,X),member(B,Y)),Z).
```

Предикат `dek(S,L)` истинен, если список L есть декартово произведение множеств списка S . Например, если

```
S = [[a,b],[c,d], [e,f]], то
L = [[a,c,e],[a,c,f],[a,d,e],[a,d,f],[b,c,e],[b,c,f], [b,d,e],[b,d,f]].
```

Предикат `dek(S,L)` задается процедурой:

```
dek(S,L) :- dek1(S,L).
dek1([H1,H2|T],L) :- dek2(H1,H2,L1),dek3(L1,T,L).
dek1([H|[]],[H|[]]).
dek1([],[]).
dek3(L1,[],L1).
dek3(L1,[H|T],L) :- dek2(L1,H,L2),append1(L2,L3),dek3(L3,T,L).
append1(L2,L3) :- S = [],append2(L2,L3,S).
append2([H|T],L3,S) :- append(H,T,L4),append(S,[L4],S1),
    append2(T1,L3,S1).
append2([],S1,S1).
```

9.7.19. Множество всех подмножеств данного множества

Предикат `powerset(X,S)` истинен, если S есть множество всех подмножеств множества X .

Предикат `powerset(X,S)` задается следующей процедурой:

```
powerset(X,S) :- findall(Y,subset1(Y,X),S).
```

9.7.20. Удаление всех повторов элементов в списке

Предикат `delrepeate(S,SF)` истинен, если список SF получается удалением всех повторов элементов в списке S . Например, если $S = [a,d,s,a,g,d,g,a]$, то $SF = [s,d,g,a]$.

Предикат `delrepeate(S,SF)` задается следующей процедурой:

```
delrepeate(S,SF) :- S1=[],delrep(S,S1,SF).
delrep([H|T],S1,SF) :- member(H,T),!,delrep(T,S1,SF).
delrep([H|T],S1,SF) :- not(member(H,T)), append(S1,[H],S2),delrep(T,S2,SF).
delrep([],S2,S2).
```

9.7.21. Принадлежность множества списку подмножеств

Множество некоторых подмножеств задается как список S списков атомов. Предикат `member_set(L,S)` истинен, если множество L является элементом во множестве S . Например:

```
member_set([a,d,s],[[a,b],[d,a,s]]) истинно;
member_set([a,s],[[a,b],[d,a,s]]) ложно.
```

Переменные L и S к моменту вычисления предиката должны быть замещены константами.

Предикат `member_set(L,S)` задается следующей процедурой:

```
member_set(L,[Y1|Y]) :- subset(L,Y1),subset(Y1,L),!.
member_set(L,[Y1|Y]) :- member_set(L,Y).
```

9.7.22. Удаление всех повторов подмножеств в данном списке множеств

S есть множество некоторых подмножеств, заданное как список списков атомов. Предикат `del_rep_sets(S,SF)` истинен, если множество SF получается удалением всех повторов элементов в S . Например, если

```
S = [[a,s],[f,h,a],[s,a,s], [s,a,a], [a]], то SF = [f,h,a],[s,a,a],[a]].
```

Удаление повторов в составляющих множествах не производится.

Предикат `del_rep_sets(S,SF)` задается следующей процедурой:

```
del_rep_sets(S,SF) :- S1=[],del_set(S,S1,SF).
del_set([H|T],S1,SF) :- member_set(H,T),del_set(T,S1,SF).
del_set([H|T],S1,SF) :- not(member_set(H,T)),append(S1,[H],S2),
    del_set(T,S2,SF).
del_set([],S2,S2).
```

9.7.23. Удаление повторов атомов в списке списков атомов

Предикат `delrep_el(S,SM)` истинен, если список `SM` получается удалением повторов атомов в списке `S` списков атомов. Например:

если `S = [[a,s,a,d,s,f],[s,k],[f,d,f,h,f]]`,
то `SM = [[a,d,s,f], [s,k],[d,h,f]]`.

Предикат `delrep_el(S,SF)` задается следующей процедурой:

```
delrep_el(S,SM) :- L=[],delrpl(S,L,S1),reverse(S1,SM).
delrpl([H|T],L,S1) :- delrepeate(H,H1),delrpl(T,[H1|L],S1).
delrpl([],S1,S1).
```

9.7.24. Последовательное порождение нумерованных атомов

Процедура `genatom(Roote,Atom)` образует атом `Atom`, в котором первым символом является атом `Roote`, а вторым – индекс. Например, при `Roote = q` процедура `genatom` при последовательном ее исполнении даст значения `Atom`, последовательно равные `q0`, `q1`, `q2`, ...

Процедура `genatom(Roote,Atom)` задается следующей программой:

```
genatom(Roote,Atom) :- get_number(Roote,Number),
    name(Roote,Name1),name(Number,Name2),
    append(Name1,Name2,Name),name(Atom,Name).
get_number(Roote,Number) :- retract(current_number(Roote,Number1)),!,
    Number is Number1+1, asserta(current_number(Roote,Number)).
get_number(Roote,0) :- asserta(current_number(Roote,0)).
```

9.7.25. Программа построения сокращенной ДНФ по конъюнктивной нормальной форме

Перемножаются две скобки КНФ. Результат последовательно умножается на другие скобки КНФ, причем при каждом таком перемножении из результата удаляются слагаемые (списки), содержащие переменную и ее отрицание, производится поглощение в каждом слагаемом (списке) и между всеми слагаемыми (списками).

Предикат `del_neg(L1,L)` удаляет из ДНФ слагаемые, содержащие переменную и ее отрицание.

Предикат `have_neg(X)` истинен, если дизъюнктивное слагаемое (список `X`) имеет переменную и ее отрицание.

Предикат `d(S,L)` по КНФ `S`, заданной списком, выдает сокращенную ДНФ `L`. Например:

если `S=[[x,-y,z,t],[x,-y,-z,t],[x,-y,-z,-t],[-x,y,z,-t],[-x,y,-z,-t]]`,
то `L=[-y,-t],[t,-z,-x],[t,-z,y],[x,y],[x,-t],[-y,-x]`.

В привычных обозначениях:

$$S = (x \vee \bar{y} \vee z \vee t)(x \vee \bar{y} \vee \bar{z} \vee t)(x \vee \bar{y} \vee z \vee \bar{t})(\bar{x} \vee y \vee z \vee t)(\bar{x} \vee y \vee \bar{z} \vee \bar{t}) = \\ \bar{y}t \vee t\bar{z}x \vee tzy \vee xy \vee x\bar{t} \vee \bar{y}\bar{x}.$$

d(S,L) :-

S=[[x,-y,z,t],[x,-y,-z,t],[x,-y,-z,-t],[x,y,z,-t],[-x,y,-z,-t]], dek_absorb(S,L).

dek_absorb(S,L) :- dek1_absorb(S,L1), del_neg(L1,L).

dek1_absorb([H1,H2|T],L) :-

dek2(H1,H2,L1N), del_neg(L1N,L11),
delrepeate(L11,L13), delrep_el(L13,L12),
absorb_set(L12,L14), dek3_absorb(L14,T,L).

dek1_absorb([H|[]], [H|[]]).

dek1_absorb([], []).

dek3_absorb(L1,[H|T],L) :- dek2(L1,H,L2),

append1(L2,L31N), del_neg(L31N,L31),
delrepeate(L31,L33), delrep_el(L33,L32),
absorb_set(L32,L3), dek3_absorb(L3,T,L).

dek3_absorb(L1,[],L1). append1(L2,L3) :- append2(L2,L3,[]).

append2([H|T]|T1,L3,S) :- append(H,T,L4),

append(S,[L4],S1), append2(T1,L3,S1).

append2([],S,S).

dek2(X,Y,Z) :- findall([A,B],(member(A,X),member(B,Y)),Z).

delrep_el(S,SM) :- delrpl(S,[],S1), reverse(S1,SM).

delrpl([H|T],L,S1) :- delrepeate(H,H1), delrpl(T,[H1|L],S1).

delrpl([],S,S).

absorb_set(S,L) :- absorb(S,S,L).

absorb([H|T],[H1|T1],L) :- absorb1(T1,H1,T2),

append(T2,[H1],T3), absorb(T,T3,L).

absorb([],L,L).

absorb1(S,A,L) :- findall(M,(member(M,S), not(subset(A,M))),L).

delrepeate(S,SF) :- S1=[], delrep(S,S1,SF).

delrep([H|T],S1,SF) :- member(H,T),!, delrep(T,S1,SF).

delrep([H|T],S1,SF) :- not(member(H,T)), append(S1,[H],S2),
delrep(T,S2,SF). delrep([],S2,S2).

del_neg(L1,L2) :- L1=[Head|L3],

ifthenelse(have_neg(Head),del_neg(L3,L2),
(del_neg(L3,L4),L2=[Head|L4])).

del_neg([], []).

have_neg([]) :- fail.

```

have_neg(X) :- X=[Head|Y],string_to_atom (H,Head),
  ifthenelse(sub_string(H,0,1,After1,$-$),
    (string_length(H,Len),Ln is Len-1,
      sub_string(H,1,Ln,After2,El),
      string_to_atom (El,Elem),member(Elem,Y)),
    (append($-$,H,Res1),string_to_atom(Res1,Res),
      member(Res,Y))).
have_neg([Head|Y]) :- have_neg(Y).

```

```

ifthenelse(P,Q,R) :- P *-> Q ; R.
subset([],Y).
subset([A|X],Y) :- member(A,Y),subset(X,Y).

```

9.7.26. Программа построения сокращенной ДНФ по конъюнктивной нормальной форме без отрицаний

Предикат d(S,L) по КНФ S, заданной списком без отрицаний, выдает сокращенную ДНФ L. Например:

если $S = [[1,2,3,5,6],[2,1,3],[3,1,2,4,6],[4,3,5,7],[5,1,4,6],[6,1,3,5,7],[7,4,6]]$,
то $L = [[2,6,7],[5,3,7],[3,4],[3,6],[1,4],[5,1,6],[1,7],[2,6,4],[2,4,7],[2,5,7],[2,5,4],[2,5,6]]$.

В привычных обозначениях:

$$S = (1 \vee 2 \vee 3 \vee 5 \vee 6)(2 \vee 1 \vee 3)(3 \vee 1 \vee 2 \vee 4 \vee 6)(4 \vee 3 \vee 5 \vee 7)(5 \vee 1 \vee 4 \vee 6). \\ (6 \vee 1 \vee 3 \vee 5 \vee 7)(7 \vee 4 \vee 6) = 267 \vee 537 \vee 34 \vee 36 \vee 14 \vee 516 \vee 17 \vee 264 \vee \\ 247 \vee 257 \vee 254 \vee 256.$$

```

d(S,L) :- S=[[1,2,3,5,6],[2,1,3],[3,1,2,4,6],[4,3,5,7],[5,1,4,6],[6,1,3,5,7],[7,4,6]],
  dek_absorb(S,L).

```

```

dek_absorb(S,L) :- dek1_absorb(S,L).
dek1_absorb([H1,H2|T],L) :- dek2(H1,H2,L11),
  delrepeate(L11,L13),delrep_el(L13,L12),
  absorb_set(L12,L14),dek3_absorb(L14,T,L).
dek1_absorb([H|[]],[H|[]]). dek1_absorb([],[]).

```

```

dek3_absorb(L1,[H|T],L) :- dek2(L1,H,L2),append1(L2,L31),
  delrepeate(L31,L33),delrep_el(L33,L32),
  absorb_set(L32,L3),dek3_absorb(L3,T,L).
dek3_absorb(L1,[],L1).

```

```

append1(L2,L3) :- append2(L2,L3,[]).
append2([H|T]|T1,L3,S) :- append(H,T,L4),
  append(S,[L4],S1), append2(T1,L3,S1).
append2([],S,S).

```

```

dek2(X,Y,Z) :- findall([A,B],(member(A,X),member(B,Y)),Z).

```

```

delrep_el(S,SM) :- delrpl(S,[],S1),reverse(S1,SM).

```

```

delrpl([H|T],L,S1) :- delrepeate(H,H1),delrpl(T,[H1|L],S1).
delrpl([],S,S).

absorb_set(S,L) :- absorb(S,S,L).
absorb([H|T],[H1|T1],L) :-
    absorb1(T1,H1,T2),append(T2,[H1],T3),absorb(T,T3,L).
absorb([],L,L).
absorb1(S,A,L) :- findall(M,(member(M,S), not(subset(A,M))),L).

delrepeate(S,SF) :- S1=[],delrep(S,S1,SF).
delrep([H|T],S1,SF) :- member(H,T),!,delrep(T,S1,SF).
delrep([H|T],S1,SF) :- not(member(H,T)),append(S1,[H],S2),delrep(T,S2,SF).
delrep([],S2,S2).

subset([],Y).
subset([A|X],Y) :- member(A,Y),subset(X,Y).

```

9.8. Некоторые встроенные предикаты Пролога

В последующих встроенных предикатах Пролога переменные сопровождаются знаками + (плюс), - (минус), ? (вопросительный знак). Эти знаки означают следующее:

- + – переменная на момент исполнения предиката должна быть замещена (т. е. является входной переменной);
- - – переменная на момент исполнения предиката не должна быть замещена (т. е. является выходной переменной);
- ? – переменная на момент исполнения предиката может быть и замещенной, и незамещенной (т. е. быть и входной, и выходной).

9.8.1. Средства управления

- **P,Q** – конъюнкция целей P и Q.
- **P ; Q** – дизъюнкция целей P и Q.
- **!** – символ оператора отсечения.
- **abort** – прекращает выполнение программы.
- **break** – приостанавливает выполнение программы; вычисления возобновляются при наборе `end_of_file`.
- **call(+P)** – успешно, если вызываемая цель P успешна, и безуспешно в противном случае.
- **P -> Q** – если цель P успешна, то вычисляется цель Q. Если цель P безуспешна, то цель Q не вычисляется, а цель `ifthen(P,Q)` считается успешной.
- **P *-> Q ; R** – если цель P успешна, то вычисляется цель Q. Если цель P безуспешна, то вычисляется цель R.
- **not(+P)** – ложно, если цель P выполняема; истинно, если нет. Другие записи этого предиката: `not P`, `\+ P`.

- **repeat** – всегда успешно. При бэктрекинге вызывает повторное выполнение последующих за repeat целей.
- **true** – всегда успешно (т. е. истинно).

9.8.2. Классификация термов

- **atom(?X)** – X есть атом.
- **atomic(?X)** – X есть одно из: атом, целое число, вещественное число, стринг.
- **float(?X)** – X есть вещественное число.
- **integer(?X)** – X есть целое число.
- **number(?X)** – X есть целое или вещественное число. **nonvar(X)** переменная X замещена.
- **var(?X)** – переменная X не замещена.



9.8.3. Унификация термов

- **T1 = T2** – истинно, если термы T1 и T2 унифицируемы.
- **T1 \= T2** – истинно, если термы T1 и T2 не унифицируемы.

9.8.4. Сравнение термов

Пролог упорядочивает термы в стандартный порядок следующим образом. Пусть T1 и T2 есть некоторые термы. Любая переменная (все переменные имеют одинаковый приоритет) минимальна. Целые и вещественные числа упорядочены естественным образом. Атомы и стринги упорядочиваются лексикографически; знак \$ при сравнении атома и стринга не учитывается. Предикатные структуры упорядочены сначала по арности (т. е. местности предиката), затем по главному функтору и далее рекурсивно по аргументам в порядке их следования. Термы эквивалентны, если они равны относительно введенного стандартного порядка.

- **T1 == T2** – истинно, если термы T1 и T2 эквивалентны.
- **T1 \== T2** – истинно, если термы T1 и T2 не эквивалентны.
- **T1 @< T2** – истинно, если терм T1 меньше терма T2 в стандартном порядке.
- **T1 @=< T2** – истинно, если терм T1 меньше или равен терму T2 в стандартном порядке.
- **T1 @> T2** – истинно, если терм T1 больше терма T2 в стандартном порядке.
- **T1 @=> T2** – истинно, если терм T1 больше или равен терму T2 в стандартном порядке.
- **compare(-C,+T1,+T2)** – сравнивает термы T1 и T2 побуквенно и возвращает с переменной C: знак = при идентичности T1 и T2; знак < при T1<T2 в стандартном порядке; знак > при T1>T2 в стандартном порядке. Переменная C на момент исполнения предиката должна быть незамещенной.



- **sort(+L1,-L2)** – сортирует список L1 в соответствии со стандартным порядком, удаляет повторы элементов и отсортированный список возвращает с L2.

9.8.5. Арифметические функции

Арифметические термы (или выражения) строятся из атомов и переменных с помощью нижеопределенных арифметических операций обычным образом. Допускаются префиксная и инфиксная записи арифметических выражений. Арифметические термы без переменных являются константами.

Пусть X и Y есть операнды арифметических выражений.

Пролог имеет следующие встроенные арифметические функции.

- **X + Y** – сложение.
- **X - Y** – вычитание.
- **X * Y** – умножение.
- **X / Y** – деление.
- **X // Y** – целое деление.
- **X ^ Y** – X в степени Y.
- **-X** – минус X.
- **X ∧ Y** – побитовая конъюнкция (только для целых чисел).
- **X ∨ Y** – побитовая дизъюнкция (только для целых чисел).
- **\(X)** – побитовое отрицание (только для целых чисел).
- **X << Y** – побитовый сдвиг X влево на Y позиций (только для целых чисел).
- **X >> Y** – побитовый сдвиг X вправо на Y позиций (только для целых чисел).
- **X mod Y** – остаток от деления X на Y (только для целых чисел).
- **abs(X)** – абсолютная величина X.
- **acos(X)** – arccos(X).
- **asin(X)** – arcsin(X).
- **atan(X)** – arctan(X).
- **cos(X)** – cos(X).
- **exp(X)** – e в степени X.
- **log(X)** – логарифм натуральный X.
- **log10(X)** – логарифм десятичный X.
- **sin(X)** – sin(X).
- **sqrt(X)** – корень квадратный X.
- **tan(X)** – tan(X).
- **round(X,N)** – X округляется до N десятичных знаков; $0 < N < 15$.

9.8.6. Арифметические предикаты

Пусть E1 и E2 есть арифметические выражения. Пролог имеет следующие встроенные арифметические предикаты. К моменту вычисления истинностного значения предиката переменные в арифметических выражениях E, E1 и E2 должны быть замещены константами.

- $E1 > E2$ – истинно, если $E1$ больше $E2$.
- $E1 < E2$ – истинно, если $E1$ меньше $E2$.
- $E1 \geq E2$ – истинно, если $E1$ больше или равно $E2$.
- $E1 \leq E2$ – истинно, если $E1$ меньше или равно $E2$.
- $E1 == E2$ – истинно, если $E1$ равно $E2$.
- $E1 \neq E2$ – истинно, если $E1$ не равно $E2$.
- $X \text{ is } E$ – незамещенной переменной X присваивается значение E

9.8.7. Обработка термов

- $\text{arg}(+N, +T, -X)$ – истинно, если переменной X присваивается аргумент N терма T . Аргументы исчисляются слева направо, начиная с 1.
- $\text{functor}(?T, ?F, ?N)$ – успешно, если F есть главный функтор терма T , а N есть местность F .
- $\text{length}(+L, -N)$ – успешно, если N есть число элементов в списке элементов L .
- $\text{name}(?A, ?L)$ – успешно, если L есть список кодов ASCII символов атома A .
- $T =.. L$ – истинно, если L есть список, содержащий главный функтор в терме T , за которым следуют его аргументы.

9.8.8. Работа со строками

- $\text{atom\_string}(?A, ?S)$ – обращает атом A в строку S и строку S в атом A .
- $\text{string\_to\_atom}(?S, ?A)$ – обращает строку S в атом A и атом A в строку S .
- $\text{append}(+S1, +S2, -R)$ – возвращает результат R конкатенации (соединения) стрингов или атомов $S1$ и $S2$. Результат есть всегда строка.
- $\text{append}(+[S1, +S2, ...], -R)$ – возвращает результат R конкатенации (соединения) стрингов или атомов $S1, S2...$. Результат есть всегда строка.
- $\text{read\_string}(+L, -S)$ – возвращает строку S длины не более L , прочитывая его с текущего входного устройства (стандартного или файла).
- $\text{read\_string}(+H, +L, -S)$ – возвращает строку S длины не более L , прочитывая его как содержимое файла, которому приписано при открытии число H .
- $\text{string\_length}(+S, ?L)$ – возвращает длину L строки S .
- $\text{string\_to\_list}(?S, ?L)$ – обращает строку S в список L -кодов ASCII-символов в S , а также список L -кодов ASCII-символов в строку S .
- $\text{sub\_string}(+String, ?Start, ?Length, ?After, ?Sub)$ – успешно, если $String$ есть строка, $Start$ есть позиция начала подстроки, $Length$ есть длина подстроки, $After$ есть количество символов строки после подстроки, Sub есть разыскиваемый подстрока.

9.8.9. Составление списков

- $\text{bagof}(?X, +P, L)$ – успешно, если L есть список тех X , для которых цель $P(X)$ успешна. Объекты X и P должны иметь общие переменные. Список L не упорядочивается. Повторы элементов в L не устраняются. Если объектов X , удовлетворяющих P , нет, то запрос $\text{bagof}(X, P, L)$ безуспешен. Если P имеет,

кроме X , и другие переменные, например Y, Z , то список L будет вычисляться для каждого возможных элементов Y и Z . Чтобы список L содержал все значения X , для которых $P(X, Y, Z)$ истинно, на переменные Y, Z следует наложить квантор существования:

$\text{bagof}(X, Y^{\wedge}, Z^{\wedge} P(X, Y, Z), L).$

Можно писать также

$\text{bagof}(X, (Y, Z)^{\wedge} P(X, Y, Z), L).$

- **findall**($?X, +P, -L$) – успешно, если L есть список всех тех X , для которых цель P успешна. Список L не упорядочивается, повторы элементов в L не устраняются. Если объектов X , удовлетворяющих P , нет, то запрос $\text{findall}(X, P, L)$ истинен, если список L пуст. Цель P , кроме X , может содержать и другие переменные. При этом предикат действует так, как будто бы на эти переменные навешен квантор существования.
- **setof**($?X, +P, -L$) – действует так же, как и $\text{bagof}(X, P, L)$, за исключением того, что список L упорядочивается; повторы элементов из L устраняются. Если объектов X , удовлетворяющих P , нет, то запрос $\text{setof}(X, P, L)$ безуспешен.

9.8.10. Взаимодействие с базой данных

Приведем некоторые встроенные процедуры, позволяющие программисту манипуляции с базой данных.

- **abolish**($+Name/Arity$) – устраняет из базы данных все предложения с указанным ключом.
- **assert**($+C$) – добавляет к базе данных предложение C в конец цепи предложений с указанным в C ключом. В качестве C не могут выступать термы, отличные от предложений Пролога. Если в БД предложения с ключом, указанным в C , нет, то предложение C добавляется в конец программы.
- **asserta**($+C$) – добавляет к БД предложение C в начало цепи предложений с указанным в C ключом. В качестве C не могут выступать термы, отличные от предложений Пролога. Если в БД предложения с ключом, указанным в C , нет, то предложение C добавляется в конец программы.
- **assertz**($+C$) – добавляет к БД предложение C в конец цепи предложений с указанным в C ключом. В качестве C не могут выступать термы, отличные от предложений Пролога. Если в БД предложения с ключом, указанным в C , нет, то предложение C добавляется в конец программы.
- **clause**($+H, -B$) – унифицирует H с головой предложения и унифицирует B с телом предложения. Переменная H должна быть замещена.
- **current_predicate**($?P$) – возвращает имя и арность предикатов в текущем мире кодов.
- **erase**($+Ref$) – удаляет из БД терм под ссылочным числом Ref . **eraseall**($+Key$) удаляет из БД все термы с ключом Key .

- **instance(+Ref, -T)** – возвращает терм T, размещенный в БД под ссылочным числом Ref.
- **listing** – просмотр программы в рабочей области.
- **listing(+N)** – просмотр всех предикатов программы с именем N.
- **recorda(+Key, ?T, -Ref)** – добавляет к БД терм T в начало цепи термов с указанным в T ключом. Возвращает ссылочное число Ref, под которым терм T размещен в базе данных. В качестве T могут выступать термы, отличные от предложений Пролога.
- **recorded(+Key, ?T, -Ref)** – выдает из БД терм T, размещенный под заданным ключом Key.
- **recordz(+Key, ?T, -Ref)** – добавляет к БД терм T в конец цепи термов с указанным в T ключом. Возвращает ссылочное число Ref, под которым терм T размещен в базе данных. В качестве T могут выступать термы, отличные от предложений Пролога.
- **retract(+C)** – убирает из БД первое предложение, согласуемое с предложением C.



9.8.11. Стандартные ввод и вывод

- **current_op(?Prec, ?Assoc, ?Op)** – возвращает при бэктрекинге определения текущих операторов БД.
- **get(?Char)** – читает во входном потоке данных первый печатаемый символ, пропуская все непечатаемые, и присваивает переменной Char код ASCII этого символа.
- **get0(?Char)** – читает во входном потоке данных очередной символ и присваивает переменной Char код ASCII этого символа.
- **nl** – переход на другую строку.
- **op(+Prec, +Assoc, +Op)** – определяет приоритет Prec, ассоциативность Assoc оператора Op.

В SWI-Prolog приняты следующие значения переменных предиката op (табл. 9.1).

Таблица 9.1

Приоритет	Ассоциативность	Оператор
1200	xfx	-->, :-
1200	fx	:-, ?-
1150	fx	dynamic, discontinuous, initialization, meta_predicate, module_transparent, multifile, public, thread_local, thread_initialization, volatile
1100	xfy	;;,
1050	xfy	->, *->
1000	xfy	,
990	xfx	:=
900	fy	\+
700	xfx	<, =, =.., =@=, \=@=, =:=, =<, ==, =\=, >, >=, @<, @=<, @>, @>=, \=, \==, as, is, >:<, :<



Таблица 9.1 (окончание)

Приоритет	Ассоциативность	Оператор
600	xfy	:
500	yfx	+, -, $\wedge$, $\vee$, xor
500	fx	?
400	yfx	*, /, //, div, rdiv, <<, >>, mod, rem
200	xfx	**
200	xfy	$\wedge$
200	fy	+, -, \
100	yfx	.
1	fx	\$

- **put(+Char)** – выдает символ, присвоенный переменной Char. read(-T) читает терм с текущего входного устройства.
- **skip(Char)** – пропускает буквы текущего входного потока, пока не встретится буква Char.
- **tab(+N)** – выполняет N пробелов.
- **write(+T)** – печатает терм T на текущее выходное устройство.
- **writeln(+T)** – аналогичен write(T); имя атома или функтора заключает в кавычки (котирует).

9.8.12. Доступ к файлам

- **open(-H,+F,+Access)** – открывает уже существующий файл с именем F и с приписанным ему числом H; Аргумент Access может принимать одно из следующих значений, каждое из которых определяет характер доступа к файлу:
 - r – файл открыт для чтения,
 - w – файл открыт для записи,
 - a – файл открыт для присоединения новой записи,
 - rw – файл открыт для чтения или записи,
 - ra – файл открыт для чтения или присоединения новой записи.
- **close(+H)** – закрывает файл с именем, которому приписано число H.

9.8.13. Стандартный доступ к файлам в Прологе

- **see(+F)** – открывает файл с именем F для чтения в качестве текущего входного устройства.
- **seeing(?F)** – выдает имя файла, открытого для чтения.
- **seen** – закрывает текущий открытый файл и делает текущим входным устройством клавиатуру дисплея.
- **tell(+F)** – открывает файл с именем F в качестве текущего выходного устройства.
- **telling(?F)** – выдает имя файла, открытого для записи.
- **told** – закрывает текущий открытый для записи файл и делает текущим выходным устройством экран дисплея.

SWI-Prolog поддерживает предикаты `see`, `seeing`, `seen`, `tell`, `telling`, `tuld`, принятые в некоторых ранее появившихся версиях Пролога. Предикаты `open`, `close` SWI-Prolog исполняет много быстрее, их использование предпочтительнее и потому настоятельно рекомендуется.

9.8.14. Движение в файле

- **`seek(+H,+Offset,+Metod, -Newloc)`** – ставит указатель в файле с именем, которому приписано число `H`; в соответствии с аргументом `Offset`, равняющимся числу байт, равным началу относительного размещения выбранного куска файла; с аргументом `Method`, равным одному из следующих атомов:
 - `bof` – число байт `Offset` отсчитывается от начала файла;
 - `eof` – число байт `Offset` отсчитывается от конца файла;
 - `current` – число байт `Offset` отсчитывается от текущей позиции; возвращает новую позицию `Newloc` указателя от начала содержимого файла. Побайтовый отсчет содержимого файла начинается с нуля.

9.8.15. Исполнение системных функций

- **`chdir(?Path)`** – возвращает текущий директорию, если переменная `Path` не замещена; меняет текущий директорию, если переменная `Path` замещена (именем `Path` нового директория).
- **`consult(+F)`** – дописывает программу с файла `F` к уже имеющейся в рабочей области программе. Расширение `ari` не обязательно.
- **`delete(+F)`** – удалить файл с именем `F`.
- **`edit(+F)`** – выход из интерпретатора в редактор для редактирования файла `F`. Отредактированный текст следует переписать в рабочую область с помощью предиката `reconsult`.
- **`halt`** – выход из системы в среду DOS.
- **`shell`** – выход в MS DOS. Для возвращения в систему набрать `exit`.
- **`shell(+DC)`** – выход в MS DOS и исполняет DOS-команду `DC`.
- **`statistics`** – возвращает информацию об использованных системой ресурсах.
- **`statistics(+Info,-Struct)`** – возвращает информацию об использованных системой ресурсах. `Info` может быть: стек, база данных, `page(X)` (т. е. страница с номером `X`), атом (и тогда система дает информацию по этому атому).
- **`time(?Time)`** – задает время в формате: `time(час, минута, секунда, сотая доля секунды)`.

9.8.16. Отладчик (Debugger)

Программа при исполнении проходит через четыре вида портов: `call`, `exit`, `redo`, `fail`.

- **`leash(+Mode)`** – определяет характер прохождения портов при режиме трассировки. Аргумент `Mode` принимает одно из следующих значений:

- full – отладчик останавливается по всем портам,
- tight – отладчик останавливается в портах: call, redo, fail,
- half – отладчик останавливается в портах: call, redo;
- loose – отладчик останавливается в порте: call,
- off – отладчик не останавливается ни в одном порте.

Значение аргумента Mode можно задать натуральным числом от 0 до 15. Двоичное представление этого числа задает остановочные порты. Например, двоичная запись числа 10 есть 1010; тогда `leash(10)` означает, что отладчик будет останавливаться в портах call, redo, ибо в последовательности call, exit, redo, fail двум единицам последовательности 1,0,1,0 соответствуют call и redo.

- **nospy(+N)** – отменяет трассировочные отметки для предиката с именем N.
- **nospy(+N/A)** – отменяет трассировочные отметки для предиката с именем N местности A.
- **notrace** – выключить отладчик.
- **spy(+N)** – отмечает для трассировки предикат с именем N.
- **spy(+N/A)** – отмечает для трассировки предикат с именем N местности A.
- **trace** – включить отладчик. После включения отладчика транслятор при трассировке начинает выдавать следующие сопроводительные знаки:
 - ** – цель помечена для трассировки,
 - *> – цель помечена для трассировки; отладчик достиг этой цели в результате прыжка,
 - > – цель не помечена для трассировки; отладчик достиг этой цели в результате прыжка,
 - => – отладчик трассирует бэктрекинг.

Режимом трассировки можно управлять, набирая на экране следующие буквы:

- ; – указывает на порт exit;
- @ – позволяет вызвать любую цель и возвратиться в режим отладки после исполнения цели;
- a – прекращает исполнение программы и возвращается в режим интерпретации;
- b – прекращает исполнение программы; отладка возобновляется набором `end_of_file`;
- c (или клавиша return) – переход к следующему порту;
- d – вызывает на экран текущую цель;
- e – выход из интерпретатора и возвращение в среду MS DOS;
- f – переход к порту fail;
- h – помощь в командах отладки;
- l – при трассировке последовательный переход от одного отмеченного (для трассировки) предиката к следующему предикату (отмеченному для трассировки);

- n – выключение отладчика;
- q – при трассировке посещение портов `exit` и `fail`. Если трассируемая цель помечена, то посещаются все порты;
- s – посещение портов `exit` и `fail` вне зависимости от трассировочных пометок;
- w – пишет текущую цель (используя предикат `write`);
- x – используется в портах `fail` и `redo`. Отладчик осуществляет `fail`, пока не встретится порт `call` или `exit`.

Замечание. Предложения стандартного Пролога с точки зрения логики являются формулами логики предикатов первого порядка с равенством, с бинарным функциональным символом $.(X,Y)$ для организации списков и одним конструктором $|$ для выделения головы и хвоста списка. Стандартный Пролог является универсальным языком программирования в том смысле, что на нем может быть вычислена любая частично рекурсивная функция. Стандартный Пролог входит в любую реализацию Пролога. Всякий конкретный Пролог для удобства вычислений имеет много встроенных предикатов и функций. Все они допустимы в формулах прологовой логики, хотя при желании без этих предикатов и функций можно было бы обойтись.

Дизъюнкт $A \vee B \vee \dots \vee C \vee \neg D \vee \neg E \vee \dots \vee \neg F$, где $A, B, \dots, C, D, E, \dots, F$ являются атомами (элементарными формулами логики предикатов), равносильен формуле $D \& E \& \dots \& F \rightarrow A \vee B \vee \dots \vee C$ или формуле $A \vee B \vee \dots \vee C \leftarrow D \& E \& \dots \& F$, которую записывали в виде секвенции $D, E, \dots, F \models A; B; \dots; C$, или с точки зрения логического программирования в виде правила $A; B; \dots; C :- D, E, \dots, F$. Хорнов дизъюнкт $A \vee \neg D \vee \neg E \vee \dots \vee \neg F$ соответствует секвенции $D \& E \& \dots \& F \models A$ или правилу $A :- D, E, \dots, F$ Пролога. В сущности, в методе резолюций обрабатываются правила вида $A; B; \dots; C :- D, E, \dots, F$. Работа с такими правилами (дизъюнктами с точки зрения логики предикатов) при построении резолютивного вывода составляет основную задачу логического программирования. Предложения Пролог-программы A можно рассматривать как нелогические аксиомы некоторой элементарной теории T . Отвечая на запрос B , транслятор вычисляет, выводим ли запрос B из аксиом A в теории T , т. е. доказуема ли формула B в элементарной теории T . Другими словами, при каждом исполнении программы вычисляется, следует ли логически B из A , т. е. является ли формула $A \rightarrow B$ в логике предикатов общезначимой. Поэтому можно сказать, что при каждом исполнении конкретной программы решается основной вопрос математической логики для некоторой конкретной формулы логики предикатов.



.....



МОНАДИЧЕСКАЯ ЛОГИКА И КОНЕЧНЫЕ АВТОМАТЫ

В результате освоения учебного материала данной части студент должен:

- знать основные понятия интерпретируемого формализма – монадической логики второго порядка на натуральных числах с функцией следования; основные понятия теории конечных автоматов, играющих важную роль в производстве компьютеров, в построении синтаксических анализаторов языков программирования, в представлении словарных языков формулами монадической логики и конечными автоматами;
- уметь анализировать и представлять формулы монадической логики в специальной «автоматной» форме; анализировать и представлять конечно автоматный словарный язык формулой монадической логики;
- иметь навыки минимизации конечных автоматов по числу состояний; навыки представления регулярных языков конечными автоматами; навыки представления конечноавтоматных языков формулами монадической логики.

Глава 10



Конечные автоматы

10.1. Автоматы Мили и Мура

Алфавит X есть любое непустое конечное множество символов (букв). Слово в алфавите X есть конечная последовательность символов алфавита X . Пусть $*$ есть символ пустого слова; длина слова есть число входящих в него символов; длина пустого слова равна нулю. Два слова (графически) равны, если они представляют одну и ту же последовательность символов алфавита. Конкатенация xu (или $x \cdot u$) двух слов x и u есть результат приписывания к слову x слова u . Пусть X^* есть множество всех слов в алфавите X . Язык есть произвольное подмножество слов во множестве X^* .

Автомат Мили есть система объектов $A = (X, Y, Q, q_0, T, B)$, где X есть входной алфавит; Y есть выходной алфавит; Q есть алфавит (внутренних) состояний; $q_0 \in Q$ есть начальное состояние; $T : X \times Q \rightarrow Q$ есть функция переходов; $B : X \times Q \rightarrow Y$ есть функция выходов.

Пример. Алфавиты $X = \{0, 1\}$; $Y = \{0, 1, 2\}$; $Q = \{q_0, q_1, q_2\}$; функции переходов и выходов задаются в табл. 10.1 или диаграммой (рис. 10.1).

Таблица 10.1

	q_1	q_2	q_3
0	$q_3/1$	$q_3/0$	$q_1/1$
1	$q_2/2$	$q_3/2$	$q_2/2$

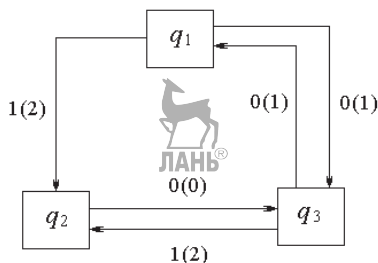


Рис. 10.1

Автомат A работает следующим образом. На вход автомата в дискретные моменты времени поступает входная последовательность $x = x(0)x(1)x(2)...x(k)$ символов алфавита X . Автомат A в ответ проходит последовательность состояний $q = q(0)q(1)q(2)...q(k)q(k+1)$. Она называется *ходом* автомата A на x . Пусть $Rn(A, x)$ есть множество всех таких последовательностей. Последовательность состояний можно задать как

$$\begin{aligned} q(0) &= q_0, \\ q(t+1) &= T(q(t), x(t)). \end{aligned}$$

Автомат выдает выходную последовательность $y = y(0)y(1)y(2)...y(k)$, для которой $y(t) = B(q(t), x(t))$, $t = 0, 1, \dots, k$.

В примере автомата из табл. 10.1:

$$\begin{aligned} x &= 0 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 0 \ 0, \\ q &= q_0 \ q_2 \ q_1 \ q_2 \ q_1 \ q_2 \ q_2 \ q_1 \ q_2 \ q_1, \\ y &= 1 \ 1 \ 2 \ 1 \ 2 \ 2 \ 1 \ 0 \ 1. \end{aligned}$$

Автомат Мили можно задать каноническими уравнениями:

$$\begin{aligned} q(0) &= q_0, \\ q(t+1) &= T(q(t), x(t)), \\ y(t) &= B(q(t), x(t)). \end{aligned}$$

Автомат Мура есть система объектов $A = (X, Y, Q, q_0, T, B)$, где X, Y, Q, q_0, T задаются как для автомата Мили, а $B : Q \rightarrow Y$ есть функция выходов. Канонические уравнения автомата Мура имеют вид:

$$\begin{aligned} q(0) &= q_0, \\ q(t+1) &= T(q(t), x(t)), \\ y(t) &= B(q(t)). \end{aligned}$$

Пример. $X = \{0, 1\}$; $Y = \{0, 1, 2\}$; функции переходов и функция выходов задаются в табл. 10.2 и диаграммой на рис. 10.2.

Таблица 10.2

	1	2	0
	q_1	q_2	q_3
0	q_3	q_2	q_0
1	q_1	q_2	q_2

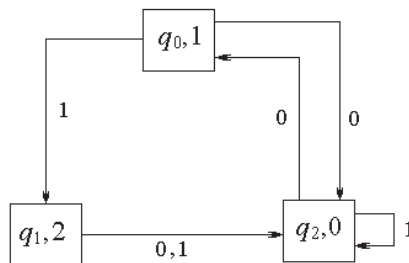


Рис. 10.2

Ниже приведена входная последовательность x , а также соответствующие ей последовательность состояний q и выходная последовательность y :



$$\begin{aligned}x &= 0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1, \\q &= q_0 \ q_2 \ q_0 \ q_1 \ q_2 \ q_0 \ q_2 \ q_2, \\y &= 1 \ 0 \ 1 \ 2 \ 0 \ 1 \ 0.\end{aligned}$$

Автомат Мура есть частный случай автомата Мили. Автомат Мура проще и потому в некоторых случаях предпочтительнее автомата Мили.

Распространим функцию переходов T автомата на множество $Q \times X^*$, положив $T(q, xa) = T(T(q, x), a)$; $q \in Q, x \in X^*, a \in X$. Значение функции $T(q, x)$ есть то состояние, в которое перейдет автомат A из состояния q под воздействием входного слова x .

Аналогичным образом распространим функцию выходов B на множество $Q \times X^*$, положив $B(q, xa) = B(B(q, x), a)$; $q \in Q, x \in X^*, a \in X$. Значение функции $B(q, x)$ есть то выходное слово y , которое выдает автомат A , находящийся в состоянии q , если на его вход подать слово x .

Пусть $Q' \subseteq Q$ и $T(Q', x) = \{q \in Q : \exists q' \in Q' T(q', x) = q\}$ есть множество всех состояний, в которые перейдет автомат из состояний множества Q' в качестве начальных состояний под воздействием входного слова x . Определим для автомата $A = (X, Y, Q, q_0, T, B)$ оператор $\mathcal{O}: X^* \rightarrow Y^*$, положив $\mathcal{O}(a) = B(q_0, a), a \in X$; $\mathcal{O}(xa) = \mathcal{O}(x) \cdot B(T(q_0, x), a), x \in X^*, a \in X$.

Оператор $\mathcal{O}$ по входному слову x выдает соответствующее ему выходное слово y . Определим оператор $\mathcal{O}_Q: X^* \rightarrow Q^*$, положив $\mathcal{O}_Q(a) = q_0 \cdot T(q_0, a), a \in X$; $\mathcal{O}_Q(xa) = \mathcal{O}_Q(x) \cdot T(T(q_0, x), a)$; $x \in X^*, a \in X$.

Оператор $\mathcal{O}_Q$ по входному слову x выдает соответствующее ему слово в алфавите состояний автомата A . Если автомат A начинает работу над словом x из некоторого состояния q , то будем писать $\mathcal{O}(q, x), \mathcal{O}_Q(q, x)$.

Два автомата с выходом эквивалентны, если они реализуют один и тот же оператор $\mathcal{O}: X^* \rightarrow Y^*$.

Теорема. По любому автомату Мили можно построить эквивалентный ему автомат Мура.

Доказательство. Пусть $A = (X, Y, Q, q_0, T, B)$ есть автомат Мили. Построим автомат Мура $A' = (X, Y, Q', q', T', B')$, положив

$$\begin{aligned}q' &= q_0; \\Q' &= \{q_0\} \cup \{(q, a) : q \in Q, a \in X\}; \\T'((q, a), b) &= (T(q, a), b), q \in Q, \{a, b\} \subseteq X; \\B'((q, a)) &= B(q, a), q \in Q, a \in X.\end{aligned}$$

Построенный автомат Мура эквивалентен автомату Мили; однако для этого необходимо пренебречь выходом автомата Мура в начальный момент времени.

Пример. Для автомата Мили из табл. 10.1 функции переходов и выходов эквивалентного ему автомата Мура приведены в табл. 10.3. Выход $B'(q_0)$ произволен.

Таблица 10.3

	q_0	1 ($q_0, 0$)	2 ($q_0, 1$)	0 ($q_1, 0$)	2 ($q_1, 1$)	1 ($q_2, 0$)	2 ($q_2, 1$)
0	($q_0, 0$)	($q_2, 0$)	($q_2, 0$)	($q_2, 0$)	($q_2, 0$)	($q_1, 0$)	($q_1, 0$)
1	($q_0, 1$)	($q_1, 1$)	($q_1, 1$)	($q_2, 1$)	($q_2, 1$)	($q_2, 1$)	($q_2, 1$)

Автомат без выхода есть система объектов $A = (X, Q, q_0, T, F)$, где (X, Q, q_0, T) задаются как у автомата с выходом, а $F \subseteq Q$ есть множество выделенных (отмеченных, заключительных, финальных) состояний. Так, определенный автомат называется также *детерминированным конечным автоматом* (ДКА). Автомат без выхода можно рассматривать как автомат Мура с выходом, полагая его выход

$$y(t) = B(q(t)) = \begin{cases} 1, & \text{если } q(t) \in F, \\ 0, & \text{если } q(t) \notin F. \end{cases}$$

Слово $x \in X^*$ допустимо (определимо) автоматом $A = (X, Q, q_0, T, F)$, если $T(q_0, x) \in F$. Если $T(q_0, x) \notin F$, то слово x не допускается (отвергается) автоматом A .

Поведение $Beh(A)$ автомата A есть множество всех слов в алфавите X , допустимых автоматом A . Язык $L \subseteq X^*$ (автоматно) определим (допустим, представим), если существует автомат A без выхода, для которого $L = Beh(A)$.

Иногда автомат $A = (X, Q, q_0, T)$ задается без выхода и без множества выделенных состояний, иногда и без указания начального состояния $A = (X, Q, T)$. Два автомата без выхода эквивалентны, если они представляют один и тот же входной язык.

Прямое (декартово) произведение автоматов $A' = (X, Q', q', T')$ и $A'' = (X, Q'', q'', T'')$ есть автомат $A' \times A'' = (X, Q' \times Q'', (q', q''), T)$, где $T((q', q''), a) = (T'(q', a), T''(q'', a))$, $q' \in Q', q'' \in Q'', a \in X$.

Пример. Пусть диаграммы автоматов A' и A'' (без выходов) приведены на рис. 10.3 и 10.4. Диаграмма автомата $A' \times A''$ изображена на рис. 10.5.

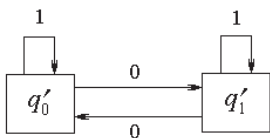


Рис. 10.3

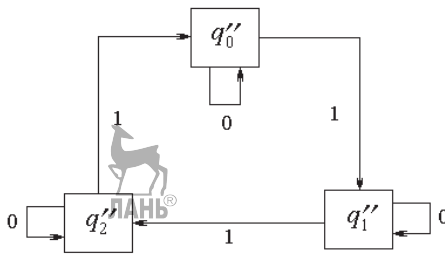


Рис. 10.4

Теорема. Класс автоматно представимых языков замкнут относительно булевых операций (объединения, пересечения, дополнения).

Доказательство. Пусть автоматы $A' = (X, Q', q', T', F')$ и $A'' = (X, Q'', q'', T'', F'')$ определяют языки $Beh(A')$ и $Beh(A'')$ соответственно.

Дополнение $CA' = X^* - Beh(A')$ определимо автоматом $(X, Q', q', T', Q' - F')$.

Пересечение $Beh(A') \cap Beh(A'')$ определимо декартовым произведением $A' \times A''$ со множеством выделенных состояний $F' \cap F''$.

Объединение $Beh(A') \cup Beh(A'')$ определимо декартовым произведением $A' \times A''$ со множеством выделенных состояний $F' \times Q'' \cup Q' \times F''$.

Следствие. Класс автоматов предствавимых языков замкнут относительно разности, ибо разность множеств $M - N = M \cap CN$.

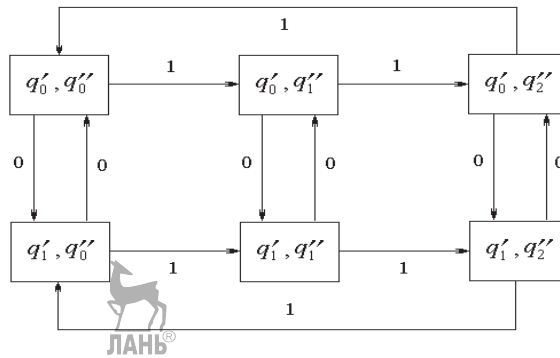


Рис. 10.5

10.2. Источники

Источник есть объект $S = (X, Q, Q_0, D, F)$, где X есть входной алфавит; Q есть алфавит состояний, $Q_0 \subseteq Q$ есть множество начальных состояний, $D \subseteq Q \times X \times Q$ есть (недетерминированная) таблица переходов (здесь в качестве входного сигнала допускается пустой символ, обозначаемый $*$), $F \subseteq Q$ есть множество выделенных состояний.

Тройка (q, a, q') из D называется *переходом* источника.

Пример. $X = \{0, 1, 2\}$; $Q = \{q_0, q_1, q_2, q_3\}$; $Q_0 = \{q_0, q_1\}$; $F = \{q_2, q_3\}$; $D = \{(q_0, 0, q_1); (q_0, 0, q_3); (q_0, 1, q_1); (q_0, *, q_0); (q_0, *, q_1); (q_1, *, q_1); (q_1, 2, q_2); (q_2, 0, q_2); (q_2, 0, q_3); (q_2, 1, q_2); (q_2, 2, q_2); (q_3, 1, q_2); (q_3, 1, q_3); (q_3, *, q_0)\}$.

Диаграмма этого источника изображена на рис. 10.6.

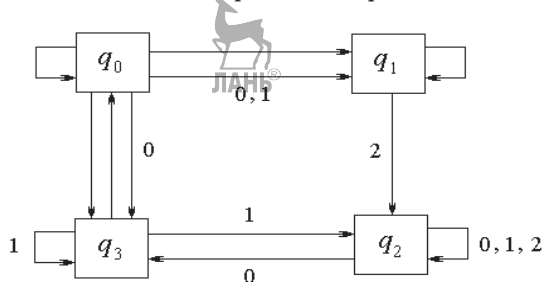


Рис. 10.6

Заметим, что в случае источника некоторые дуги (стрелки) в его диаграмме могут быть помечены пустым символом (т. е. дуги ничем не помечены).

Недетерминированный автомат есть частный случай источника, в котором нет дуг, помеченных пустым символом. Детерминированный автомат есть частный случай недетерминированного автомата, а потому и частный случай источника.

Входное слово $x = x(0)x(1)...x(k)$ допустимо источником S , если существует последовательность $Q(0), Q(1), ..., Q(i), ..., Q(k), Q(k+1)$ (она называется ходом источника S на входном слове x ; множество всех таких последовательностей обозначается через $Rn(S, x)$), где каждое $Q(i)$ есть конечная последовательность состояний $q(i, 0), q(i, 1), ..., q(i, n_i)$, для которой:

$$\begin{aligned} q(0, 0) &\in Q_0; \\ (q(i, j), *, q(i, j+1)) &\in D; i=0, 1, ..., k; j=0, 1, ..., n_i-1; \\ (q(i, n_i), x(i), q(i+1, 0)) &\in D; i=0, 1, ..., k; \\ q(k+1, n_{k+1}) &\in F. \text{ (См. рис. 10.6.)} \end{aligned}$$

В противном случае слово x источник S не допускает (отвергает). Для примера источника из рис. 10.6 на входном слове $x = 1\ 2\ 2\ 0\ 1\ 1\ 1\ 2$ возможна следующая последовательность состояний:

$x =$	$x(0)$	$x(1)$	$x(2)$	$x(3)$	$x(4)$	$x(5)$	$x(6)$	$x(7)$	
	1	2	2	0	1	1	1	2	
	q_0	q_1	q_2	q_2	q_3	q_3	q_3	q_2	q_2
	q_0	q_1			q_0		q_0		
	q_3				q_0		q_3		
	q_0				q_3				
					q_0				
					q_3				
					q_0				
					q_3				
	$Q(0)$	$Q(1)$	$Q(2)$	$Q(3)$	$Q(4)$	$Q(5)$	$Q(6)$	$Q(7)$	$Q(8)$
	$q(0, 0)$	$q(1, 0)$	$q(2, 0)$	$q(3, 0)$	$q(4, 0)$	$q(5, 0)$	$q(6, 0)$	$q(7, 0)$	$q(8, 0)$
	$q(0, 1)$	$q(1, 1)$			$q(4, 1)$		$q(6, 1)$		
	$q(0, 2)$				$q(4, 2)$		$q(6, 2)$		
	$q(0, 3)$				$q(4, 3)$				
					$q(4, 4)$				
					$q(4, 5)$				

Так как $q_2 \in F$, то слово x допустимо источником S .

Поведение $Beh(S)$ источника S есть множество всех слов, допустимых источником S . Два источника эквивалентны, если они имеют одинаковые поведения (т. е. если они допускают одно и то же множество входных слов). Язык $L \subseteq X^*$ представим (доустим, определим) источником, если существует источник S , для которого $L = Beh(S)$. Множество $Q' \subseteq Q$ состояний источника $S = (X, Q, Q_0, D, F)$ замкнуто, если $\forall q \in Q \forall q' \in Q' ((q', *, q) \in D \rightarrow q \in Q')$. Замыкание $[Q']$ множества Q' есть наименьшее замкнутое множество состояний, содержащее Q' .

Для источника на рис. 10.1 замыкание $[\{q_0\}] = \{q_0, q_1, q_3\}$; $[\{q_1\}] = \{q_1\}$; $[\{q_2, q_3\}] = \{q_0, q_1, q_2, q_3\}$; $[\{q_2\}] = \{q_2\}$.

Замечание. Всякий источник A эквивалентен некоторому двухполюсному источнику B с единственным начальным s и единственным финальным f состояниями. Источник B строится по источнику A следующим образом. В диаграмме источника A начальное состояние s соединяется пустыми стрелками с начальными состояниями в A . Все финальные состояния в A соединяются пустыми стрелками с финальным состоянием f . Источник B построен.

Теорема (о детерминизации источника). Пусть язык $L \subseteq X^*$.

Следующие утверждения эквивалентны.

1. Язык L представим конечным автоматом.
2. Язык L представим источником.

Доказательство. Следование $1 \rightarrow 2$ тривиально, ибо всякий автомат есть частный случай источника.

Покажем, что $2 \rightarrow 1$. Пусть язык L представим источником $S = (X, Q, Q_0, D, F)$. Построим автомат $A = (X, S, s_0, T, U)$, где $S \in P(Q)$ есть множество всех замкнутых подмножеств множества Q (включая пустое подмножество); начальное состояние $s_0 = [Q_0]$; функция переходов $T: S \times X \rightarrow S$ такова, что $T(s, a) = [\{q \in Q : \exists q' \in s (q', a, q) \in D\}]$; множество выделенных состояний $U = \{s \in S : \exists q \in s (q \in F)\}$.

Утверждение. $Beh(S) = Beh(A)$.

Доказательство. Пусть слово $x = x(0)x(1)...x(k) \in Beh(A)$. Тогда существует последовательность состояний $s(0), s(1), ..., s(k), s(k+1)$ автомата A (рис. 10.7), для которой

$$\begin{aligned} s(0) &\in s_0; \\ T(s(j), x(j)) &= s(j+1), j = 0, 1, ..., k; \\ s(k+1) &\in U. \end{aligned}$$

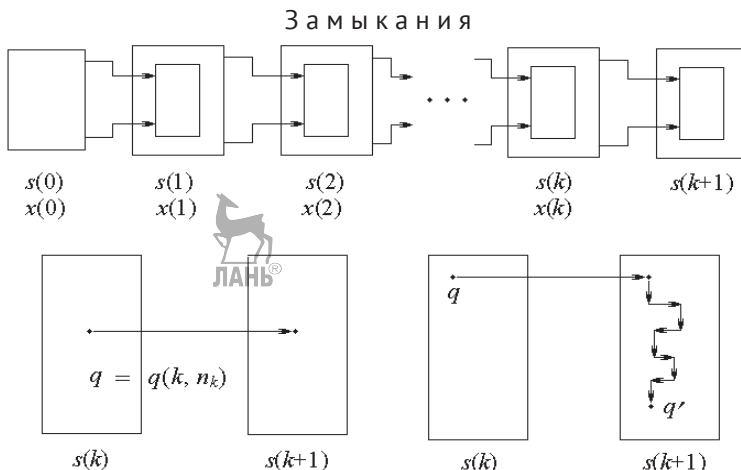


Рис. 10.7

Состояние $s(k+1) = [\{q \in Q : \exists q' \in s(k) ((q', a, q) \in D)\}] = s'$.

Так как $s(k+1) \in U$, то существует состояние $q' \in s(k+1)$, причем $q' \in F$. Возможны следующие случаи:

$q' \in s'$. Тогда $\exists q \in s(k) ((q, x(k), q') \in D)$;
 $q' \in [s']$, $q' \notin s'$, т.е., $q' \in [s'] - s'$.

Тогда существует последовательность состояний $Q(k+1)$, начинающаяся с некоторого состояния $q(k+1, 0) \in s'$ и заканчивающаяся состоянием $q(k+1, n) \in [s']$, причем $(q(k+1, j), *, q(k+1, j+1)) \in D, j = 0, 1, \dots, n_{k+1}-1$. Найдется состояние $q \in s(k)$, для которого $(q, x(k), q(k+1, 0)) \in D$. Продвигаясь к началу последовательности состояний автомата A , построим последовательность состояний источника

$$Q(0), Q(1), \dots, Q(k), Q(k+1), \quad (10.1)$$

для которой:

$$\begin{aligned} q(0, 0) &\in Q_0; \\ (q(i, n), x(i), q(i+1, 0)) &\in D, i = 0, 1, \dots, k; \\ (q(i, j), *, q(i, j+1)) &\in D, i = 0, 1, \dots, k+1; \\ j &= 0, 1, \dots, n_i-1; \\ q(k+1, n_{k+1}) &\in F. \end{aligned} \quad (10.2)$$

Поэтому слово $x \in Beh(S)$ и, следовательно, $Beh(A) \subseteq Beh(S)$.

Пусть теперь слово $x = x(0)x(1)\dots x(k) \in Beh(S)$. Тогда существует последовательность состояний (10.1), для которой выполняются условия (10.2). Для соответствующей последовательности $s(0), s(1), \dots, s(k+1)$ автомата A состояние $q(k+1, n_{k+1}) \in s(k+1)$, и потому $s(k+1) \in U$. Следовательно, слово $x \in Beh(A)$. Так что $Beh(S) \subseteq Beh(A)$, что вместе с $Beh(A) \subseteq Beh(S)$ дает $Beh(A) = Beh(S)$.

Утверждение установлено. Теорема доказана.

Следствие. Класс множеств, представимых источниками, замкнут относительно булевых операций.

Заметим, что если язык L в алфавите X автоматно представим, то язык L автоматно представим над любым расширением X' алфавита X , ибо автомат, представляющий язык L над X , можно рассматривать как источник, представляющий язык L над X' .

Алгоритм детерминизации источника

Пусть источник $S = (X, Q, Q_0, D, F)$. Возьмем $Q' \subseteq Q, a \in X$. Пусть $S(Q', a) = \{q \in Q : \exists q' \in Q' ((q', a, q) \in D)\}$ есть множество всех состояний, в которые источник S переходит из состояний множества Q' под воздействием входной непустой буквы a из X . Автомат A с тем же поведением, что и источник S , строим следующим образом.

1. Формируем замыкание множества начальных состояний источника и объявляем это замыкание начальным состоянием конструируемого автомата.

2. Если состояние $s = Q' \subseteq Q$ автомата A уже построено, то $T(s, a) = [S(Q', a)]$ есть состояние, в которое перейдет автомат A из состояния s под воздействием буквы a .

3. Применяем п. 2 алгоритма до тех пор, пока его применение порождает новые состояния автомата A .

4. Объявляем выделенными те состояния $s = Q' \subseteq Q$ автомата A , которые содержат в себе выделенные состояния источника S .

Пример. $X = \{0, 1\}$; $Q = \{q_0, q_1, q_2\}$; $Q_0 = \{q_0\}$; $F = \{q_0, q_2\}$. Таблица переходов D источника $S = (X, Q, Q_0, D, F)$ изображена на рис. 10.8. Переходы детерминированного автомата A , эквивалентного источнику S , приведены в табл. 10.4. Выделенные состояния автомата A помечены звездочками.

Таблица 10.4

	*		*	$\emptyset$
	$\{q_0\}$	$\{q_1\}$	$\{q_0, q_2\}$	
0	$\{q_0, q_2\}$		$\{q_2, q_2\}$	$\emptyset$
1	$\{q_1\}$	$\{q_0, q_2\}$	$\{q_1\}$	$\emptyset$

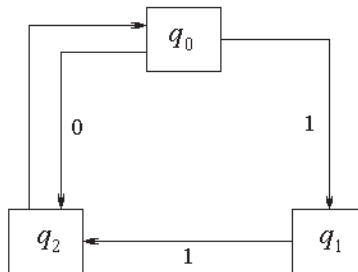


Рис. 10.8

Замечание. 1. Иногда будем задавать источник в виде $S = (X, Q, D)$ без указания множеств начальных и выделенных состояний.

2. **Недетерминированный конечный автомат** (НКА) есть источник без переходов вида $(q, *, q')$. Диаграмма такого источника не содержит стрелок, помеченных пустым символом ϵ .

3. Иногда источник называют недетерминированным конечным автоматом с ϵ -переходами.

Пусть Q', Q'', Q''' есть подмножества из Q ; $L(S, Q', Q'')$ есть поведение источника $S = (X, Q, Q', D, Q'')$, а язык $L_1 = L(S, Q', Q'', Q''')$ есть множество всех входных слов x , для которых существует ход $Rn(S, x)$, в котором все состояния лежат во множестве Q'' . Язык L_1 представим источником, который получается из источника S удалением всех ребер, инцидентных на диаграмме источника S , состояниям вне Q'' .

Замечание. Конечные автоматы могут применяться для моделирования программы поиска слов в тексте и распознавания ключевых слов.

10.3. Регулярные языки и регулярные выражения

10.3.1. Операции Клини и регулярные языки

Пусть X есть произвольный конечный алфавит.

Конкатенация (умножение) двух языков L_1 и L_2 из X^* есть множество $L_1 \cdot L_2 = \{z \in X^* : \exists x, y \in X^* (x \in L_1, y \in L_2, z = x \cdot y)\}$. **Степень k** языка L есть множество $L^k = L \cdot L \cdot \dots \cdot L$, k раз. Примем по определению, что $L^0 = \{*\}$, $\emptyset^0 = \{*\}$, $\emptyset^k = \emptyset$ для $k > 0$. Справедливо равенство $L \cdot \emptyset = \emptyset \cdot L = \emptyset$.

Итерация языка L из X^* есть множество $L^* = L^0 \cup L^1 \cup L^2 \cup L^3 \cup L^4 \cup \dots$

Операции Клини есть операции конкатенации, объединения, итерации.

Примеры. 1. $\{0,011,00\} \cdot \{11,101\} = \{011,0101,01111,011101,0011,00101\}$.

2. $\{0\}^* = \{0, 00, 000, \dots\}$.

3. $\{0,1\}^*$ есть множество всех непустых слов в алфавите $\{0,1\}$.

4. $\{0,1\}^k$ есть множество всех слов длины k в алфавите $\{0,1\}$.

Замечание. Положительная итерация $L^+ = L^1 \cup L^2 \cup L^3 \cup L^4 \cup \dots$

Определение (регулярного языка). Одноэлементные языки из X регулярны и имеют глубину построения 1. Если языки L, L_1, L_2 глубины построения k, k_1, k_2 соответственно регулярны, то языки $(L_1 \cup L_2)$, $(L_1 \cdot L_2)$, (L^*) регулярны и имеют глубину построения $\max(k_1, k_2)+1$, $\max(k_1, k_2)+1$, $k+1$ соответственно.

Будем опускать скобки при записи регулярного языка, принимая следующий приоритет операций: $*$, $\cdot$, $\cup$; для одноэлементного языка вместо $\{a\}$, писать и просто a ; вместо $\{a, b, \dots, c\}$ писать иногда $a \vee b \vee \dots \vee c$; вместо объединения $\cup$ при записи регулярных языков использовать также знак $\vee$.

Теорема. Класс языков, представимых источниками, замкнут относительно операций Клини.

Доказательство. Пусть языки L_1 и L_2 представимы источниками $S_1 = (X, Q', Q'_0, D', F')$ и $S_2 = (X, Q'', Q''_0, D'', F'')$ соответственно.

Объединение $L_1 \cup L_2$ представимо источником $S = (X, Q' \cup Q'', Q'_0 \cup Q''_0, D' \cup D'', F' \cup F'')$.

Конкатенация $L_1 \cdot L_2$ представима источником $S = (X, Q' \cup Q'', Q'_0, D, F'')$, где $D = D' \cup D'' \cup \{(q', *, q'') : q' \in F', q'' \in Q''_0\}$.

Итерация $(L_1)^*$ представима источником $S = (X, Q', Q'_0, D, F')$, где $D = D' \cup \{(q, *, q') : q \in F', q' \in Q'_0\}$.

Диаграммы источников S_1 и S_2 при объединении объединяются. Начальные состояния для S_1 и S_2 становятся начальными состояниями для S . Выделенные состояния для S_1 и S_2 являются выделенными состояниями для S .

Диаграммы источников S_1 и S_2 при конкатенации объединяются. Добавляются пустые (т. е. ничем не помеченные) стрелки, ведущие из выделенных состояний для S_1 в начальные состояния для S_2 . Начальные состояния для S_1 являются начальными состояниями для S . Выделенные состояния для S_2 являются выделенными состояниями для S .

При итерации в диаграмме для S_1 добавляются пустые стрелки из выделенных состояний для S_1 в его начальные состояния.

Следствие. Класс языков, представимых автоматами, замкнут относительно операций Клини.

Теорема (синтеза). Каждый регулярный язык представим некоторым источником.

Доказательство. Индукция по глубине k построения языка.

Базис. $k = 1$. Одноэлементные языки автоматно представимы; следовательно, они представимы и источниками.

Предположение индукции. Предположим, что все регулярные языки с глубиной построения меньше k представимы источниками.

Шаг индукции. Покажем, что все регулярные языки с глубиной построения k представимы источниками. Пусть регулярный язык L имеет глубину построения k . Тогда L имеет один из видов: $L_1 \cup L_2, L_1 \cdot L_2, L_1^*$. Так как регулярные языки L_1, L_2 имеют глубину построения меньше k , то по предположению индукции они представимы некоторыми источниками. Тогда язык L представим источником по теореме о замкнутости класса языков, представимых источниками, относительно операций Клини.

Следствие. Каждый регулярный язык автоматно представим.

Теорема (анализа). Всякий язык, представимый источником, регулярен.

Доказательство. Так как классы языков, представимых источниками и автоматами, совпадают, то теорему достаточно доказать для автоматов. Пусть автомат $A = (X, Q, q_0, T, F)$, где $Q = \{q_0, q_1, \dots, q_n\}$, $F = \{q_{i_1} \dots q_{i_m}\}$, представляет язык $L = Beh(A)$. Построим множество входных слов $E_{ij}^{(k)}$ с помощью следующей индуктивной процедуры:

$$\begin{aligned} E_{ij}^{(0)} &= \{a \in X : T(q_i, a) = q_j\}; \\ E_{ij}^{(k)} &= E_{ij}^{(k-1)} \cup E_{ik}^{(k-1)} \cdot (E_{kk}^{(k-1)})^* \cdot E_{kj}^{(k-1)}, \quad k = 0, 1, \dots, n. \end{aligned}$$

Если автомат A при подаче входного слова $x = x(0)x(1)\dots x(r-1)$ проходит последовательность состояний $q(0) = q_i, q(1), q(2), \dots, q(r-1), q(r) = q_j$, то скажем тогда, что x переводит A из q_i в q_j , и автомат A проходит при этом через состояния $q(1), q(2), \dots, q(r-1)$. Индукцией по k можно показать, что множество входных слов $E_{ij}^{(k)}$ переводит A из q_i в q_j и не проводит A через состояния q_v с номерами $v > k$. Пусть $M_{ij}^{(k)}$ есть множество всех входных слов, которые переводят A из q_i в q_j и при этом не проводят A через состояния q_v с номерами $v > k$.

Утверждение. $E_{ij}^{(k)} = M_{ij}^{(k)}$.

Доказательство. Индукция по k . Для $k = 0$ утверждение очевидно. Пусть оно верно для $k = r-1$, т. е. $E_{ij}^{(r-1)} = M_{ij}^{(r-1)}$. Покажем, что оно верно для $k = r$. Пусть $\hat{M}_{ij}^{(r)}$ (или $\hat{E}_{ij}^{(r)}$) есть множество всех входных слов из $M_{ij}^{(r)}$ (или из $E_{ij}^{(r)}$), которые проводят A через состояние q_r . Ясно, что $\hat{E}_{ij}^{(r)} = E_{ir}^{(r-1)} \cdot (E_{rr}^{(r-1)})^* \cdot E_{rj}^{(r-1)}$. Множество $M_{ij}^{(r)} = M_{ij}^{(r-1)} \cup \hat{M}_{ij}^{(r)} = E_{ij}^{(r-1)} \cup \hat{M}_{ij}^{(r)}$.

Покажем, что $M_{ij}^{(r)} = \hat{E}_{ij}^{(r)}$. Если входное слово $x \in \hat{M}_{ij}^{(r)}$ проводит A через q_r s раз, то x представимо в виде $x = x_0 x_1 \dots x_{s-1} x_s$, (рис. 10.9). Тогда

$$\begin{aligned} x_0 &\in M_{ir}^{(r-1)} = E_{ir}^{(r-1)}; \quad x_s \in M_{rr}^{(r-1)} = E_{rr}^{(r-1)}; \\ x_i &\in M_{rr}^{(r-1)} = E_{rr}^{(r-1)}, \quad i = 1, 2, \dots, s-1; \end{aligned}$$

$$x_1 \cdot x_2 \cdot \dots \cdot x_{s-1} \in (E_{rr}^{(r-1)})^*.$$

Поэтому $x \in E_{ir} \cdot (E_{rr}^{(r-1)})^* \cdot E_{rj}^{(r-1)} = \hat{E}_{ij}^{(r)}$. Отсюда $\hat{M}_{ij}^{(r)} \subseteq \hat{E}_{ij}^{(r)}$.

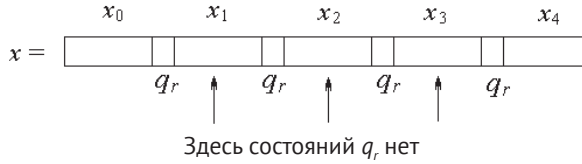


Рис. 10.9

Если $x \in \hat{E}_{ij}^{(r)}$, то слово x переводит A из q_i в q_j , проводит A через q_r , не проводит A через состояния q_v с номерами $v > r$, т. е. $x \in \hat{M}_{ij}^{(r)}$. Поэтому $\hat{E}_{ij}^{(r)} \subseteq \hat{M}_{ij}^{(r)}$. Равенство $\hat{M}_{ij}^{(r)} = \hat{E}_{ij}^{(r)}$ доказано. Тогда

$$M_{ij}^{(r)} = E_{ij}^{(r-1)} \cup \hat{M}_{ij}^{(r)} = E_{ij}^{(r-1)} \cup \hat{E}_{ij}^{(r)} = E_{ij}^{(r)}.$$

Утверждение доказано. Продолжим доказательство теоремы. Исходный автомат A имеет $n+1$ состояний $q_0, q_1, \dots, q_n$.

Язык $L = E_{0,i_1}^{(n)} \cup \dots \cup E_{0,i_m}^{(n)}$ регулярен согласно утверждению о регулярности всех дизъюнктивных слагаемых.

Теорема доказана.

Замечание. Доказанные выше теоремы останутся справедливыми, если вместо L^* взять $L^+ = L \cup L^2 \cup \dots$. Следует только слегка изменить их доказательства, сделав поправку на пустое слово.

10.3.2. Алгебра регулярных выражений Клини

Алфавит (формальные символы).

1. $e, \emptyset$ есть символы констант для пустого слова e и пустого множества $\emptyset$.
2. $a, b, c, \dots$ есть символы констант для одноэлементных словарных множеств.
3. $L, G, H, \dots$ есть символы переменных для словарных множеств.
4. $+$ есть символ для операции объединения словарных множеств.
5. $\cdot$ есть символ для операции конкатенации словарных множеств.
6. $*$ есть символ для операции итерации словарных множеств.

Замечание. $+, \cdot, *$ есть операции Клини.

Регулярные выражения (Клини).

1. Символы $e, \emptyset$ есть регулярные выражения глубины построения 0.
2. Символы $a, b, c, \dots$ есть регулярные выражения глубины построения 1.
3. Если L и G есть регулярные выражения глубины построения r и s соответственно, то $(L + G)$, $(L \cdot G)$, (L^*) есть регулярные выражения глубины построения $\max(r, s) + 1$, $\max(r, s) + 1$, $r + 1$ соответственно.

Примем соглашение об опускании скобок в регулярном выражении в соответствии со следующим приоритетом операций Клини: $*$, $\cdot$, $+$.

Интерпретация. Пусть языки $L(e) = \{e\}$, $L(\emptyset) = \emptyset$, $L(a) = \{a\}$ для всякого символа a . Если регулярные выражения R, R_1, R_2 представляют регулярные язы-

ки $L(R)$, $L(R_1)$, $L(R_2)$ соответственно, то регулярные выражения $R_1 + R_2$, $R_1 \cdot R_2$, R^* представляют регулярные языки $L(R_1 + R_2) = L(R_1) \cup L(R_2)$, $L(R_1 \cdot R_2) = L(R_1) \cdot L(R_2)$, $L(R^*) = L(R)^*$ соответственно. Тогда всякое регулярное выражение R представляет некоторый регулярный язык $L(R)$.

Определение. Два регулярных выражения эквивалентны (равны), если они представляют один и тот же регулярный язык.

Замечание. Два регулярных выражения эквивалентны, если им соответствующие регулярные языки представляются эквивалентными конечными автоматами.

Построим формально аксиоматическое исчисление, описывающее эквивалентные регулярные выражения.

Пусть A, B, C, X есть регулярные выражения. Пусть нейтральный элемент 1 по конкатенации есть пустое слово ϵ и нейтральный элемент 0 по объединению есть пустое множество $\emptyset$. Пусть для простоты записи $A \cdot B = AB$. Пусть $A \leq B$ есть частичный порядок, задаваемый как $A \leq B \leftrightarrow A + B = B$.

Схемы аксиом (D. Kozen).

$$A + (B + C) = (A + B) + C. \quad A + B = B + A. \quad A + 0 = A. \quad A + A = A.$$

$$A(BC) = (AB)C. \quad 1A = A. \quad A1 = A.$$

$$A(B + C) = AB + AC. \quad (A + B)C = AC + BC.$$

$$0A = 0. \quad A0 = 0.$$

$$1 + AA^* \leq A^*. \quad 1 + A^*A \leq A^*.$$

$$B + AX \leq X \rightarrow A^*B \leq X. \quad B + XA \leq X \rightarrow BA^* \leq X.$$

Теорема (D. Kozen). Регулярные выражения A и B эквивалентны (равны), если и только если равенство $A = B$ доказуемо в аксиоматической системе Козена.

Доказательство основано на алгебраическом анализе алгоритма минимизации конечных автоматов для регулярных выражений A и B и эквивалентности автоматов.

Замечание. Из теоретико-множественных соображений в алгебре Клини можно доказать следующие равенства. Их можно доказать и в аксиоматической системе Козена.

$$A + (B + C) = (A + B) + C. \quad A + B = B + A. \quad A + 0 = A. \quad A + A = A.$$

$$A(BC) = (AB)C. \quad 1A = A. \quad A1 = A.$$

$$A(B + C) = AB + AC. \quad (A + B)C = AC + BC.$$

$$0A = 0. \quad A0 = 0.$$

$$(A^*)^* = A^*. \quad 0^* = 1. \quad 1^* = 1. \quad A^+ = AA^*. \quad A^* = A^+ + 1. \quad (A + B)^* = (A^*B^*)^*.$$

10.4. Теоремы замкнутости для класса автоматов представимых языков

Пусть X есть конечный алфавит. Обращение слова $x = x(0)x(1)\dots x(k-1)x(k)$ из X^* есть слово $x^{-1} = x(k)x(k-1)\dots x(1)x(0)$. Если множество $M \subseteq X^*$, то $M^{-1} = \{x^{-1} : x \in M\}$.

Теорема. Класс языков, представимых источниками, замкнут относительно операции обращения.

Доказательство. Пусть язык M представим источником $S = (X, Q, Q_0, D, F)$. Тогда язык M^{-1} представим источником $S' = (X, Q, F, D', Q_0)$, где $D' = \{(q', a, q) : (q, a, q') \in D\}$, т. е. в диаграмме источника S все стрелки меняют свое направление на противоположное.

Следствие. Класс автоматов представимых языков замкнут относительно операции обращения.

Замечание. Если язык $L(E)$ представим регулярным выражением E , то регулярное выражение E^{-1} для языка $L(E)^{-1}$ строится индукцией по построению регулярного выражения. Если E есть e , $\emptyset$ или a , то $E^{-1} = E$. Если E есть $F+G$, $F \cdot G$ или F^* , то E^{-1} есть $F^{-1} + G^{-1}$, $G^{-1} \cdot F^{-1}$ или $(F^{-1})^*$ соответственно. $\square$

Пусть $X = \{a_0, a_1, \dots, a_k\}$, $Y = \{b_0, b_1, \dots, b_l\}$ есть два конечных алфавита, и пусть функция $f : X \rightarrow Y$ осуществляет проекцию с одного алфавита на другой (т. е. с алфавита X на алфавит Y). Пусть $x = x(0)x(1)\dots x(r)$ есть слово в алфавите X . Тогда слово $f(x) = f(x(0))f(x(1))\dots f(x(r))$ есть проекция слова x при отображении f . Если $M \subseteq X^*$, то $f(M) = \{f(x) : x \in M\}$ есть проекция множества M при отображении f .

Теорема. Класс языков, представимых источниками, замкнут относительно проекции.

Доказательство. Пусть язык M представим источником $S = (X, Q, Q_0, D, F)$ и функция $f : X \rightarrow Y$ осуществляет проекцию с алфавита X на алфавит Y . Тогда язык $f(M)$ представим источником $S' = (f(X), Q, Q_0, D', F)$, где $D' = \{(q, b, q') : \exists a \in X (f(a) = b \ \& \ (q, a, q') \in D)\}$, т. е. в диаграмме источника S всякая пометка a из X заменяется на пометку $f(a)$ из Y .

Следствие. Класс автоматов представимых языков замкнут относительно проекции.

Замечание. Если язык $L(E)$ представим регулярным выражением E , то при проекции f с одного алфавита на другой язык $f(L(E))$ представим регулярным выражением $f(E)$, получающимся из E заменой каждого символа a в E на символ $f(a)$. $\square$

Пусть $x = x(0)x(1)\dots x(k)$ есть некоторое слово в алфавите X , причем слово x содержит букву a . Аннулирование буквы a в слове x есть удаление в слове x буквы a всюду, где она встречается. Обозначим эту операцию через $An(x, a)$. Пусть $M \subseteq X^*$. Пусть тогда $An(M, a) = \{An(x, a) : x \in M\}$.

Теорема. Класс языков, представимых источниками, замкнут относительно операции аннулирования.

Доказательство. Пусть язык M представим источником $S = (X, Q, Q_0, D, F)$. Тогда язык $An(M, a)$ представим источником $S' = (X, Q, Q_0, D', F)$, где $D' = \{(q, b, q') \in D : b \neq a\} \cup \{(q, *, q') : (q, a, q') \in D\}$, т. е. в диаграмме источника S удаляются буквы a всюду, где они встречаются, но сами стрелки, которые были помечены буквой a , остаются.

Следствие. Класс автоматов представимых языков замкнут относительно операции аннулирования. $\square$

Пусть $x = x(0)x(1)...x(k)aa...a$ при $x(k) \neq a$ есть слово в алфавите X , содержащем букву a . Тогда операция усечения слова x по букве a (обозначение: $Trunc(x, a)$) определяется как $Trunc(x, a) = x(0)x(1)...x(k)$. Если $M \subseteq X^*$, то множество $Trunc(M, a) = \{Trunc(x, a) : x \in M\}$.

Теорема. Класс языков, представимых источниками, замкнут относительно операции усечения.

Доказательство. Пусть язык M представим источником $S = (X, Q, Q_0, D, F)$. Пусть $a \in X, a^k = aaa...a, k$ раз. Построим источник $S' = (X, Q, Q_0, D, G)$, где G есть F плюс множество всех тех состояний $q \in Q$, для которых существует слово a^k при некотором натуральном k , переводящее источник S из состояния q в состояние q' из F . Источник S' с поведением $M' = Beh(S')$ допускает все те слова, которые можно продолжить буквами a до слова, допустимого источником S , а также все слова, допустимые источником S .

Пусть S'' есть источник, допускающий множество M'' всех слов в алфавите X , не заканчивающихся на букву a . Источник, допускающий множество $M' \cap M''$, искомым.

Следствие. Класс автоматов представимых языков замкнут относительно операции усечения.

Пусть X и Y есть два алфавита и множество $M \subseteq X^*$. Цилиндр, восстановленный из множества M по второй оси, есть множество $Cyl_2 M = \{(x, y) \in (X \times Y)^* : x \in M\}$.

Теорема. Класс языков, представимых источниками, замкнут относительно операции восстановления цилиндра.

Доказательство. Пусть язык $M \subseteq X^*$ представим источником $S = (X, Q, Q_0, D, F)$ и Y есть другой алфавит. Тогда множество $Cyl_2 M$ представимо источником $S' = (X \times Y, Q, Q_0, D', F)$, где $D' = \{(q, (a, b), q') : (q, a, q') \in D \text{ и } b \in Y\}$, т. е. в диаграмме источника S все ребра, помеченные символом a из X , помечаем символами (a, b) , где $b \in Y$.

Следствие. Класс автоматов представимых языков замкнут относительно операции восстановления цилиндра. $\square$

Операцию восстановления цилиндра можно задать и по первой оси. Эту операцию можно распространить на случай восстановления цилиндра по любому количеству осей.

Принцип Дирихле (принцип голубятни). Пусть $A = \{a_1, a_2, \dots, a_m\}$ есть конечное множество из m элементов. Всякая последовательность элементов из A длины $n > m$ содержит пару одинаковых элементов.

Если A есть множество из m клеток, в которых размещается $n > m$ голубей, то одна из клеток содержит пару голубей.

Теорема. Алгоритмически разрешимы следующие свойства конечных автоматов.

1. Представимый автоматом язык пуст (проблема пустоты).
2. Представимый автоматом язык бесконечен.

Доказательство. Пусть $A = (X, Q, q_0, T, F)$ есть детерминированный конечный автомат.

1. Представимый автоматом язык пуст тогда и только тогда, когда A не допускает ни одного слова длины не более мощности алфавита X .

2. Представимый автоматом A язык бесконечен тогда и только тогда, когда автомат A допускает слово $x = x_0 x_1 x_2$, где $x_i \in X^*$, $i = 0, 1, 2$, причем для одного из $q \in Q$:

- x_0 переводит q_0 в q ;
- x_1 переводит q в q ;
- x_2 переводит q в выделенное состояние.

Последние три условия проверяются по п. 1.

Замечание. Пункт 2 можно уточнить следующей теоремой.

Теорема (лемма о накачке для регулярных языков). Пусть L есть регулярный язык. Существует константа n (зависящая от L), для которой каждое слово w из языка L , удовлетворяющее неравенству $|w| \geq n$, можно разбить на три слова $w = хуз$ так, что выполняются следующие условия:

1. $y \neq \epsilon$.
2. $|xy| \leq n$.
3. Для любого $k \geq 0$ слово $хy^kz$ также принадлежит L .

Это значит, что всегда можно найти такое слово y недалеко от начала слова w , которое можно «накачать». Таким образом, если слово y повторить любое число раз или удалить (при $k = 0$), то результирующее слово будет принадлежать языку L .

Доказательство. Пусть L есть регулярный язык. Тогда $L = L(A)$ для некоторого ДКА A . Пусть A имеет n состояний. Пусть

$$\begin{aligned} x &= x_0 x_1 \dots x_{i-1} x_i x_{i+1} \dots x_{k-1} x_k \dots x_{n-1}, \\ s &= s_0 s_1 \dots s_{i-1} s_i s_{i+1} \dots s_{k-1} s_k \dots s_{n-1} s_n, \end{aligned}$$

есть последовательности, где x есть произвольное допустимое автоматом A слово длины n , и автомат A на этом слове проходит последовательность состояний s длины $n+1$. При этом s_0 и s_n есть начальное и финальное состояния соответственно. Так как у автомата n состояний, то последовательность состояний s имеет два одинаковых состояния, скажем $s_i = s_k$, между которыми повторов состояний нет. Тогда слово $y = x_i x_{i+1} \dots x_{k-1}$ можно удалить из x и можно повторить между x_{i-1} и x_k сколько угодно раз. Результат будет допустим автоматом A .

На этом соображении строятся следующие более строгие рассуждения.

Рассмотрим произвольное слово w длины не менее n , скажем $w = a_1 a_2 \dots a_m$, где $m \geq n$ и каждый a_i есть входной символ. Для $i = 0, 1, 2, \dots, n$ определим состояние p_i как $d(q_0, a_1 a_2 \dots a_m)$, где d есть функция переходов автомата A , а q_0 есть его начальное состояние. Заметим, что $p_0 = q_0$.

Рассмотрим $n+1$ состояний p_i при $i = 0, 1, 2, \dots, n$. Поскольку автомат A имеет n различных состояний, то найдутся два разных целых числа i и j ($0 \leq i < j \leq n$), при которых $p_i = p_j$. Разобьем слово w на $хуз$.

1. $x = a_1 a_2 \dots a_i$.
2. $y = a_{i+1} a_{i+2} \dots a_j$.
3. $z = a_{j+1} a_{j+2} \dots a_m$.

Слово x приводит в состояние p_i , y из p_i обратно в p_i (так как $p_i = p_j$), а z есть остаток цепочки w . Слово x может быть пустым при $i = 0$, а z при $j = n = m$. Слово y не может быть пустым, ибо i строго меньше j .

Пусть на вход автомата A поступает слово xu^kz для любого $k \geq 0$. При $k = 0$ автомат переходит из начального состояния q_0 (которое есть также p_0) в p_i , прочитав x . Так как $p_i = p_j$, то z переводит A из p_i в допускающее состояние.

Если $k > 0$, то по x автомат A переходит из q_0 в p_i , затем, читая u^k , k раз циклически проходит через p_i и по z переходит в допускающее состояние. Поэтому для любого $k \geq 0$ слово xu^kz также допускается автоматом A , т. е. принадлежит языку L .

Определение. Пусть Σ и Ω есть конечные алфавиты. *Словарный гомоморфизм* (относительно операции конкатенации слов) есть функция $h: \Sigma \rightarrow \Omega^*$. $\square$

Пусть для всякого слова $w = a_1a_2\dots a_n$ из Σ^* слово $h(w) = h(a_1)h(a_2)\dots h(a_n)$ из Ω^* . Пусть $h(L) = \{h(w) : w \in L\}$ есть *гомоморфный образ языка L* в алфавите Σ .

Определение. Пусть E есть регулярное выражение в алфавите Σ . Пусть $h(E)$ есть результат замены в E каждого символа a из Σ на $h(a)$. Значение $h(E)$ определяется индуктивно по построению выражения E .

1. Если E есть e , $\emptyset$ или a из Σ , то $h(E)$ есть e , $\emptyset$ или $h(a)$ соответственно.
2. Если $E = F + G$, $E = F \cdot G$, $E = F^*$, то $h(E) = h(F+G) = h(F) + h(G)$, $h(E) = h(F \cdot G) = h(F) \cdot h(G)$, $h(E) = h(F^*) = h(F)^*$.

Замечание. Если E есть регулярное выражение, то $h(E)$ есть тоже регулярное выражение.

Пример. Пусть $\Sigma = \{0,1\}$, $\Omega = \{a, b\}$, $h(0) = ab$, $h(1) = e$, язык L задается регулярным выражением $E = 10^*1$. Тогда $h(0011) = h(0)h(0)h(1)h(1) = eababe = abab$, $h(E) = h(10^*1) = h(1)h(0)^*h(1) = e(ab)^*e = (ab)^*$.

Теорема. Если 1) регулярный язык R в алфавите Σ определяется регулярным выражением E , 2) h есть гомоморфизм на Σ , то язык $h(R)$ тоже регулярен и регулярное выражение $h(E)$ определяет $h(R)$, то есть $L(h(E)) = h(R)$.

Доказательство. Индукция по глубине k построения регулярного выражения E для R .

Базис. $K = 1$. Регулярный язык R определяется регулярным выражением E глубины построения 1. Тогда E есть e или $\emptyset$, $h(E) = E$, R есть $\{e\}$ или $\emptyset$, $h(R) = R$. Регулярное выражение $h(E)$ (равное E) определяет язык $h(R)$ (равный R). Если E есть a из Σ , то $R = \{a\}$, $h(E) = h(a)$, $h(R) = h(\{a\}) = \{h(a)\}$. Поэтому регулярное выражение $h(E)$ определяет язык $h(R)$. Во всех случаях $L(h(E)) = h(R)$.

Предположение индукции. Пусть теорема справедлива для регулярных языков, представимых регулярными выражениями глубины $k < r$.

Шаг индукции. Покажем, что теорема справедлива для регулярных языков, представимых регулярными выражениями глубины $k = r$. Пусть регулярный язык R определяется регулярным выражением E глубины построения r . Покажем, что регулярное выражение $h(E)$ представляет язык $h(R)$, то есть что $L(h(E)) = h(R)$.

Возможны три случая: $E = F+G$, $E = F \cdot G$, $E = F^*$.

Пусть $E = F+G$. Тогда по определению функции h и по определению операции $+$ для регулярных выражений имеем следующее:

$$\begin{aligned} R &= L(E) = L(F+G) = L(F) \cup L(G), \\ h(E) &= h(F+G) = h(F) + h(G), \\ L(h(E)) &= L(h(F+G)) = L(h(F) + h(G)) = L(h(F)) \cup L(h(G)). \end{aligned} \quad (*)$$

Так как h применяется к языку путем применения его к каждому слову этого языка по отдельности, то

$$h(R) = h(L(E)) = h(L(F+G)) = h(L(F) \cup L(G)) = h(L(F)) \cup h(L(G)). \quad (**)$$

Так как глубина построения F и G меньше r , то по предположению индукции $L(h(F)) = h(L(F))$, $L(h(G)) = h(L(G))$. Отсюда следует, что правые части в равенствах $(*)$ и $(**)$ равны. Поэтому равны и левые:

$$L(h(E)) = h(R).$$

Так как язык $h(R)$ представляется регулярным выражением $h(E)$, то язык $h(R)$ регулярен.

Два других случая рассматриваются аналогично.

Шаг индукции установлен. Теорема доказана.

Определение. Пусть Σ и Ω есть конечные алфавиты. Пусть функция $h: \Sigma \rightarrow \Omega^*$ есть гомоморфизм из Σ в Ω^* . Пусть L есть язык в алфавите Ω . Тогда обратный гомоморфизм $h^{-1}(L)$ есть множество всех слов w из Σ^* , для которых $h(w) \in L$.

Теорема. Если функция $h: \Sigma \rightarrow \Omega^*$ есть гомоморфизм из Σ в Ω^* и L есть регулярный язык в алфавите Ω , то язык $h^{-1}(L)$ тоже регулярен.

10.5. Минимизация числа состояний автомата

с выходом

Будем рассматривать автоматы с общим входным и общим выходным алфавитами. Два автомата эквивалентны, если они реализуют один и тот же оператор.

Пусть $A = (X, Y, Q, q_0, T, B)$ есть автомат с выходом, реализующий оператор $\mathcal{O}: X^* \rightarrow Y^*$.

Два состояния q', q'' автомата A r -отличимы, если существует входное слово $x = x(0)x(1)\dots x(r-1)$ длины r , для которого $B(q', x) \neq y' = y'' = B(q'', x)$. Состояния q, q' отличимы, если они r -отличимы при некотором r . Состояния q', q'' r -неотличимы, если для любого входного слова x длины не более r $B(q', x) = B(q'', x)$. Состояния q', q'' строго r -отличимы, если они r -отличимы, но не отличимы ни каким входным словом длины меньше r (т. е. $(r-1)$ -неотличимы). Состояния q' и q'' неотличимы, если они r -неотличимы при любом r .

Аналогичные определения отличимости состояний q' и q'' можно ввести для двух автоматов A и B , причем q' есть состояние автомата A , а q'' – автомата B .

Автоматы A и B , реализующие операторы O_A и O_B , r -отличимы, если существует входное слово x длины r , для которого $O_A(x) \neq O_B(x)$. Автоматы A и B отличимы, если они отличимы при некотором r . Автоматы A и B r -неотличимы, если $O_A(x) = O_B(x)$ для всякого входного слова x длины не более r . Автоматы A и B строго r -отличимы, если они r -отличимы и $(r-1)$ -неотличимы. Автоматы A и B неотличимы, если они r -неотличимы при любом r .

Если автоматы A и B неотличимы, то они реализуют одинаковые операторы и, следовательно, эквивалентны.

Состояние q'' *достижимо* из состояния q' автомата A , если $T(q', x) = q''$ для некоторого входного слова x .

Автомат A называется *приведенным*, если все его состояния достижимы из начального состояния и попарно отличимы.

Автомат A называется *минимальным*, если он имеет наименьшее число состояний среди всех автоматов, эквивалентных автомату A .

Теорема. Приведенный автомат является минимальным.

Доказательство. Допустим противное: приведенный автомат $A = (X, Y, Q, q_0, T, B)$ не является минимальным, а минимальным является автомат $A' = (X, Y, Q', q'_0, T', B')$, эквивалентный автомату A , но с числом состояний меньше, чем в A . Пусть $Q' = \{q'_0, q'_1, \dots, q'_m\}$. Пусть входные слова $x_0, x_1, \dots, x_m$ переводят автомат A' из начального состояния q'_0 в состояния $q'_0, q'_1, \dots, q'_m$, а автомат A в состояния $q_{i0}, q_{i1}, \dots, q_{im}$ соответственно. Так как автоматы A и B эквивалентны, то неразличимы пары состояний

$$q'_0, q_{i0}; q'_1, q_{i1}; \dots q'_m, q_{im}. \quad (10.3)$$

Пусть состояние $q \notin \{q_{i0}, q_{i1}, \dots, q_{im}\} = S$, но $q \in Q$. Так как A есть приведенный автомат, то состояние q отлично от каждого состояния из Q . Так как q достижимо из q_0 , то для некоторого входного слова x $T(q_0, x) = q$. Пусть $T'(q'_0, x) = q' \in \{q'_0, q'_1, \dots, q'_j, \dots, q'_m\}$ при некотором j . Так как автоматы A и B эквивалентны, то состояния q и q' неотличимы. Пусть состояние q' соответствует согласно (35.3) состоянию $q_{ij} \in S$. По (10.3) состояние q'_j неотлично от q_{ij} , но q'_j неотлично от q и потому q неотлично от q_{ij} . Противоречие. Следовательно, автомат A минимален.

Теорема. Если два состояния q, q' автомата A с k состояниями отличимы, то они отличимы входным словом длины не более k (т. е. k -отличимы).

Доказательство. Пусть состояния q, q' отличимы. Тогда они строго r -отличимы при некотором r , т. е. найдется входное слово $x = x(0)x(1)\dots x(r-1)$, для которого автомат A , начав работу над словом x в состояниях q и q' , перерабатывает его в различные слова $y = y(0)y(1)\dots y(r-1)$ и $y' = y'(0)y'(1)\dots y'(r-1)$ (т. е. $B(q, x) = y, B(q', x) = y'$) и проходит последовательность состояний

$$\begin{array}{ll} q(0), q(1), \dots, q(r); & q(0) = q; \\ q'(0), q'(1), \dots, q'(r); & q'(0) = q' \end{array}$$

соответственно. При этом две последние буквы в словах y и y' различны (т. е. $y(r) \neq y'(r)$). Для любых (дискретных) моментов времени t_1, t_2 ($0 \leq t_1 < t_2 \leq r-1$) пары (столбцы)

$$\begin{aligned} & q(t_1), q(t_1+1), \dots, q(t_2); \\ & q'(t_1), q'(t_1+1), \dots, q'(t_2) \end{aligned}$$



попарно различны, ибо при $q(t_1) = q(t_2)$, $q'(t_1) = q'(t_2)$, удалив из слова $x = x(0)x(1)\dots x(t_1)\dots x(t_2)\dots x(r-1)$ часть $x(t_1)x(t_1+1)\dots x(t_2-1)$, получим более короткое слово $x(0)x(1)\dots x(t_1-1)x(t_2)\dots x(r-1)$, перерабатываемое автоматом A в различные слова

$$\begin{aligned} & y(0)y(1)\dots y(t_1-1)y(t_2)\dots y(r-1), \\ & y'(0)y'(1)\dots y'(t_1-1)y'(t_2)\dots y'(r-1), \end{aligned}$$

в противоречие со строгой r -отличимостью состояний q и q' . Так как число различных пар состояний равно k , то длина входного слова x не превосходит k^2 .

Следствие. Если два состояния автомата A с k состояниями k -неотличимы, то они вообще неотличимы.

Доказанная теорема позволяет разбить множество состояний автомата A с k состояниями на попарно непересекающиеся классы неотличимых состояний. Использование этой теоремы при больших k затруднительно, ибо необходимо проверить все состояния на отличимость для всех входных слов длины k . Поэтому сначала упрощают автомат, исходя из особенностей задания его функций переходов и выходов, а затем уже применяют доказанную теорему.

10.5.1. Склеивание неразличимых состояний

Пусть задан автомат $A = (X, Y, Q, q_0, T, B)$ с выходом. Разобьем множество его состояний на попарно непересекающиеся классы неотличимых состояний: $Q = Q_0 \cup Q_1 \cup \dots \cup Q_p$, $Q_i \cap Q_j = \emptyset$; $i, j = 0, 1, \dots, p$; $i \neq j$. Для всякого i множество Q_i содержит попарно неотличимые состояния. Пусть q_i есть какой-либо представитель из Q_i . Построим автомат $A' = (X, Y, Q', q_0, T', B')$, где $Q' = \{Q_0, Q_1, \dots, Q_p\}$; Q_0 есть тот класс неотличимых состояний, который содержит q_0 ; $T'(Q_i, a) = \{T(q, a) : q \in Q_i\}$; $B'(Q_i, a) = B(q, a)$, $q \in Q_i$. Автоматы A и A' эквивалентны. В самом деле, для всякого входного слова $x = x(0)x(1)\dots x(r)$, если автомат A перерабатывает его в выходное слово $y(0)y(1)\dots y(r)$, проходя последовательность состояний $q(0), q(1), \dots, q(r)$, то автомат A' , проходя последовательность состояний $Q(0), Q(1), \dots, Q(r)$, $Q(0) = Q_0$, выдает выходное слово $y' = y'(0)y'(1)\dots y'(r)$, причем $y'(t) = B(Q(t), x(t)) = B(q(t), x(t)) = y(t)$, где $q(t) \in Q(t)$, т. е. $y' = y$.

Состояния автомата A' попарно отличимы, ибо если бы некоторые Q_i, Q_j были неотличимы, то их объединили бы на этапе построения автомата A в один класс.

10.5.2. Алгоритм минимизации автомата

1. Если автомат $A = (X, Y, Q, q_0, T, B)$ имеет состояния, недостижимые из начального состояния q_0 , то построим автомат $A' = (X, Y, Q', q_0, T', B')$, эквивалентный A , где Q' есть множество состояний автомата A , достижимых из начального

состояния q_0 ; функции T' и B' есть ограничения функций T и B на множество Q' . Если все состояния в Q достижимы из q_0 , то A' есть A .

2. Построим классы неотличимости состояний автомата A' и проведем операции их склеивания. Получим автомат A'' , эквивалентный автомату A' .

3. Автомат A'' приведенный и, следовательно, минимальный.

Пример 1. Пусть автомат A задан в табл. 10.5.

Таблица 10.5

	0	1	2	3	4	5	6	7
0	0,0	2,0	3,1	1,1	1,1	7,1	2,0	4,0
1	1,1	7,1	2,0	4,0	4,0	6,0	7,1	5,1

Множество столбцов, помеченных состояниями 1,6 и 3,4, одинаково; эти пары состояний заведомо склеиваются. В результате склеивания указанных пар состояний получаем автомат с шестью состояниями (табл. 10.6). Одинаковых столбцов в табл. 10.6 больше нет.

Таблица 10.6

	0	1	2	3	5	7
0	0,0	2,0	3,1	1,1	7,1	4,0
1	1,1	7,1	2,0	4,0	6,0	5,1

Строим теперь разбиение множества состояний автомата, приведенного в табл. 10.6, на попарно непересекающиеся классы неотличимых состояний. Классы эти строятся последовательно: сначала классы 1-неотличимых состояний, затем 2-неотличимых, 3-неотличимых и т. д.

Исходим из множества состояний $\{0,1,2,3,5,7\}$ автомата из табл. 10.6. Запишем это множество матрицей-строкой $M_0 = 012357$. Подаем на вход автомата слово $x = 0$. Автомат переходит в состояния 023173 и выдает на выходе 001110. Состояния 012357 разбиваются на два класса $\{0,1,7\}$ и $\{2,3,5\}$ состояний, не отличимых словом $x = 0$. Запишем эти два множества строкой 017.235, разделив точкой два полученных класса. Слову $x = 0$ сопоставим матрицу M_1 , приведенную на рис. 10.10.

Исходим теперь из последней строки состояний 017.235 в M_1 . На вход автомата подаем слово $x = 1$. Автомат переходит в состояния 175231 и выдает на выходе 111000. Классы состояний 017 и 235 словом $x = 1$ не отличимы. Слову $x = 1$ ставим в соответствие матрицу M_2 (см. рис. 10.10).

Слова 0 и 1 исчерпывают все слова длины 1, поэтому состояния классов 017 и 235 1-неотличимы.

Исходим из последней строки 017.235 в M_2 . На вход автомата подаем слово $x = 00$. Автомат перейдет в состояния 031123 и выдаст на выходе 011100. Класс 017 разбивается на два класса 0.17, а класс 235 – на два класса 2.35 состояний. Запишем эти множества строкой 0.17.2.35, разделив точками полученные классы. Сопоставим слову $x = 00$ матрицу M_3 (см. рис. 10.10).

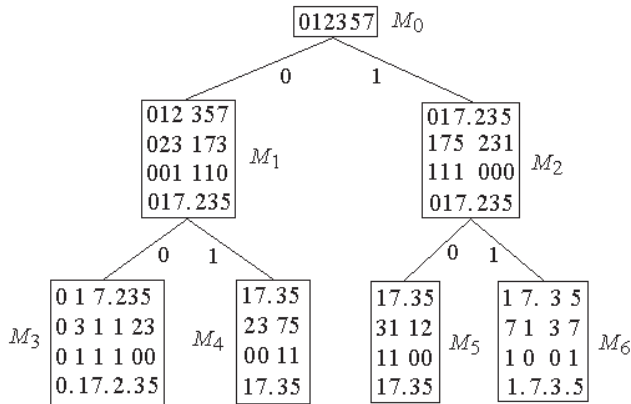


Рис. 10.10

Исходим из последней строки 0.17.2.35 в M_3 . Так как одноэлементные множества 0 и 2 дальнейшего подразделения не допускают, то достаточно взять строку состояний 17.35. На вход автомата подаем слово $x = 01$; сопоставим ему матрицу M_4 , (см. рис. 10.10). Разбиения классов 17 и 35 не произошло. Не произойдет нового разбиения и при подаче на вход слова $x = 10$ (матрица M_5).

Исходим из последней строки 17.35 матрицы M_5 . На вход автомата подаем слово $x = 11$; ему соответствует матрица M_6 , приведенная на рис. 10.10. Классы 17 и 35 разделились на одноэлементные множества и дальнейшего разбиения не допускают.

На этом построение классов неотличимости заканчивается. Все шесть состояний оказались попарно отличимыми словами (табл. 10.7).

Таблица 10.7

	1	2	3	5	7
0	00	0	0	0	00
1		0	0	0	11
2			00	00	0
3				11	0
5					0



На пересечении строки i и столбца j ($i < j$) стоит слово, которое отличает состояния i и j .

Автомат (см. табл. 10.6) является минимальным. Построение классов неотличимости состояний автомата из табл. 10.6, оформленное в виде дерева, приведено на рис. 10.10. Дерево строится от корня к листьям, сверху вниз, а в каждом ярусе узлов – слева направо.

Пример 2. Минимизировать автомат с функцией переходов и выходов, приведенных в табл. 10.8.

Таблица 10.8

	0	1	2	3	4	5
0	2,0	3,0	2,0	3,0	4,1	5,1
1	4,0	5,0	4,1	5,1	0,1	1,1

Одинаковых столбцов в табл. 10.8 нет. Построим разбиение множества состояний автомата на попарно непересекающиеся классы неотличимых состояний. Построение оформим в виде дерева (рис. 10.11).

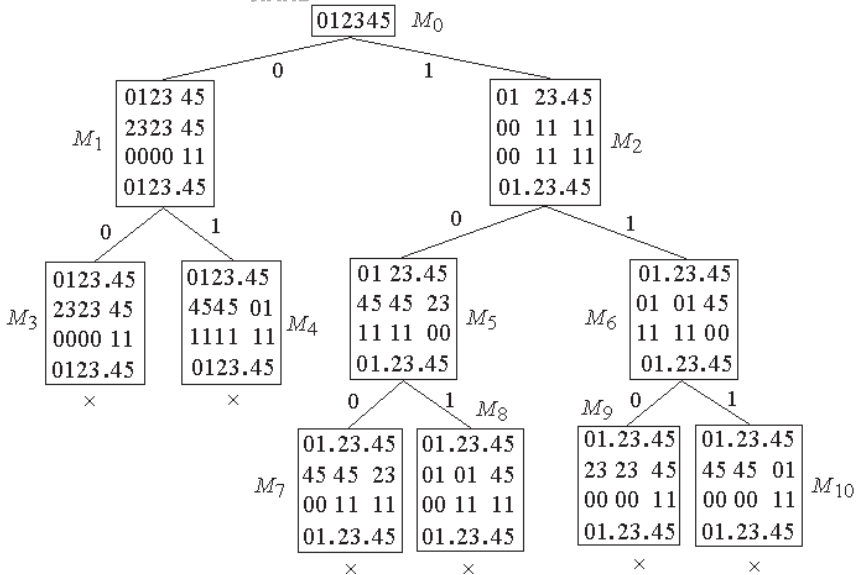


Рис. 10.11

Исходим из множества состояний автомата, заданного строкой $M_0 = 012345$. Слову $x = 0$, поданному на вход автомата, сопоставим матрицу M_1 так же, как это делали в предыдущем примере. Разделение множества состояний на одноэлементные множества не произошло. Затем так же, как это делали в предыдущем примере, последовательно строим матрицы $M_0, M_1, \dots, M_{10}$. Первые две строки матрицы M_i задают функцию $f_i: Q \rightarrow Q$ с графиком G_i . Если при построении матрицы M_i окажется, что $G_i \subseteq G_j$ для некоторого $j < i$, то эту матрицу помечаем крестом; на такой матрице дальнейший рост дерева прекращается: ответвлений от этого узла больше не будет. На рис. 10.11 таковы матрицы $M_3, M_4, M_7, M_8, M_9, M_{10}$, у которых $G_3 \subseteq G_1, G_4 \subseteq G_2, G_7 \subseteq G_5, G_8 \subseteq G_6, G_9 \subseteq G_3, G_{10} \subseteq G_2$. Строка 4 в последней построенной матрице M_{10} указывает классы неотличимости состояний $\{0,1\}, \{2,3\}, \{4,5\}$. Склеиваем их и получаем эквивалентный исходному минимальный автомат; он приведен в табл. 10.9. Входные слова, отличающие его состояния 0, 2 и 4, указаны в табл. 10.10.

Таблица 10.9

	0	2	4
0	2,0	2,0	4,1
1	4,0	4,1	0,1

Таблица 10.10

	2	4
0	1	1
2		0

10.5.3. Алгоритм разбиения множества состояний на классы неотличимых состояний

Пусть $A = (X, Y, Q, q_0, T, B)$ есть автомат, в котором $X = \{1, 2, \dots, |X|\}$, $Y = \{1, 2, \dots, |Y|\}$, $Q = \{1, 2, \dots, |Q|\}$, $q_0 = 1$. Разбиение множества состояний Q на классы неотличимых состояний оформим в виде дерева. Будем строить его от корня к листьям, сверху вниз, а в каждом ярусе узлов – слева направо.

0. В корне дерева помещаем матрицу-строку $M_0 = \{1, 2, \dots, |Q|\}$ состояний автомата.

1. Допустим, что матрицы $M_0, M_1, \dots, M_{i-1}$ построены. Переходим к построению матрицы M_i . Возможны следующие случаи:

1) из матрицы M_{i-1} исходит ребро, помеченное входным символом $a < |X|$. Тогда M_i строим следующим образом. Из последней строки матрицы M_{i-1} удаляем одноэлементные классы. Результат

$$q_{11} q_{12} \dots q_{1,n_1} \cdot q_{21} q_{22} \dots q_{1,n_2} \cdot \dots \cdot q_{r1} q_{r2} \dots q_{r,n_r}$$

делаем строкой 1 в M_i . Строки

$$\begin{aligned} &T(q_{11}, a+1) \cdot T(q_{12}, a+1) \cdot \dots \cdot T(q_{r,n_r}, a+1), \\ &B(q_{11}, a+1) \ B(q_{12}, a+1) \ \dots \ B(q_{r,n_r}, a+1) \end{aligned}$$

делаем в M_i строками 2 и 3 соответственно. Каждый класс $q_{m1} \dots q_{m,n_m}$ ($m = 1, 2, \dots, r$) строки 1 в M_i разбиваем на подклассы так, что два состояния q и q' из $\{q_{m1}, \dots, q_{m,n_m}\}$ попадут в один класс тогда и только тогда, когда $B(q, a+1) = B(q', a+1)$. Вновь полученные классы $q_{m1} \dots q_{m,l_1} \cdot q_{m,l_1+1} \dots q_{m,l_2} \cdot \dots \cdot q_{m,l_{s-1}} \dots q_{m,l_s}$ ($m = 1, 2, \dots, r$), отделенные друг от друга точками, запишем в одну строку и сделаем ее строкой 4 в M_i . Матрица M_i построена.

Первые две строки в M_i задают функцию $f_i : Q \rightarrow Q$ с графиком G_i . Если $G_i \subseteq G_v$ для некоторого v , $0 < v < i$, то матрицу M_i помечаем крестом.

Если строка 4 в M_i представляет собой разбиение на одноэлементные множества, то алгоритм работу заканчивает, и строка 4 в M_i дает искомое разбиение;

2) из матрицы M_{i-1} исходит ребро (с матрицей на конце), помеченное входным символом $|X|$. Возможны следующие случаи:

а) все построенные матрицы помечены крестом. Тогда алгоритм заканчивает работу, и строка 4 в матрице с наибольшим номером дает искомое разбиение;

б) не все построенные матрицы помечены крестом. Среди таких матриц выбираем матрицу с наименьшим номером, испускаем из нее ребро, помечаем его входным сигналом 1 и в концевой вершине этого ребра строим матрицу M_i так же, как это делали в случае 1.

2. Переходим к п. 1.

Замечание. При функционировании алгоритма не нужно хранить все дерево матриц M_i . Из всего дерева при его построении следует сохранять только множества ML листовых матриц этого дерева, а также получающиеся в процессе построения дерева матриц M_i множество SF функций f_i и текущее разбиение D множества состояний на классы неотличимости.

Приведенный алгоритм минимизации числа состояний автомата имеет достаточно большую вычислительную сложность. Поэтому исходный автомат предпочтительнее упростить сначала каким-либо другим более простым способом, например склеив состояния, столбцы для которых в таблице переходов (и выходов) автомата одинаковы.

Замечание. Порождение классов неотличимых состояний можно провести согласно следующему алгоритму. Если корень дерева (рис. 10.11) пометим начальным состоянием, а в каждый другой узел дерева поместим состояние, в которое перейдет автомат, если на его вход подать входной сигнал, помечающий ребро, входящее в этот узел, и соответствующий выходной сигнал, то в результате получим бесконечное дерево переходов состояний автомата и его выходов. Это дерево имеет конечное число попарно различных бесконечных поддеревьев. Можно ограничиться конечными поддеревьями с длиной ветвей, равной числу состояний автомата. Класс неотличимых состояний образуют все состояния, стоящие в корнях тех конечных деревьев, которые совпадают, если с их узлов удалить (стереть) пометки состояний, оставив пометки выходных символов.

Детерминизацию источника тоже можно организовать в виде дерева перехода состояний. В корень дерева помещается замыкание начального состояния источника. Если затем в некотором узле дерева помещено подмножество состояний S и если от этого узла отходит ребро, помеченное входным символом a , то концевой узел этого ребра помечается соответствующим макросостоянием. Построение дерева в узле прекращаем, если этот узел помечается макросостоянием, уже построенным ранее.

Аналогично в виде дерева перехода состояний можно реализовать алгоритм построения автомата для объединения, пересечения, конкатенации, итерации автоматных языков. Если соответствующие языки заданы источниками, то результирующий автомат можно получить детерминированным и минимальным.

10.5.4. Алгоритм проверки эквивалентности автоматов

Пусть A и B есть два автомата.

1. Минимизировать автоматы A и B и получить им эквивалентные минимальные автоматы C и D .
2. Если автоматы C и D имеют различное число состояний, то C и D не эквивалентны, а потому A и B не эквивалентны.
3. Если число состояний автоматов C и D одинаково и равно r , то выполнить следующее.

4. Для всех входных слов длины $k \leq r+1$ для $k = 1, 2, \dots, r+1$ последовательно строить множества MA_k и MB_k пар из входных и им соответствующих выходных слов автоматов C и D соответственно. Если на некотором шаге $k < r$ множества MA_k и MB_k различны, то автоматы C и D не эквивалентны, а потому A и B тоже не эквивалентны. Если $MA_{r+1} = MB_{r+1}$, то автоматы эквивалентны.



Автоматы и сверхъязыки

11.1. Макроавтоматы

Пусть X есть произвольный алфавит. Бесконечное слово $x = x(0)x(1)x(2)\dots$ в алфавите X называется сверхсловом. Сверхъязык есть некоторое множество сверхслов в алфавите X . Пусть X^∞ есть множество всех сверхслов в алфавите X . Пусть $\lim x(t)$ есть предельное множество сверхслова x при $t \rightarrow \infty$, т. е. $\lim x(t)$ есть множество всех тех букв в алфавите X , которые в сверхслове x встречаются бесконечное число раз. Если Q есть множество внутренних состояний некоторого автомата, то макросостояние есть подмножество $Q' \subseteq Q$. Пусть $P(Q)$ есть множество всех подмножеств множества Q .

Определение. (Детерминированный) макроавтомат (по МакНотону) есть система объектов $MA = (X, Q, q_0, T, F)$, где X, Q, q_0, T задаются как в обычном (детерминированном) автомате, а $F \subseteq P(Q)$ есть некоторое множество макросостояний. Вместо слова макроавтомат иногда будем говорить просто автомат, если из контекста ясно, о каком автомате идет речь.

Пусть на вход макроавтомата MA подается сверхслово $x = x(0)x(1)\dots$ из X^∞ . Автомат MA , начиная работу в начальном состоянии, выдает бесконечную последовательность состояний $q(0), q(1), \dots$ (или сверхслово $q = q(0)q(1)\dots$ состояний), называемую ходом автомата MA на сверхслове x . Пусть $\lim q(t)$ есть (предельное) множество всех состояний, которые в последовательности $q(t)$ встречаются бесконечно много раз. Макроавтомат MA допускает сверхслово x , если $\lim q(t) \in F$.

Поведение $Beh(MA)$ макроавтомата MA есть множество всех входных сверхслов x , допустимых автоматом MA . Макроавтомат MA представляет (допускает, определяет) сверхъязык L , если $L = Beh(MA)$. Сверхъязык L автоматом представим (допустим, определим), если существует макроавтомат MA , для которого $L = Beh(MA)$. Пусть множество $L(MA, Q, F)$ есть сверхъязык, допустимый макроавтоматом MA со множеством состояний Q и со множеством макросостояний F .

Макроисточник есть система $MS = (X, Q, Q_0, D, F)$, где X, Q, Q_0, D задаются как в источнике, а множество $F \subseteq P(Q)$ есть некоторое множество макросостояний.

Иногда вместо слова макроисточник будем говорить просто источник, если из контекста ясно, о каком источнике идет речь.

Сверхслово $q(0)q(1)q(2)\dots$ в алфавите состояний есть ход макроисточника MS на входном сверхслове $x = x(0)x(1)x(2)\dots$ (множество всех таких ходов обозначим через $Rn(MS, x)$), если при последовательной подаче входных символов $x(0), x(1), \dots$ (возможна подача и пустых символов) источник, исходя из начального состояния, проходит последовательность состояний $q(0), q(1), \dots$ согласно своей таблице переходов.

Макроисточник MS допускает сверхслово x , если существует ход $q(t)$ MS на x , для которого $\lim q(t) \in F$. В противном случае макроисточник MS сверхслово x отвергает (не допускает). Поведение макроисточника MS есть множество $Beh(MS)$ всех входных сверхслов, допустимых источником MS . Макроисточник MS представляет (допускает, определяет) сверхязык L , если $L = Beh(MS)$. Два макроисточника эквивалентны, если их поведения совпадают.

Операции над языками и сверхязыками

Введем некоторые операции над языками и сверхязыками. Пусть M есть язык, а L, L_1, L_2 есть сверхязыки в алфавите X .

Объединение $L_1 \cup L_2 = \{x \in X^* : x \in L_1 \vee x \in L_2\}$.

Пересечение $L_1 \cap L_2 = \{x \in X^* : x \in L_1, x \in L_2\}$.

Дополнение $CL = X^* - L$.

Сверхитерация (сильная итерация) языка M есть сверхязык $M^* = M \cdot M \cdot M \cdot \dots$

Конкатенация (умножение) языка M и сверхязыка L есть сверхязык $M \cdot L = \{x = y \cdot z \in X^* : y \in M, z \in L\}$.

Пусть множество $L(MS, Q_0, F)$ означает сверхязык, допустимый макроисточником MS со множеством начальных состояний Q_0 и множеством выделенных макросостояний F .

Теорема. Алгоритмически разрешимы (распознаваемы) следующие свойства макроавтоматов.

1. Представляемый макроавтоматом сверхязык пуст (проблема пустоты).
2. Представляемый макроавтоматом сверхязык бесконечен.

Доказательство. Достаточно рассмотреть случай, когда макроисточник $MS = (X, Q, Q_0, D, F)$ имеет одноэлементное множество F макросостояний, а именно $Q = \{q_0, q_1, \dots, q_k\}$, $Q' = \{q_1, q_2, \dots, q_s\}$, $s \leq k$, $F = \{Q'\}$. Докажем для этого автомата два утверждения теоремы.

1. Допустим, что существует сверхслово x , переводящее начальное состояние в макросостояние Q' , т. е. для некоторого хода $q(t)$ предельное макросостояние $\lim q(t) = Q'$. Тогда, во-первых, существует слово $y \in X^*$, переводящее начальное состояние q_0 в q_1 , и, во-вторых, для всякого $i = 1, 2, \dots, s-1$ существует слово $z_i \in X^*$, переводящее автомат из состояния q_i в состояние q_{i+1} , и слово $z_s \in X^*$, переводящее МА из q_s в q_1 , причем все промежуточные состояния лежат в Q' . Наличие или отсутствие таких слов проверяется для каждого автомата, в том числе и для МА.

2. Представимый макроавтоматом сверхязык либо пуст, либо бесконечен.

Теорема. Класс автоматов представимых сверхъязыков замкнут относительно булевых операций.

Доказательство. Пусть сверхъязыки L' и L'' представимы макроавтоматами $A' = (X, Q', q'_0, T', F')$ и $A'' = (X, Q'', q''_0, T'', F'')$ соответственно.

Пересечение $L' \cap L''$ представимо макроавтоматом $A = (X, Q, q_0, T, F)$, где $Q = Q' * Q''$; $q_0 = (q'_0, q''_0)$; $T((q', q''), a) = (T(q', a), T(q'', a))$; множество выделенных макросостояний F включает в себя любое подмножество из $F' \times F''$, в котором первая компонента лежит в F' , а вторая – в F'' .

Дополнение $X^{\omega} - L'$ представимо макроавтоматом $A = (X, Q', q'_0, T', P(Q) - F')$.

Объединение сверхъязыков выражается через их пересечение и дополнение. Макроавтомат для объединения можно построить, как и в случае пересечения, с той лишь разницей, что множество выделенных макросостояний F включает в себя любое подмножество из $F' \times F''$, в котором первая компонента лежит в F' или вторая лежит в F'' . На практике множество всех достижимых выделенных макросостояний для объединения можно получить, построив из начального состояния все пути до первого повтора состояния на каждом пути, выделив в каждом полученном пути цикл и отнеся все состояния этого цикла во множество выделенных состояний, если множество первых компонент этих состояний лежит в F' или множество вторых лежит в F'' . Аналогично поступаем и для пересечения.

Замечание. Детерминированный макроавтомат по Бюхи отличается от детерминированного макроавтомата по Макнотону лишь тем, что множество выделенных состояний по Бюхи $F \subseteq Q$. Допустимость по Бюхи отличается от допустимости по Макнотону лишь тем, что в допустимости по Бюхи $\lim q(t) \cap F \neq \emptyset$. Аналогично для макроисточников по Макнотону и по Бюхи. Отметим, что при допустимости по Бюхи макроисточник не допускает детерминизации.

11.2. Конкатенация языка и сверхъязыка

Теорема. Конкатенация конечноавтоматного языка и конечноавтоматного сверхъязыка есть конечноавтоматный сверхъязык.

Доказательство. Пусть $L = L(A, q_0, Q')$ есть автоматный язык, представимый автоматом $A = (X, Q, q_0, T, Q')$, и $SL = SL(MA, s_0, G)$ есть сверхъязык, представимый макроавтоматом $MA = (X, S, s_0, P, G)$. Тогда искомая конкатенация

$$L \cdot SL = \bigcup_{q' \in Q', F \in G} L(A, q_0, q') \cdot SL(MA, s_0, F).$$

Теорему достаточно доказать для случая, когда $L \cdot SL = L(A, q_0, q') \cdot SL(MA, q_0, F)$.

Пусть имеем один автомат A и неограниченный запас копий макроавтоматов MA . Принадлежность сверхслова $x = x(0)x(1)\dots$ сверхъязыку $L \cdot SL$ распознаем следующим образом. Запустим на сверхслове x автомат A . Если далее в некоторый момент t_1 автомат A впервые придет в состояние q' , то в этот момент запускаем копию 1 макроавтомата MA из неограниченного запаса копий макроавтоматов MA . Копия 1 макроавтомата MA запускается на сверхслове

$x[t_1, \infty) = x(t_1)x(t_1+1)\dots$. Если в момент t_2 автомат A во второй раз придет в состояние q' , то в этот момент запускаем копию 2 для MA . И так далее. Пусть $c_0(t)$ есть состояние автомата A в момент t , а $c_m(t)$ есть состояние копии m в момент t , если она в момент t активна (т. е. работает), и $c_m(t) = \wedge$, если копия m в момент t не активна (т. е. пребывает в запасе). Тогда сверхслово x принадлежит $L \cdot SL$ тогда и только тогда, когда

- 1) активна хотя бы одна копия макроавтомата MA ;
- 2) хотя бы одна из активных копий макроавтомата MA вырабатывает предельное макросостояние F . Поэтому

$$x \in L \cdot SL \leftrightarrow (\exists m)(\lim c_m(t) = F).$$

Заметим, что если для некоторого момента t' имеет место $c_m(t') = c_n(t')$, то $\forall t > t' c_m(t) = c_n(t)$; поэтому дальнейшая работа одной из этих копий становится излишней. Следовательно, копию m или копию n можно вернуть в запас, например копию с большим номером. Так как макроавтомат MA имеет $k+1$ состояний $s_0, s_0, \dots, s_k$, то понадобится $k+1$ копий макроавтомата MA .

Построим макроавтомат MB , допускающий сверхязык $L \cdot SL$. Состояния MB есть векторы $c = (c_0, c_1, \dots, c_{k+1})$ из множества $Q \times F' \times F' \times \dots \times F'$, где $F' = F \cup \{\wedge\}$, c_0 есть состояние автомата A ; $c_1, c_2, \dots, c_{k+1}$ есть состояния макроавтомата MA или знаки $\wedge$, при этом среди $c_1, c_2, \dots, c_{k+1}$ все состояния из MA попарно различны; $(q_0, \wedge, \dots, \wedge)$ есть начальное состояние. При воздействии буквы a из X на состояние $c(t) = (c_0(t), c_1(t), \dots, c_{k+1}(t))$ вектор $c(t+1) = (c_0(t+1), c_1(t+1), \dots, c_{k+1}(t+1))$ получим следующим образом. Компонета $c_0(t+1) = T(c_0(t), a)$. Каждое неравное $\wedge$ состояние $c_i(t)$ для $i \geq 1$ перейдет в состояние $c_i(t+1) = P(c_i(t), a)$ макроавтомата MA . Далее осуществим «чистку»: в векторе $c(t+1)$ из нескольких равных компонент (если таковые имеются) оставляем компоненту с меньшим номером, а остальные заменяем на $\wedge$. Если теперь $c_0(t+1) = q'$, то первая (по порядку) компонента $\wedge$ в $c(t+1)$ замещается на начальное состояние c_0 макроавтомата MA . Если же $c_0(t+1) \neq q'$, то все «пустые» компоненты $\wedge$ в $c(t+1)$ остаются пустыми. Функция переходов макроавтомата MB построена.

Множество выделенных макросостояний $\{F_1, \dots, F_h, \dots, F_l\}$ для MB содержит всякое макросостояние $F_h = \{(c_{0i}^j, c_1^j, \dots, c_i^j, \dots, c_{k+1}^j) : i = 1, 2, \dots, r \text{ при некотором } r\}$, для которого некоторая проекция на ось i ($1 \leq i \leq k+1$) : $\text{Pr}_i F_h = F$. При этом компоненты вне i в векторах из F_h безразличны.

Покажем, что сверхязык $L \cdot SL$ представим построенным макроавтоматом MB . Пусть ход $c(0)c(1)\dots c(t)\dots$ макроавтомата MB на сверхслове x из X^∞ таков, что $\lim c(t) = F_h$ при некотором $h \in \{1, 2, \dots, l\}$. Тогда $(\exists m)(1 \leq m \leq k+1 \ \& \ \lim c_m(t) = F)$. Это означает, что, включившись в некоторый момент, копия m для MA уже никогда не отключается (т. е. не переводится в запас). Далее копия m вырабатывает предельное макросостояние F . Поэтому входное сверхслово x допускает представление $x = x[0, t) \cdot x[t, \infty)$, причем $x[0, t) \in L$, $x[t, \infty) \in SL$ и потому $x \in L \cdot SL$.

Пусть $x \in L \cdot SL$. Покажем, что ход $c(0)c(1)\dots$ макроавтомата MB на x имеет предельное макросостояние из числа построенных, т. е. $\lim c(t) \in \{F_1, \dots, F_l\}$. Так

как $x \in L \cdot SL$, то сверхслово x допускает представление $x = x[0, t) \cdot x[t, \infty)$, причем $x[0, t) \in L$, $x[t, \infty) \in SL$. Так как $x[0, t) \in L$, то для вектора $c(t)$ имеем $(\exists p)(1 \leq p \leq k+1 \ \& \ c_p(t) = s_0)$. В ходе дальнейшей работы копия MA может перейти в запас, «передав» свою работу копии с меньшим номером. И так далее. Так как индекс p не может убывать неограниченно, то $(\exists m \leq p)(\lim c_m(t) = F)$. Тогда макроавтомат MB в компоненте m вектора состояний $c(t)$ даст предельное макросостояние F , и потому $\lim c(t) \in \{F_1, \dots, F_j\}$, т. е. сверхслово x допустимо построенным макроавтоматом MB . Теорема доказана.

11.3. Сверхитерация автоматных языков

Цель этого параграфа состоит в доказательстве того факта, что сверхитерация автоматного языка является автоматным сверхязыком.

Определение. Сверхслово $x = x(0)x(1)\dots$ в алфавите X устойчиво относительно автомата $A = (X, Q, q_0, T, Q')$, если существует бесконечно много хвостов $x[t_i, \infty)$, $i = 1, 2, \dots$, для которых:

- 1) все слова $x[0, t_i) \in L(A, q_0, Q')$, $i = 1, 2, \dots$;
- 2) автомат A склеивает каждый из хвостов $x[t_i, \infty)$, $i = 1, 2, \dots$, с исходным словом x .

Пусть L_{st} есть множество всех сверхслов, устойчивых относительно автомата A , допускающего язык L .

Лемма 1. Если язык L представим автоматом A , то $L_{st} \subseteq L^\infty$.

Доказательство. Пусть сверхслово $x = x(0)x(1)\dots \in L_{st}$. Тогда существует бесконечно много хвостов $x[t_i, \infty)$, $i = 1, 2, \dots$, для которых:

- 1) все слова $x[0, t_i) \in L$;
- 2) автомат A склеивает каждый из хвостов $x[t_i, \infty)$ с исходным сверхсловом x .

Запустим на x автомат A (рис. 11.1). В момент t'_1 запустим копию A_1 автомата A . Слово $x[0, t'_1) \in L$. Копия A_1 в некоторый момент $v_1 \geq t'_1$ склеивается с A . Так как хвостов $x[t_i, \infty)$ бесконечно много, то существует хвост $x[t'_2, \infty)$ с $t'_2 > v_1$. В момент t'_2 запустим копию A_2 автомата A . Слово $x[0, t'_2) \in L$, поэтому автомат A вместе со склеившейся с ним копией A_1 придет в выделенное состояние. Тогда слово $x[t'_1, t'_2) \in L$. Хвост $x[t'_2, \infty)$ склеится автоматом A с x в некоторый момент v_2 . Так как хвостов $x[t_i, \infty)$ бесконечно много, то найдется хвост $x[t'_3, \infty)$ с $t'_3 > v_2$. Слово $[0, t'_3) \in L$, поэтому автомат A вместе со склеившейся с ним копией A_2 в момент t'_3 придет в выделенное состояние. Тогда слово $x[t'_2, t'_3) \in L$. Продолжая эти рассуждения, мы покажем, что сверхслово $x = x[0, t'_1) \cdot x[t'_1, t'_2) \cdot x[t'_2, t'_3) \dots$ составлено из слов $x[t'_i, t'_{i+1})$ из L , т. е. $x \in L^\infty$.

Лемма 2. Если представимый автоматом A язык L замкнут относительно итерации (т. е. $L = L^*$), то $L^\infty = L \cdot L_{st}$.

Доказательство. Пусть язык L представим автоматом $A = (X, Q, q_0, T, Q')$, где $|Q| = k$ и $L^* = L$. По лемме 1 множество $L_{st} \subseteq L^\infty$, поэтому любое сверхслово z из L_{st} разбивается на бесконечное число слов из L .

Добавив к z спереди любое слово из L , снова получим сверхслово из L^∞ . Следовательно, $L \cdot L_{st} \subseteq L^\infty$.

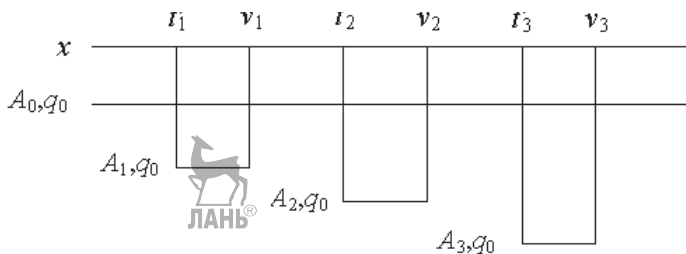


Рис. 11.1

Пусть теперь сверхслово $x \in L^\infty$. Для распознавания принадлежности сверхслова $x = x(0)x(1)\dots$ сверхязыку L^∞ поступим следующим образом. Запасем бесконечно много копий автомата A . Запустим на сверхслове x первую копию автомата A .

В каждый момент времени $t_0, t_1, t_2, \dots$, в который копия для A приходит в выделенное состояние, запускаем следующую (находящуюся в запасе) копию автомата A . Все начальные отрезки $x[0, t_i]$, $i = 1, 2, \dots$, входного сверхслова лежат в L . Если автомат A имеет k состояний, то достаточно иметь работающими (активными) не более k копий автомата A , ибо при большем их числе найдутся такие, которые пребывают в одном и том же состоянии. Из всех таких копий достаточно оставить активной только одну копию A , например имеющую меньший номер, т. е. копию, ранее всех включившуюся. Заметим, что первая копия не отключается никогда.

Так как $x \in L^\infty$, то сверхслово x составлено из бесконечного числа слов, принадлежащих L . Так как все начальные отрезки для x до мест соединений этих слов лежат в L , то существует бесконечно много моментов, в которые включаются все новые копии автомата A . Так как в любой момент активно не более k копий автомата A , то происходит бесконечное число склеиваний активных копий автомата A . Пусть несколько копий A склеивается с копией с номером l_1 , затем происходит склеивание с копией l_2 и т. д. В последовательности склеиваний копий A

$$(l_1, \dots); (l_2, \dots); (l_3, \dots); \dots; (l_i, \dots); \dots$$

скобка $(l_1, \dots)$ означает, что с копией l_1 автомата A склеились все копии этой скобки (находящиеся в том же состоянии, что и копия l_1). Номера копий в скобке $(l_i, \dots)$ возрастают по величине. В ряду чисел $l_1, l_2, \dots$ некоторые числа встречаются бесконечно много раз. Выберем среди них наименьшее число l . С копией l автомата A происходит бесконечно много склеиваний других копий; при этом копия l , включившись в некоторый момент t , уже никогда не отключается. Сверхслово x разбивается на два куска: $x[0, t)$ и $x[t, \infty)$, причем слово $x[0, t) \in L$, а сверхслово $x[t, \infty)$ таково, что существует бесконечно много хвостов $x[t'_1, \infty)$, $x[t'_2, \infty)$, ... этого сверхслова, для которых:

- 1) все слова $x[t'_i, t'_i] \in L$, $i = 1, 2, \dots$;
- 2) автомат A склеивает каждый из хвостов $x[t'_i, \infty)$ со сверхсловом $x[t'_i, \infty)$.

Следовательно, $x[0, t'_1] \in L$, $x[t'_1, \infty) \in L_{st}$, т. е. $x \in L \cdot L_{st}$.

Показано, что $L^\infty \subseteq L \cdot L_{st}$, что вместе с ранее доказанным обратным включением дает искомое равенство.

Лемма 3 (об устойчивости). Существует алгоритм, который по любому конечному автомату A строит макроавтомат MA , поведение которого есть множество всех сверхслов, устойчивых относительно A .

Доказательство. Пусть автомат $A = (X, Q, q_0, T, Q')$ с k состояниями допускает язык L . Пусть L_{st} есть множество всех сверхслов, устойчивых относительно автомата A . Состояниями автомата MA являются векторы $c = (c_1, c_2, \dots, c_{k+1})$, где c_1 есть состояние автомата A ; $c_{k+1} \in \{\wedge, *\}$; $c_i \in Q \cup \{\wedge\}$, $i = 2, 3, \dots, k$; при этом те из $c_2, c_3, \dots, c_{k+1}$, которые являются состояниями автомата A , попарно различны. Вектор c всегда имеет в запасе пустую компоненту $\wedge$. Множество S состояний для MA построено. Вектор $s_0 = (q_0, \wedge, \dots, \wedge)$ есть начальное состояние макроавтомата MA . Функция M переходов для MA строится следующим образом. Пусть при состоянии $c(t) = (c_1(t), \dots, c_{k+1}(t))$ на вход в MA поступает входной символ a . Построим вектор $c(t+1) = (c_1(t+1), \dots, c_{k+1}(t+1))$, положив $c_1(t+1) = T(c_1(t), a)$. Для $i = 2, 3, \dots, k+1$ при $c_i(t) \neq \wedge$ полагаем $c_i(t+1) = T(c_i(t), a)$, а при $c_i(t) \in \{\wedge, *\}$ полагаем $c_i(t+1) = \wedge$. Вектор $c'(t+1)$ (и тогда $M(c(t), a) = c'(t+1)$) сделаем результатом следующих операций.

1. Если хотя бы одна координата из $c_2(t+1), \dots, c_{k+1}(t+1)$ совпадает с $c_1(t+1)$, то $c'(t+1) = (c_1(t+1), \wedge, \dots, \wedge, *)$.

2. Если ни одна из компонент $c_2(t+1), \dots, c_{k+1}(t+1)$ не есть $c_1(t+1)$, то проводим следующую «чистку» вектора $c(t+1)$: из всех одинаковых компонент в $c_2(t+1), \dots, c_{k+1}(t+1)$ оставляем одну с меньшим номером. Такую «чистку» проводим до тех пор, пока все состояния в получившемся векторе не станут попарно различными.

Выделенным для MA является множество макросостояний $H = \{H_1, H_2, \dots, H_i, \dots, H_l\}$, где H_i есть любое множество векторов c , содержащих звездчатое состояние $(q, \wedge, \dots, \wedge, *)$.

Макроавтомат $MA = (X, S, s_0, M, H)$ построен.

На практике вектор состояний экономнее строить без пустых символов, изменяя длину вектора, когда это необходимо.

Утверждение 1. Сверхслово $x \in L_{st}$ тогда и только тогда, когда звездчатые состояния в ходе макроавтомата MA встречаются бесконечно много раз.

Доказательство. Пусть в ходе MA на x первое звездчатое состояние есть $c(t_1)$. Это значит, что в момент t_1 одна из копий автомата A , включившись в момент между 0 и t_1 , склеилась с первой копией. Если $c(t_2)$ есть второе звездчатое состояние, то это значит, что некоторая копия для A , включившись в момент между t_1 и t_2 , склеилась в момент t_2 с первой копией автомата A . И так далее. Следовательно, существует бесконечно много хвостов $x[t_i, \infty)$, $i = 1, 2, \dots$, для которых:

1) все слова $x[0, t_i] \in L$;

2) автомат A склеивает каждый из хвостов $x[t_i, \infty)$ с исходным сверхсловом x . Каждая склейка означает появление звездчатого состояния в ходе MA на x .

Утверждение 1 доказано.

Утверждение 2. $L_{st} = Beh(MA)$.

Доказательство. Справедливы следующие высказывания.

1. Сверхслово $x \in L_{st}$.
2. Звездчатое состояние в ходе MA на x встречается бесконечно много раз.
3. Предельное макросостояние в ходе MA на x содержит звездчатое состояние.
4. Для $c(t) \in Rn(MA, x)$ предельное множество $\lim c(t) \in H$.
5. $x \in Beh(MA)$.

Следовательно, $L_{st} = Beh(MA)$.

Утверждение 2 установлено. Лемма 3 доказана.

Теорема (о сверхитерации). Если L есть конечноавтоматный язык, то L^∞ есть конечноавтоматный сверхязык.

Доказательство. Пусть L есть автоматный язык, представимый автоматом A с начальным состоянием q_0 . Так как $L^\infty = (L^*)^\infty$, то можно считать, что множество L замкнуто относительно итерации. Если этого нет, то вместо L следует взять язык L^* и представляющий его автомат. По лемме 2 сверхязык $L^\infty = L \cdot L_{st}$. По лемме 3 сверхязык L_{st} автоматно представим. По теореме из предыдущего параграфа автоматно представим сверхязык $L \cdot L_{st}$. Следовательно, сверхязык L автоматно представим.

Замечание. По автомату A макроавтомат MA , представляющий сверхязык L^∞ , строится эффективно.

11.4. Детерминизация макроисточника

Теорема. Существует алгоритм, который по любому заданному макроисточнику строит эквивалентный ему детерминированный макроавтомат.

Доказательство. Пусть $MA = (X, Q, Q_0, D, F)$ есть макроисточник; $F \subseteq P(Q)$ и множество $F = \{F_1, \dots, F_k\}$. Тогда представимый макроисточником MA сверхязык $SL = L(MA, Q_0, F) = \bigcup_{q_0 \in Q_0, H \in F} L(MA, q_0, H)$.

Пусть $H = \{q_1, q_2, \dots, q_s\}$ и $MA_1 = (X, Q, q_0, D, H)$ есть макроисточник, полученный из MA с сохранением в MA только состояний из H и ребер, им инцидентных (т. е. этим состояниям принадлежащих). Макроисточник MA_1 можно рассматривать как источник A_1 , допускающий или не допускающий конечный язык обычным образом. Макроисточник MA_1 представляет сверхязык SL_1 , содержащий все сверхслова x из X^* , на которых любой ход MA_1 содержит состояния только лишь из H . Если множество всех сверхслов $SL_1 = L(MA_1, q_1, H)$ пусто, то исходный сверхязык пуст. Если нет, то

$$SL_1 = (L(A_1, q_1, q_2) \cdot L(A_1, q_2, q_3) \cdot \dots \cdot L(A_1, q_{s-1}, q_s) \cdot (L(A_1, q_s, q_1))^\infty,$$

и тогда представимое в макроисточнике MA множество сверхслов

$$SL = L(MA, q_0, H) = \begin{cases} L(A, q_0, q_1) \cdot SL_1, & \text{если } L(A, q_0, q_1) \neq \emptyset, \\ SL_1, & \text{если } L(A, q_0, q_1) = \emptyset \text{ и } q_1 = q_0, \\ \emptyset & \text{в остальных случаях.} \end{cases}$$

Далее, применяя теорему о свёрхитерации языка, представимого конечным автоматом, и теорему о конкатенации языка, представимого автоматом, со свёрхязыком, представимым макроавтоматом, строим искомый детерминированный макроавтомат, представляющий свёрхязык SL .

Следствие. Класс свёрхязыков, представимых макроисточниками, замкнут относительно дополнения.

Общерегулярные свёрхязыки

Определение. 1. Если язык A (множество конечных слов) регулярен, то свёрхязык A^∞ общерегулярен.

2. Если язык A регулярен, а свёрхязык B общерегулярен, то свёрхязык $A \cdot B$ общерегулярен.

3. Если свёрхязыки A и B общерегулярны, то свёрхязык $A \cup B$ общерегулярен.

Теорема. Всякий свёрхязык общерегулярен тогда и только тогда, когда он автоматно представим.

Доказательство. *Синтез.* Автоматность общерегулярного свёрхязыка доказывается индукцией по его построению.

Анализ. Пусть макроисточник $MA = (X, Q, q_0, T, F)$ допускает свёрхязык L . Достаточно показать общерегулярность L для одноэлементного множества $F = \{q_1, \dots, q_r\}$. Пусть источник $A = (X, Q, q_0, T, \{q_1\})$ допускает множество слов, переводящих A_1 из q_0 в q_1 . Построим источники $A_{1 \rightarrow 2}, A_{2 \rightarrow 3}, \dots, A_{r-1 \rightarrow r}, A_{r-1}$ следующим образом. Источник $A_{i \rightarrow j} = (X, Q, q_i, D, \{q_j\})$, где $(q, a, q') \in D \leftrightarrow (q, a, q') \in T \& q \in F \& q' \in F$. Тогда $A_{i \rightarrow j}$ допускает такие и только такие слова, которые переводят A из q_i в q_j и A не проходит через состояния вне F . Тогда

$$Beh(A) = Beh(A_1) \cdot Beh(A_{1 \rightarrow 2}) \cdot Beh(A_{2 \rightarrow 3}) \cdot \dots \cdot Beh(A_{r-1 \rightarrow r}) \cdot \dots \cdot Beh(A_{r-1})^\infty.$$

Поэтому множество $Beh(MA)$ общерегулярно. Теорема доказана.

Построение общерегулярного множества свёрхслов, представляющего дополнение свёрхязыка $Beh(MA)$, представимого макроисточником $MA = (X, Q, Q_0, D, F)$, можно осуществить следующим образом. Свёрхслово $w \in X^\infty$ принадлежит $Beh(MA) \leftrightarrow (\forall r)_{r \in Rn(MA, w) \& r(0) \in Q_0} \lim(r) \in F$.

Пусть дополнение $Pow(Q)$ есть F и для простоты состоит из единственного макросостояния $G = \{q_1, \dots, q_r\}$. Построим источники $A_1, A_{1 \rightarrow 2}, A_{2 \rightarrow 3}, \dots, A_{r-1 \rightarrow r}, A_{r-1}$ следующим образом. A_1 строится по $A'_1 = (X, Q, Q_0, D, \{q_1\})$ так, что для всякого хода A'_1 на входном конечном слове заключительным оказалось бы состояние q_1 . Для источника $A_{i \rightarrow j} = (X, Q, q_i, T, \{q_j\})$ положим $(q, a, q') \in T \leftrightarrow (q, a, q') \in D \& q \in G \& q' \in G$. Тогда $A_{i \rightarrow j}$ допускает такие и только такие слова, которые переводят A из q_i в q_j и при этом A не проходит через состояния вне G . Тогда дополнение

$$X^\infty - Beh(MA) = Beh(A_1) \cdot Beh(A_{1 \rightarrow 2}) \cdot Beh(A_{2 \rightarrow 3}) \cdot \dots \cdot Beh(A_{r-1 \rightarrow r}) \cdot Beh(A_{r-1})^\infty,$$

т. е. дополнение $X^\infty - Beh(MA)$ общерегулярно.



Проекция сверхслова с одного алфавита на другой осуществляется побуквенно, как и в случае проекции конечного слова. Проекция сверхъязыка осуществляется проекцией каждого его сверхслова.

Теорема. Класс автоматов представимых сверхъязыков замкнут относительно проекции.

Доказательство. Пусть макроавтомат $MA = (X, Q, q_0, T, F)$ представляет сверхъязык $SL \subseteq X^\infty$ и функция $f: X \rightarrow Y$ осуществляет проекцию сверхъязыка SL с алфавита X на алфавит Y . Тогда макроисточник $MAf = (X, Q, Q_0, D, F)$, где $(q, f(a), q') \in D \leftrightarrow (q, a, q') \in T$, представляет сверхъязык $f(SL)$. По макроисточнику $MA f$ можно построить детерминированный макроавтомат, представляющий сверхъязык $f(SL)$.



Глава 12

Проблема униформизации

12.1. Языки и операторы



Пусть зафиксированы входной X и выходной Y алфавиты. Всякий конечный автомат с выходом реализует некоторый словарный (сверхсловарный) оператор $\mathbb{O}(x) = y$, определенный на множестве слов (сверхслов) в алфавите X и принимающий значения во множестве слов (сверхслов) в алфавите Y . График оператора $\mathbb{O}$ есть множество $G = \{(x, y) : \mathbb{O}(x) = y\}$. В словарном операторе пара $(x, y) \in X^* \times Y^*$, а в сверхсловарном операторе пара $(x, y) \in X^\infty \times Y^\infty$.

Поведение автоматов без выхода с выделенными состояниями определяют в терминах представимости языков (сверхязыков). Поведение автоматов с выходом определяется в терминах реализации (т. е. представимости) операторов. Между обоими понятиями представимости имеется естественная связь, основанная на сводимости друг к другу свойств множеств и свойств функций. А именно произвольное множество, рассматриваемое как подмножество некоторого универсального множества (универсума), однозначно определяется своей характеристической функцией. Произвольная функция, в свою очередь, может рассматриваться как множество точек ее графика. В нашем случае в роли множеств выступают языки (сверхязыки), а в роли функций – операторы.

С каждым подалфавитом $Y' \subseteq Y$ оператор $\mathbb{O}$ ассоциирует язык $\mathbb{O}^{-1}(Y') = \{x = x(0)x(1)\dots x(k) \in X : \mathbb{O}(x) = y = y(0)y(1)\dots y(k) \in (Y')^*\}$. Скажем тогда, что оператор $\mathbb{O}$ *представляет* язык $\mathbb{O}^{-1}(Y')$ множеством выходов (т. е. выходными буквами из Y'). Оператор $\mathbb{O}$ допускает все слова из $\mathbb{O}^{-1}(Y')$ и отвергает все другие слова.

Аналогичные построения можно выполнить относительно сверхязыков. А именно с каждым подмножеством $Y' \subseteq Y$ оператор $\mathbb{O}$ индуцирует сверхязык $\mathbb{O}^{-1}(Y') = \{x = x(0)x(1)\dots \in X^\infty : \mathbb{O}(x) = y = y(0)y(1)\dots \in Y^\infty, \lim y(t) = Y'\}$. Скажем тогда, что оператор $\mathbb{O}$ *представляет* сверхязык $\mathbb{O}^{-1}(Y')$, т. е. допускает все сверхслова из $\mathbb{O}^{-1}(Y')$ и отвергает все слова вне $\mathbb{O}^{-1}(Y')$.

Автомат $A = (X, Y, Q, q_0, T, B)$ с выходом представляет язык (сверхязык) L выходами из Y' и Y , если оператор $\mathbb{O}$, соответствующий этому автомату, пред-

ставляет этот язык (сверхязык) выходами из алфавита Y' .

Теорема. Класс языков (сверхязыков), представимых в конечных автоматах с выходом, совпадает с классом языков (сверхязыков), представимых в конечных автоматах без выхода.

Доказательство. С произвольным автоматом без выхода $A = (X, Q, q_0, T, F)$, где $Q = \{q_0, q_1, \dots, q_k\}$, $F = \{q_{i1}, \dots, q_{ir}\}$, можно ассоциировать автомат с выходом $A' = (X, Y, Q, q_0, T, B)$, где $Y = \{y_0, y_1, \dots, y_k\}$, а функция выхода B строится следующим образом. Если $T(q_i, x) = q_j$, то $B(q_i, x) = y_j$. Язык $Beh(A)$ совпадает с языком, представимым автоматом A' множеством выходов $Y' = \{y_{i1}, \dots, y_{ir}\}$.

Пусть макроавтомат без выхода $A = (X, Q, q_0, T, F)$, где $Q = \{q_0, q_1, \dots, q_k\}$, $F = \{\{q_{i1}, \dots, q_{ir}\}, \dots, \{q_{j1}, \dots, q_{jl}\}\} = P(Q)$, представляет сверхязык $Beh(A)$. Построим автомат с выходом $A' = (X, Q, q_0, T, B)$, где $Y = \{y_0, y_1, \dots, y_k\}$, а функция выходов B строится так же, как и в предыдущем случае. Сверхязык $Beh(A)$ представим автоматом с выходом A' макромножеством выходов $Y' = \{\{y_{i1}, \dots, y_{ir}\}, \dots, \{y_{j1}, \dots, y_{jl}\}\}$.

Пусть автомат Мура с выходом $A = (X, Y, Q, q_0, T, B)$ представляет язык $Beh(A)$ множеством $Y' \subseteq Y$. Положим $F = \{q \in Q : \exists y \in Y' (B(q) = y)\}$. Тогда автомат без выхода $A' = (X, Q, q_0, T, F)$ представляет тот же язык $Beh(A)$.

Если теперь автомат Мура с выходом $A = (X, Y, Q, q_0, T, B)$ представляет сверхязык $\mathbb{O}^{-1}(Y')$ некоторым макромножеством $Y' = \{\{y_{i1}, \dots, y_{ir}\}, \dots, \{y_{j1}, \dots, y_{jl}\}\} = \{Y_i, \dots, Y_j\} \in P(Y)$, то при

$$F = \{F_i, \dots, F_j\}, \text{ где}$$

$$F_i = \{q \in Q : \exists y \in Y_i (B(q) = y)\},$$

...

$$F_j = \{q \in Q : \exists y \in Y_j (B(q) = y)\},$$

автомат $A = (X, Q, q_0, T, F)$ представляет тот же самый язык.

Замечание. Алгоритм минимизации автоматов без выхода получается сочетанием алгоритма минимизации автоматов с выходом с доказанной теоремой.

Операторы можно рассматривать и вне понятия конечного автомата. Словарный оператор можно рассматривать как функцию $\mathbb{O}: X^* \rightarrow Y^*$, а сверхсловарный оператор – как функцию $\mathbb{O}: X^\infty \rightarrow Y^\infty$.

Сверхсловарный оператор $\mathbb{O}: X^\infty \rightarrow Y^\infty$ есть оператор без предвосхищения (его еще называют ограниченно детерминированным оператором), если для всяких входных сверхслов

$$x' = x'(0)x'(1) \dots x'(t) \dots$$

$$x'' = x''(0)x''(1) \dots x''(t) \dots$$

соответствующие им входные слова

$$\mathbb{O}(x') = y' = y'(0)y'(1) \dots y'(t) \dots,$$

$$\mathbb{O}(x'') = y'' = y''(0)y''(1) \dots y''(t) \dots$$

таковы, что для всякого t равенство

$$x'(0)x'(1) \dots x'(t) = x''(0)x''(1) \dots x''(t)$$

влечет равенство

$$y'(0)y'(1) \dots y'(t) = y''(0)y''(1) \dots y''(t).$$

В противном случае оператор $\oplus$ является оператором с предвосхищением.

Словарный оператор $\oplus: X^* \rightarrow Y^*$ есть оператор без предвосхищения, если:

- 1) равенство $\oplus(x) = y$ влечет равенство длин слов x и y ;
- 2) для всяких входных слов x', x'' и для всякого t равенство $x'(0)x'(1) \dots x'(t) = x''(0)x''(1) \dots x''(t)$ влечет равенство $\oplus(x') = y' = y'(0)y'(1) \dots y'(t) = y''(0)y''(1)y''(2) \dots y''(t) = y'' = x''$;
- 3) оператор $\oplus$, применимый к слову x , применим и ко всякому его начальному отрезку.

В противном случае словарный оператор $\oplus$ есть оператор с предвосхищением.

В операторе без предвосхищения выходная буква в момент t не зависит от входных букв в момент $t' > t$. В операторах с предвосхищением это условие может не выполняться. В конечных автоматах реализуются только операторы без предвосхищения.

Утверждение. Если оператор $\oplus$ реализуем в некотором конечном автомате, то его график представим конечным автоматом.

Доказательство. Пусть для определенности $\oplus$ есть сверхсловарный оператор, реализуемый автоматом $A = (X, Y, Q, q_0, T, B)$. Если диаграмму для A (где каждое ребро снабжено входом и выходом) рассматривать как макроисточник в алфавите $X \times Y$, то график оператора A совпадает с поведением этого макроисточника, имеющего в качестве множества выделенных макросостояний множество всех подмножеств множества состояний исходного автомата. Утверждение доказано.

Поставим теперь следующий вопрос. Если график некоторого оператора $\oplus: X^\infty \rightarrow Y^\infty$ представим в некотором макроавтомате, то можно ли реализовать оператор $\oplus$ в подходящем конечном автомате? Ответ отрицателен. Это показывает следующий пример.

Пусть сверхсловарный оператор $\oplus: X^\infty \rightarrow Y^\infty$ с $X = Y = \{0, 1\}$ сопоставляет сверхслову $00\dots 0\dots$ (сплошь из нулей) то же самое сверхслово $00\dots 0\dots$, а всякому другому сверхслову сверхслово $11\dots 1\dots$ (сплошь из единиц). Оператор $\oplus$ является оператором с предвосхищением, и потому он не реализуем с помощью конечного автомата. Между тем график оператора $\oplus$ представим в макроисточнике

$$MS = (X \times Y, Q, Q_0, D, F),$$

где $Q = \{q_0, q_1, q_2\}$; $Q_0 = \{q_0, q_1\}$;

$$D = \{(q_0, 00, q_0), (q_1, 01, q_1), (q_1, 11, q_2), (q_2, 11, q_2), (q_2, 01, q_2)\};$$

$$F = \{\{q_0\}, \{q_2\}\}.$$

Ранее показано, что классы сверхслов, представимых в макроисточниках и макроавтоматах, совпадают.

Униформизация

Пусть дан сверхязык SL в алфавите $X \times Y$. Сверхсловарный оператор $\mathbb{O}: X^\infty \rightarrow Y^\infty$ униформизирует сврхязык SL , если для всякого $x \in X^\infty$ равенство $\mathbb{O}(x) = u$ влечет $(x, u) \in SL$.

Можно говорить об униформизации сверхязыка SL оператором $Y^\infty \rightarrow X^\infty$. Понятие униформизации можно сформулировать и для сверхязыков в алфавите $X_1 \times X_2 \times \dots \times X_n$ по любой переменной.

Понятие униформизации есть одно из важнейших понятий теории множеств. Очень важно оно и в теории конечных автоматов. Проблема униформизации конечноавтоматного сверхязыка состоит в следующем. Пусть задан макроавтомат A , представляющий некоторый сверхязык SL в алфавите $X \times Y$. Требуется выяснить, существует ли сверхсловарный конечноавтоматный оператор $\mathbb{O}$, униформизирующий сверхязык SL ; если да, то построить конечный автомат с выходом, реализующий такой оператор.

Покажем, что проблема униформизации конечноавтоматного сверхязыка решается положительно. Решение этой задачи дадим в терминах предложенной Макнотонем игровой интерпретации, в той редакции, которую предложил для этого Б. А. Трахтенброт.

12.2. Игры

Будем рассматривать игры с полной информацией. Назовем их просто играми. Игре присущи следующие особенности.

1. В игре участвуют два игрока, называемые белыми и черными. Первый ход делают белые; черные отвечают своим ходом. Затем ходы чередуются: белые, черные, белые, черные и т. д., образуя партию игры.

2. Ход белых есть выбор одной буквы из алфавита $X = \{x_1, \dots, x_m\}$. Ход черных есть выбор одной буквы из алфавита $Y = \{y_1, \dots, y_n\}$.

3. При каждом ходе очередной игрок знает все предыдущие ходы данной партии (полная информация).

4. Для каждой партии имеет место либо выигрыш белых, либо выигрыш черных.

Партия может быть конечной и бесконечной. В конечной партии партия прекращается после конечного числа шагов. Конечную партию можно рассматривать как слово в алфавите $X \times Y$:

$$\begin{aligned} &x(0)x(1) \dots x(t), \\ &y(0)y(1) \dots y(t), \end{aligned}$$

или как пару слов $x[0, t], y[0, t]$ отдельно ходов белых и ходов черных. Длина партии есть t .

Бесконечная партия продолжается неограниченно.

Бесконечную партию можно рассматривать как сверхслово

$x(0)x(1) \dots x(t) \dots$
 $y(0)y(1) \dots y(t) \dots$



в алфавите $X \times Y$ или как пару сверхслов $x[0, \infty)$, $y[0, \infty)$ отдельно ходов белых и ходов черных.

Игра конечна, если для некоторого числа p любая партия не длиннее p . В противном случае игра бесконечна.

Всякую конечную игру можно задать в виде дерева, ребра которого помечены буквами из $X \times Y$, причем ребра, выходящие из одной вершины, по-разному помечены. Концевые вершины (листья) разбиты на два класса: класс вершин, помеченных знаком «+» (это выигрыш белых), и класс вершин, помеченных знаком «-» (это выигрыш черных). В корне дерева помещена пешка. Белые выбирают любую букву $x(0)$ из X ; черные в ответ выбирают любую букву $y(0)$ из Y . Пешка перемещается в вершину по ребру, помеченному парой $(x(0), y(0))$. Далее белые выбирают одну X -букву (т. е. букву из X) $x(1)$, на что черные отвечают выбором одной Y -буквы $y(1)$. Пешка перемещается в вершину по ребру, помеченному парой $(x(1), y(1))$. И так далее. Партия заканчивается, если пешка окажется в концевой вершине; ее пометка указывает, кто выиграл. Множество партий, выигрышных для белых, есть язык в алфавите $X \times Y$.

На рис. 12.1 приведена конечная игра с $X = \{0, 1\}$, $Y = \{a, b\}$.

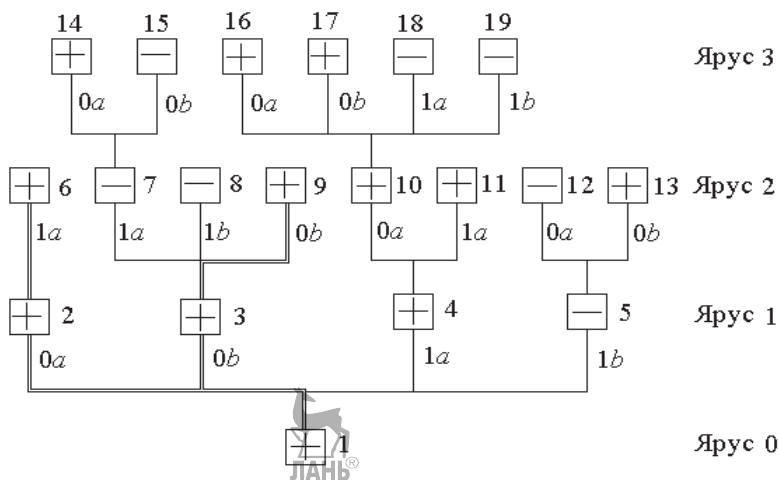


Рис. 12.1

Для всякой конечной игры можно определить, какая из сторон имеет выигрыш, и указать последовательность ходов, приводящую к выигрышу. Покажем, как это делается для игры, дерево которой изображено на рис. 12.1. Просматриваем все поддеревья, имеющие корень в предпоследнем ярусе 2; это поддеревья с корнем в вершинах 7 и 10. Из вершины 7 выигрывают черные,

ибо при единственно возможном ходе белых 0 черные отвечают выбором b и выигрывают. Вершину 7 в знак выигрыша черных помечаем знаком «-». Из вершины 10 выигрывают белые, выбирая ход 0; любой ответ черных приводит их к проигрышу.

Вершину 10 в знак выигрыша белых помечаем знаком «+». Далее просматриваем все поддеревья с корнем в ярусе 1. Из вершины 2 выигрывают белые; из вершины 3 ходом 0 выигрывают белые; из вершины 4 произвольным ходом выигрывают белые; из вершины 5 при единственно возможном ходе белых 0 ответом выигрывают черные. Вершины 2, 3, 4 помечаем знаком «+», а вершину 5 – знаком «-». Из вершины 1 ходом 0 выигрывают белые; помечаем ее знаком «+». В этой игре белые начинают и выигрывают (в любой партии). Соответствующие ходы отмечены двойными линиями. В этой игре белые имеют выигрывающую стратегию (т. е. белые начинают и выигрывают).

В бесконечных играх все партии бесконечны. Такую игру можно изобразить бесконечным деревом (рис. 12.2), из каждой вершины которого исходят mn ребер, помеченных парами (x, y) из $X \times Y$, где $X = \{x_1, \dots, x_m\}$, $Y = \{y_1, \dots, y_n\}$.

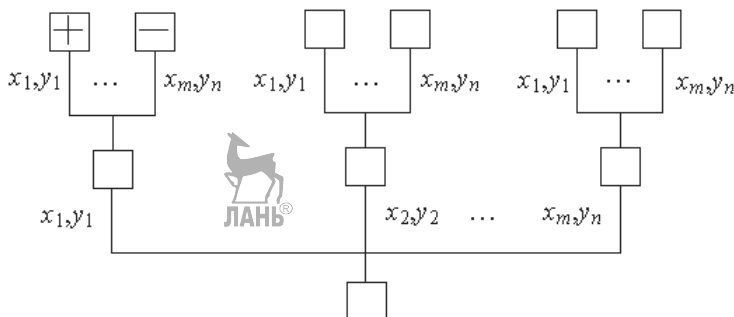


Рис. 12.2

Игра есть дерево. Партия есть путь в дереве. В этом бесконечном дереве отмечается множество путей (партий), выигрышных для белых. Остальные пути (партии) выигрышны для черных. Таким образом, различные игры при фиксированных X и Y отличаются только множествами партий (путей), выигрышных для белых. Это множество есть некоторый сверхязык в алфавите $X \times Y$. С другой стороны, всякий сверхязык L из $(X \times Y)^\infty$ задает бесконечную игру, в которой множество партий (путей) из L выигрышно для белых.

Игры с конечным числом состояний

Пусть $A = (X \times Y, Q, q_0, T, F)$ есть макроавтомат, где $F \subseteq P(Q)$. Игра с конечным числом состояний (автоматная игра) осуществляется следующим образом. Пешка помещается в вершину $q(0) = q_0$ диаграммы автомата A . Белые называют свой первый ход $x(0)$ из X ; черные отвечают выбором $y(0)$ из Y . Пешка перемещается в вершину

$$q(1) = T(q(0), (x(0), y(0))).$$

Далее белые выбирают $x(1)$ из X , а черные отвечают выбором $y(1)$ из Y . Пешка перемещается в вершину

$$q(2) = T(q(1), (x(1), y(1))).$$

Такой обмен ходами продолжается неограниченно:

$$x(0) \ x(1) \ \dots \ x(t) \ \dots$$

$$y(0) \ y(1) \ \dots \ y(t) \ \dots;$$

в результате этого пешка последовательно перемещается в вершины

$$q(0) \ q(1) \ \dots \ q(t) \ \dots,$$

образуя сверхслово состояний q из Q^* . Белые выигрывают, если $\lim q(t)$ лежит в F , и проигрывают в противном случае.

12.3. Стратегии

Пусть бесконечная игра с конечным числом состояний осуществляется с помощью макроавтомата $A = (X \times Y, Q, q_0, T, F)$.

Стратегия черных есть оператор без предвосхищения, перерабатывающий входные сверхслова $x = x(0)x(1)\dots$, являющиеся последовательными ходами белых, в выходные сверхслова $y = y(0)y(1)\dots$, являющиеся ответными ходами черных. При этом ход черных $y(t)$ зависит от предшествующих ходов белых $x(0)x(1)\dots x(t)$. Стратегия белых есть оператор без предвосхищения, перерабатывающий сверхслова $y = y(0)y(1)\dots$, являющиеся ходами черных, в сверхслова $x = x(0)x(1)\dots$, являющиеся ходами белых. При этом ход белых $x(t)$ в момент t зависит от $y(0)y(1)\dots y(t-1)$, а ход $x(0)$ от ходов черных не зависит.

Стратегия белых есть оператор с задержкой. Всякий оператор $y = \mathbb{O}''(x)$ без предвосхищения есть некоторая стратегия черных. Всякий оператор $x = \mathbb{O}'(y)$ без предвосхищения с задержкой есть некоторая стратегия белых.

Если белые и черные выбрали стратегии (операторы) $\mathbb{O}'$ и $\mathbb{O}''$ соответственно, то партия, которая будет сыграна, полностью определена; обозначим ее через $(\mathbb{O}', \mathbb{O}'')$. Если в партии $(\mathbb{O}', \mathbb{O}'')$ белые начинают и выигрывают, то будем говорить, что стратегия белых $\mathbb{O}'$ бьет стратегию черных $\mathbb{O}''$. Если стратегия белых $\mathbb{O}'$ бьет любую стратегию черных $\mathbb{O}''$, то $\mathbb{O}'$ есть выигрывающая стратегия белых. Аналогично для черных.

Оператор $\mathbb{O}'$ (как и оператор $\mathbb{O}''$) может оказаться конечноавтоматным. В таком случае говорят о конечноавтоматных стратегиях. При этом автомат, реализующий стратегию белых, есть автомат с задержкой: выход белых зависит только от состояния.

Если $\mathbb{O}'$ и $\mathbb{O}''$ есть конечноавтоматные стратегии бесконечной игры, то партия $(\mathbb{O}', \mathbb{O}'')$ есть периодическое сверхслово в алфавите $X \times Y$. По заданным ав-

томатам для $\mathbb{O}'$ и $\mathbb{O}''$ предпериод и период этого периодического сверхслова строятся эффективно. В самом деле, пусть автоматы $A' = (Y, X, Q', q', T', B')$ и $A'' = (X, Y, Q'', q'', T'', B'')$ реализуют операторы $x = \mathbb{O}'(y)$ и $y = \mathbb{O}''(x)$ соответственно; при этом

$$\begin{aligned} q'(0) &= q'_0, & q''(0) &= q''_0, \\ q'(t+1) &= T'(q'(t), y(t)), & q''(t+1) &= T''(q''(t), x(t)), \\ x(t) &= B'(q'(t), y(t-1)), & y(t) &= B''(q''(t), x(t)). \end{aligned}$$

Значение $x(0)$ выбирается произвольно. Всякая партия $(\mathbb{O}', \mathbb{O}'')$ тогда имеет вид

$$\begin{aligned} &x(0)x(1) \dots x(t-1)x(t) \dots x(n-1)x(n) \dots \\ &y(0)y(1) \dots y(t-1)y(t) \dots y(n-1)y(n) \dots \\ &q'(0)q'(1) \dots q'(t) \dots q'(n) \dots \\ &q''(0)q''(1) \dots q''(t) \dots q''(n) \dots, \end{aligned}$$

где $x(0)$ из X произвольно, а все последующие ходы $(x(t), y(t))$ вычисляются с помощью функций T', B', T'', B'' . Так как мощности алфавитов Y, Q', Q'' конечны, то в этой партии впервые встретятся моменты m и n , для которых тройки $(y(m-1), q'(m), q''(m))$ и $(y(n-1), q'(n), q''(n))$ одинаковы. Тогда

$$\begin{aligned} x(m) &= B'(q'(m), y(m-1)) = B'(q'(n), y(n-1)) = x(n), \\ y(m) &= B''(q''(m), x(m)) = B''(q''(n), x(n)) = y(n), \\ q'(m+1) &= T'(q'(m), y(m)) = T'(q'(n), y(n)) = q'(n+1), \\ q''(m+1) &= T''(q''(m), x(m)) = T''(q''(n), x(n)) = q''(n+1), \end{aligned}$$

и потому длины предпериода и периода не превосходят произведения мощностей алфавитов Y, Q', Q'' , ибо отрезок партии от 0 до $m-1$ составляет ее предпериод, а отрезок от m до $n-1$ есть ее период.

Для всякой конечной игры, как отмечено, либо белые, либо черные имеют выигрывающую стратегию. Она может быть сделана конечноавтоматной. Для конечной игры, изображенной на рис. 12.1, белые имеют выигрывающую стратегию. Диаграмма конечного автомата (Мура) $A = (Y, X, Q, q_0, T, B)$, вычисляющего оператор с задержкой $\mathbb{O} : Y^* \rightarrow X^*$, реализующего выигрывающую стратегию белых, приведена на рис. 12.3. Состояния автомата помечены ходами белых; стрелки – ходами черных. В конечной игре, приведенной на рис. 12.4, выигрывают черные. Диаграмма автомата (Мили) $A = (X, Y, Q, q_0, T, B)$, вычисляющего выигрывающую стратегию черных, приведена на рис. 12.5.

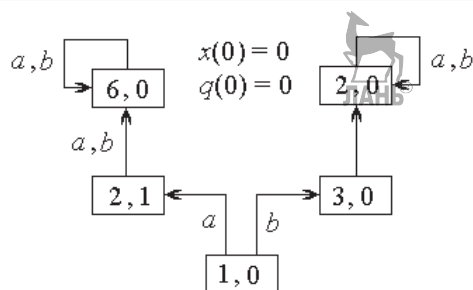


Рис. 12.3

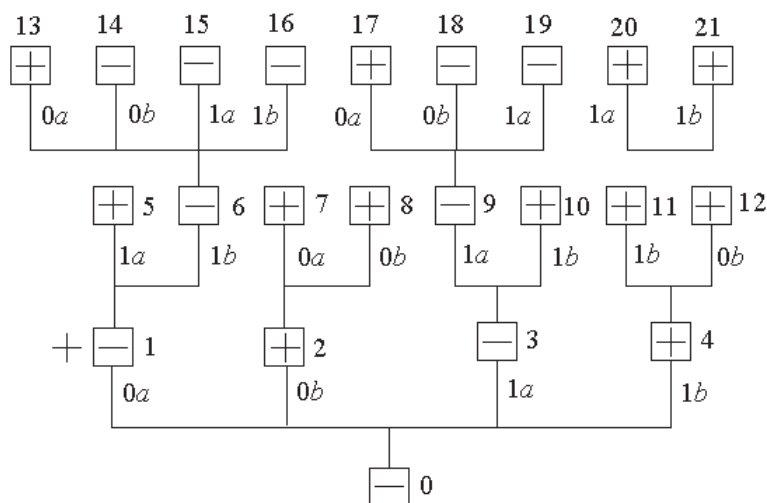


Рис. 12.4

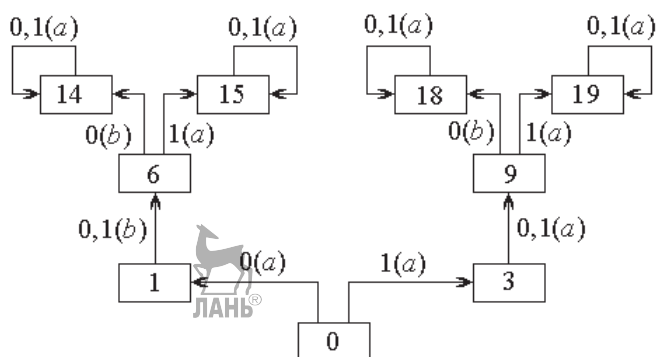


Рис. 12.5

В классе всех бесконечных игр существуют такие игры, в которых никакая из сторон не имеет выигрывающей стратегии. В случае конечноавтоматных игр положение иное: здесь всегда одна из сторон имеет выигрывающую стратегию и даже конечноавтоматную. К ее построению и переходим.

12.4. Униформизация конечноавтоматных языков

Проблема униформизации конечноавтоматного сверхязыка $L = (X \times Y)^\infty$, представимого макроавтоматом A , может быть интерпретирована как задача о существовании и построении выигрывающей конечноавтоматной стратегии, которая является оператором без предвосхищения $y = \mathbb{O}''(x)$ для черных или оператором без предвосхищения $x = \mathbb{O}'(y)$ для белых, ибо в конечноавтоматной игре всегда одна из сторон имеет выигрывающую конечноавтоматную стратегию.

12.4.1. Порядковые векторы и порядковые стратегии

Пусть $A = (X \times Y, Q, q_0, T, F)$ есть макроавтомат, представляющий сверхязык $L = L(A, q_0, F)$; $Q = \{q_0, q_1, \dots, q_n\}$ есть множество состояний в конечноавтоматной игре $G(A)$, соответствующей сверхязыку L .

Возможны $(n+1)!$ перестановок состояний из Q ; назовем каждую такую перестановку $(q_{i_0}, q_{i_1}, \dots, q_{i_n})$ порядковым вектором и будем обозначать его через $(i_0, i_1, \dots, i_n)$. Рассмотрим некоторую партию

$$\begin{array}{l} x(0) \ x(1) \dots x(t) \dots \\ y(0) \ y(1) \dots y(t) \dots \end{array} \quad (12.1)$$

в конечноавтоматной игре $G(A)$. Партии (12.1) соответствует последовательность состояний

$$q(0) \ q(1) \dots q(t) \dots \quad (12.2)$$

Наряду с последовательностью (12.2) сопоставим партии (12.1) последовательность порядковых векторов

$$\mathbf{q}(0) \ \mathbf{q}(1) \dots \mathbf{q}(t) \dots, \quad (12.3)$$

индуктивно определяемую следующим образом:

- 1) $\mathbf{q}(0) = (0, 1, \dots, n)$;
- 2) если $\mathbf{q}(t) = (i_0, i_1, \dots, i_k, i_{k+1}, \dots, i_n)$ и $\mathbf{q}(t+1) = q_{i_k}$, то $\mathbf{q}(t+1) = (i_k, i_1, i_2, \dots, i_{k-1}, i_{k+1}, \dots, i_n)$.

Ниже следует пример последовательности состояний и соответствующей последовательности порядковых векторов.

q_0	q_2	q_2	q_2	q_1	q_0	q_1	q_1	q_3	q_4	q_1	q_2	q_4
0	2	2	2	1	0	1	1	3	4	1	2	4
1	0	0	0	2	1	0	0	1	3	4	1	2
2	1	1	1	0	2	2	2	0	1	3	4	1

3	3	3	3	3	3	3	3	3	0	0	3	3
4	4	4	4	4	4	4	4	4	2	2	0	0

Справедлива следующая лемма.

Лемма 1. Если $\mathbf{q}(t) = (\dots, i, \dots, j, \dots)$, $\mathbf{q}(t+n) = (\dots, j, \dots, i, \dots)$, то в промежутках между t и $t+n$ вершина j (состояние q_j) посещалась, т. е. занимала в векторе первое место, по крайней мере один раз.

Пусть макросостояние $Q' = \{q_0, q_1, \dots, q_s\} \subseteq Q$. Пусть $n(t)$ есть номер позиции в векторе $\mathbf{q}(t)$, которая в следующем векторе $\mathbf{q}(t+1)$ стала первой. Например:

$$\begin{aligned}\mathbf{q}(t) &= (q_2, q_1, q_4, q_0, q_3, q_5), n(t) = 3, \\ \mathbf{q}(t+1) &= (q_4, q_2, q_1, q_0, q_3, q_5).\end{aligned}$$



Пусть в последовательности (12.3), которую запишем в виде

$$q(0) q(1) \dots q(t') q(t'+1) \dots q(t'') q(t''+1) \dots,$$

предельное макросостояние Q' формируется начиная с момента t' . Пусть t'' выбрано так, что множество состояний, встречающихся в слове $q(t')q(t'+1)\dots q(t'')$, совпадает со множеством Q' . Тогда

$$\forall t \geq t':$$

1) в порядковом векторе $\mathbf{q}(t)$ первые $s+1$ позиций заняты состояниями предельного макросостояния Q' ;

2) $n(t) = s$; кроме того, $(\exists^* t)(n(t) = s)$, где $\exists^* t$ означает: существует бесконечно много t .

В последовательности $t''+1, t''+2, \dots$ существует подпоследовательность натуральных чисел $m_0 < m_1 < \dots$, для которой $n(m_0) = n(m_1) = \dots = s$, и между m_i и m_{i+1} в первой позиции вектора $\mathbf{q}(t)$ побывали все состояния из Q' (и только они). Так как в последовательности $m_0 < m_1 < \dots$ встречается не более $(n+1)!$ векторов, то один из них, например $\mathbf{q} = (i_0, i_1, \dots, i_s, i_{s+1}, \dots, i_n)$ в последовательности порядковых векторов $\mathbf{q}(m_0), \mathbf{q}(m_1), \dots$, встречается бесконечно много раз, допустим, что на местах $t_0 < t_1 < \dots$, составляющих подпоследовательность в последовательности $m_0 < m_1 < \dots$ показана справедливость следующей леммы.

Лемма 2. Для макросостояния $Q' = \{q_0, q_1, \dots, q_s\}$ существует последовательность натуральных чисел $t_0 < t_1 < \dots$, для которой

$$1) \mathbf{q}(t_0) = \mathbf{q}(t_1) = \dots = (i_0, i_1, \dots, i_s, \dots, i_n);$$

$$2) n(t_0) = n(t_1) = \dots = s \text{ (а потому } \mathbf{q}(t_0+1) = \mathbf{q}(t_1+1) = \dots = (i_s, i_0, i_1, \dots, i_{s-1}, i_{s+1}, \dots, i_n);$$

$$3) \forall t \geq t_0 n(t) = s;$$

4) $\forall t \geq t_0$ в векторе $\mathbf{q}(t)$ первые s позиций заняты элементами из Q' ; причем между всякими t_i и t_{i+1} в первой позиции порядковых векторов $\mathbf{q}(t)$ побывали все состояния из Q' .

Стратегия белых, т. е. оператор без предвосхищения $x = \mathbb{O}(y)$, называется *порядковой*, если ход белых $x(t)$ зависит только от порядкового вектора $\mathbf{q}(t)$, а именно $x(t) = F_x(\mathbf{q}(t))$ для некоторой функции F_x . Стратегия черных называ-

ется *порядковой*, если ход черных $y(t)$ зависит только от $\mathbf{q}(t)$ и $x(t)$, а именно $y(t) = F_y(\mathbf{q}(t), x(t))$ для некоторой функции F_y .

Порядковые стратегии являются конечноавтоматными стратегиями с числом состояний не более $(n+1)!$. В самом деле, пусть в конечноавтоматной игре отображение $G(\mathbf{q}(t), x(t), y(t))$ есть функция, указывающая порядковый вектор, в которой пара ходов $(x(t), y(t))$ переводит порядковый вектор $\mathbf{q}(t)$. По заданной конечноавтоматной игре функция G может быть построена эффективно. Тогда для белых:

$$\begin{aligned}\mathbf{q}(0) &= (0, 1, \dots, n) = \mathbf{q}_0; \\ \mathbf{q}(t+1) &= G(\mathbf{q}(t), F_x(\mathbf{q}(t)), y(t)) = G_x(\mathbf{q}(t), y(t)); \\ x(t) &= F_x(\mathbf{q}(t));\end{aligned}$$

а для черных

$$\begin{aligned}\mathbf{q}(0) &= (0, 1, \dots, n) = \mathbf{q}_0; \\ \mathbf{q}(t+1) &= G(\mathbf{q}(t), x(t), F_y(\mathbf{q}(t), x(t))) = G_y(\mathbf{q}(t), x(t)); \\ y(t) &= F_y(\mathbf{q}(t), x(t)).\end{aligned}$$

Пусть $\mathbf{Q}$ есть множество всех порядковых векторов. Тогда автоматы $A_x = (X, Y, \mathbf{Q}, G_x, F_x)$, $A_y = (X, Y, \mathbf{Q}, G_y, F_y)$ вычисляют (реализуют) стратегии $\mathbb{O}'$ и $\mathbb{O}''$ белых и черных соответственно.

Заметим, что в рассмотренном случае выходные функции $x(t) = F_x(\mathbf{q}(t))$, $y(t) = F_y(\mathbf{q}(t), x(t))$, т. е. в порядковых стратегиях функция $x(t)$ не зависит от входа, поэтому построенные автоматы A_x и A_y являются автоматами Мура и Мили соответственно.

12.4.2. Теоремы о порядковых стратегиях

Теорема 1. Если в конечноавтоматной игре, осуществляемой макроавтоматом $A = (X \times Y, \mathbf{Q}, q_0, T, F)$, стратегия черных $y = \mathbb{O}''(x)$ бьет порядковую стратегию белых $x = \mathbb{O}'(y)$, то черные имеют порядковую стратегию $\hat{\mathbb{O}}''$, которая бьет $\hat{\mathbb{O}}'$.

Доказательство. Пусть в данной конечноавтоматной игре пара стратегий $(\hat{\mathbb{O}}', \mathbb{O}'')$ определяет партию

$$\begin{aligned}x(0) \ x(1) \ \dots \ x(t) \ \dots \\ y(0) \ y(1) \ \dots \ y(t) \ \dots \\ q(0) \ q(1) \ \dots \ q(t) \ \dots \\ \mathbf{q}(0) \ \mathbf{q}(1) \ \dots \ \mathbf{q}(t) \ \dots,\end{aligned}\tag{12.4}$$

выигрышную для черных. Тогда в этой партии предельное макросостояние $\lim q(t) = \mathbf{Q}' = \{q_0, q_1, \dots\} \in F$. Ходы противников в партии (12.4) в момент t

$$x(t) = F_x(\mathbf{q}(t)), \quad y(t) = \mathbb{O}''(x(t))\tag{12.5}$$

таковы, что белые действуют порядково, т. е. ход белых зависит только от порядкового состояния, а ход черных выбирается, вообще говоря, не порядково. По (непорядковой) выигрышной стратегии черных $\mathbb{O}''$ построим порядковую

стратегию $\hat{\Theta}''$, тоже выигрышную для черных. Пусть $t_0 < t_1 < \dots < t_s < \dots$ есть существующая по лемме 2 последовательность, для которой:

- 1) предельное макросостояние $Q' \forall t \geq t_0$ уже определилось;
- 2) $q(t_0) = q(t_1) = \dots = (i_0, i_1, \dots, i_s, i_{s+1}, \dots, i_n)$;
- 3) $n(t_1+1) = n(t_2+1) = \dots = s$, т. е. $q(t_1+1) = q(t_2+1) = \dots = (i_s, i_0, i_1, \dots, i_{s-1}, i_{s+1}, \dots, i_n)$;
- 4) в порядковых векторах на первом месте между t_i и t_{i+1} побывали все состояния из Q' (и только они), т. е. первые компоненты (позиции) порядковых векторов между t_i и t_{i+1} дают в точности множество Q' .

Рассмотрим в партии (Θ', Θ'') следующее начало:

$$\begin{array}{l}
 x(0) x(1) \dots x(t_0) x(t_0+1) \dots x(t_1) x(t_1+1) \\
 y(0) y(1) \dots y(t_0) y(t_0+1) \dots y(t_1) y(t_1+1) \\
 q(0) q(1) \dots q(t_0) q(t_0+1) \dots q(t_1) q(t_1+1) \\
 \mathbf{q}(0) \mathbf{q}(1) \dots \mathbf{q}(t_0) \mathbf{q}(t_0+1) \dots \mathbf{q}(t_1) \mathbf{q}(t_1+1)
 \end{array}
 \quad \text{ЛАНЬ}^{\circledR} \quad (12.6)$$

Пусть $\mathbf{q}$ есть некоторый порядковый вектор в (12.6). Так как стратегия белых $\hat{\Theta}'$ порядковая, то при появлении $\mathbf{q}$ белые действуют всегда одинаково: $x' = F_x(\mathbf{q})$. Ход черных при повторных появлениях вектора $\mathbf{q}$ не обязан быть одинаковым, ибо стратегия черных Θ'' не является порядковой. Построим порядковую стратегию черных $\hat{\Theta}''$ следующим образом. Полагаем вектор $\mathbf{r}(0) = \mathbf{q}(0)$. Проверяем, есть ли в (29.6) другая пара $(\mathbf{q}, x)$, равная $(\mathbf{r}(0), x(0))$. Если нет, то ход черных $y(0)' = y(0)$. Если да, то пусть n_0 есть ближайший слева к t_1 момент, для которого пара $(\mathbf{r}(0), x(0)) = (\mathbf{q}(n_0), x(n_0))$. Полагаем тогда $y'(0) = y(n_0) = F_y(\mathbf{r}(0), x(0))$.

Далее, пусть $\mathbf{r}(1) = F(\mathbf{r}(0), x(0), y'(0))$. Проверяем, есть ли в (12.6) другая пара $(\mathbf{q}, x)$, равная $(\mathbf{r}(1), x(1))$. Если нет, то $y'(1) = y(1)$. Если да, то пусть n_1 есть ближайший слева к t_1 момент, для которого $(\mathbf{r}(1), x(1)) = (\mathbf{q}(n_1), x(n_1))$. Тогда полагаем $y'(1) = y(n_1) = F_y(\mathbf{r}(1), x(1))$. И так далее до t_1 . Во всех остальных случаях значение $F_y(\mathbf{q}, x)$ выбираем произвольно. Порядковая стратегия черных $\hat{\Theta}''$ построена. Партия $(\hat{\Theta}', \hat{\Theta}'')$ имеет вид

$$\begin{array}{l}
 x(0) x(1) \dots x(t_0) \dots x(t_1) \dots \\
 y(0) y(1) \dots y(t_0) \dots y(t_1) \dots \\
 r(0) r(1) \dots r(t_0) \dots r(t_1) \dots \\
 \mathbf{r}(0) \mathbf{r}(1) \dots \mathbf{r}(t_0) \dots \mathbf{r}(t_1) \dots
 \end{array}
 \quad (12.7)$$

Для построенной партии (12.7) последовательность состояний, проходимых автоматом A , и соответствующая последовательность порядковых векторов $\mathbf{r}(t)$, в которых первая компонента есть $r(t)$, вообще говоря, отличаются от таковых в партии (12.5); при этом $\mathbf{r}(t) = \mathbf{q}(t)$ до тех пор, пока в (12.6) не встретилась ситуация $(\mathbf{q}, x)$, которая встречалась в (12.6) позже.

Покажем, что последовательность (12.7) порядковых векторов $\mathbf{r}(t)$ имеет начальный отрезок векторов

$$\mathbf{r}(0) \mathbf{r}(1) \dots \mathbf{r}(m) \dots \mathbf{r}(m+n), \quad (12.8)$$

первые позиции в которых на отрезке $[m, m+n]$ дадут в точности предельное макросостояние Q' . Другими словами, мы покажем, что в последовательности

(12.8) на отрезке $[m, m+n]$ в первой позиции побывали все состояния предельного макросостояния Q' и только они. Пусть $\forall p \in [0, t_1]$,

$$a_p = (\max t)_{t < t_1} (r(p) = q(t)).$$

Так как $r(0) = q(0)$, то величина

$$a_0 = (\max t)_{t < t_1} (r(0) = q(t))$$

существует, причем $0 \leq a_0 < t_1$ и

$$r(0) = q(a_0) \ \& \ (\forall t) (a_0 < t \leq t_1 \rightarrow r(0) \neq q(t)).$$

Если $a_0 = t_1$, то полагаем $m = 1$. Если $a_0 < t_1$, то по построению порядковой стратегии $\hat{\mathbb{O}}''$ ввиду

$$r(0) = q(a_0), \ x(0) = x(a_0), \ y'(0) = y(a_0)$$

вектор $r(1) = q(a_0+1)$, и потому a_1 существует; при этом

$$0 \leq a_0 < a_1 < t_1 \text{ и}$$

$$r(1) = q(a_1) \ \& \ (\forall t) (a_1 < t < t_1 \rightarrow r(1) \neq q(t)).$$

Если $a_1 = t_1$, то полагаем $m = t_1$. Если $a_1 < t_1$, то указанное выше построение продолжается. Процесс выделения возрастающей цепи $0 < a_0 < a_1 < \dots$ необходимо оборвется на некотором $a_m = t_1$, и тогда $r(m) = q(t_1) = (i_0, i_1, \dots, i_s, i_{s+1}, \dots, i_n)$.

Так как $q(t_1) = q(t_0)$, то $r(m) = q(t_0) = q(t_1)$.

Так как $x(m) = x(t_1)$, $y'(m) = y(t_1)$, то

$$r(m+1) = q(t_1+1) = q(t_0+1) = (i_s, i_0, i_1, \dots, i_{s-1}, i_{s+1}, \dots, i_n).$$

Поэтому a_{m+1} существует, $t_0 < a_{m+1} < t_1$ и

$$r(m+1) = q(a_{m+1}) \ \& \ (\forall t) (a_{m+1} < t < t_1 \rightarrow r(m+1) \neq q(t)).$$

Если $a_{m+1} = t_1$, то $n = 1$. Если $a_{m+1} < t_1$, то ввиду

$$r(m+1) = q(a_{m+1}), \ x(m+1) = x(a_{m+1}), \ y'(m+1) = y(a_{m+1})$$

вектор $r(m+2) = q(a_{m+1}+1)$; поэтому a_{m+2} существует и для него:

$$r(m+2) = q(a_{m+2}) \ \& \ (\forall t) (a_{m+1} < a_{m+2} < t_1 \rightarrow r(m+2) \neq q(t)).$$

Продолжая таким же образом, получим последовательность

$$t_0 < a_{m+1} < a_{m+2} < \dots < a_{m+n} = t,$$

причем $r(m+n) = q(t_1) = (i_0, i_1, \dots, i_s, i_{s+1}, \dots, i_n)$.

Так как $r(m+1) = (i_s, i_0, i_1, \dots, i_{s-1}, i_{s+1}, \dots, i_n)$, то по лемме 1 между $m+1$ и $m+n$ первые позиции векторов $r(m+1), \dots, r(m+n)$ дадут в точности предельное макросостояние Q' .

Теорема 1 доказана.

Аналогично доказывается следующая теорема.

Теорема 2. Если стратегия белых $\hat{O}'$ бьет порядковую стратегию черных $\hat{O}''$, то белые имеют порядковую стратегию $\hat{O}'$, которая бьет стратегию черных $\hat{O}''$.

Теорема 3. Пусть в конечноавтоматной бесконечной игре G белые и черные прибегают только к порядковым стратегиям. Тогда одна из сторон имеет выигрышную порядковую стратегию (которая бьет любую порядковую стратегию противника); эта выигрышная стратегия по игре G может быть эффективно построена.

Доказательство. Пусть оба игрока прибегают только к порядковым стратегиям. Тогда последовательности порядковых векторов, соответствующих сыгранным партиям, будут периодическими с длиной предпериода и периода в сумме не более $(n+1)!$. И предпериод, и период можно эффективно найти и по периоду найти предельное макросостояние (первые позиции в порядковых векторах периода). Если поменяем порядковую стратегию, то сыгранная партия поменяется, но выигрыш той или иной стороны определяется партией длины не более $(n+1)!$. То есть бесконечная игра G эквивалентна конечной игре G' с длительностью каждой партии не более $(n+1)!$. Правила игры G' таковы. Пусть уже сделано t ходов

$$\begin{aligned} &x(0) \ x(1) \ \dots \ x(t-1) \\ &y(0) \ y(1) \ \dots \ y(t-1) \\ &\mathbf{q}(0) \ \mathbf{q}(1) \ \dots \ \mathbf{q}(t-1) \ \mathbf{q}(t), \end{aligned}$$

после которых возник порядковый вектор $\mathbf{q}(t)$. Если этот вектор ранее не встречался, то игроки могут выбирать любой ход из своих алфавитов X и Y . Если вектор $\mathbf{q}(t)$ ранее возникал, т. е. $\mathbf{q}(t) = \mathbf{q}(t')$ при некотором $t' < t$, то белые и черные обязаны повторять ходы: $x(t) = x(t')$, $y(t) = y(t')$. Партия прекращается при первом повторении порядкового вектора. При этом в концевой вершине дерева игры ставим знак плюс, если предельное макросостояние между этими повторениями благоприятно для белых, и знак минус в противном случае.

Далее построенную конечную игру G' анализируем на предмет определения выигрышной стороны и построения для нее конечного автомата, вычисляющего для нее выигрышную стратегию, как это делали для конечных игр.

Теорема 4 (основная о конечноавтоматных играх). Во всякой игре с конечным числом состояний одна из сторон имеет выигрышную конечноавтоматную стратегию. Существует алгоритм, который по любой заданной автоматной игре:

- 1) выясняет, какая из сторон имеет выигрышную стратегию;
- 2) строит для выигрышной стороны одну из выигрывающих конечноавтоматных стратегий.

Доказательство. Алгоритм теоремы 3 строит порядковую стратегию $\hat{O}$, ко-

торая бьет любую порядковую стратегию $\hat{\mathbb{O}}'$ противника. Но эта же стратегия $\hat{\mathbb{O}}$ бьет и любую другую стратегию $\mathbb{O}'$ противника (не обязательно порядковую). В самом деле, если бы это было не так, т. е. если бы некоторая стратегия $\mathbb{O}'$ была $\hat{\mathbb{O}}$, то по теореме 1 и некоторая порядковая стратегия $\hat{\mathbb{O}}'$ была бы $\hat{\mathbb{O}}$, чего нет.

Следствие. Если в конечноавтоматной игре одна из сторон имеет какую-нибудь выигрывающую стратегию, то она имеет и конечноавтоматную выигрывающую стратегию.

Теорема 5 (перефразировка теоремы 4). Пусть дан произвольный сверхъязык L в алфавите $X \times Y$, который представим в некотором макроавтомате. Тогда либо сверхъязык L униформизируется конечноавтоматным оператором $u = \mathbb{O}'(x)$, либо его дополнение $(X \times Y)^* - L$ униформизируется конечноавтоматным оператором с задержкой $x = \mathbb{O}(y)$. Существует алгоритм, который по макроавтомату, представляющему сверхъязык L , выясняет, какая ситуация в указанной альтернативе имеет место, и строит конечный автомат, реализующий соответствующий оператор.

Следствие. Если существует оператор без предвосхищения, униформизирующий конечноавтоматный сверхъязык L , то существует и конечноавтоматный оператор, униформизирующий сверхъязык L . В частности, если график некоторого оператора без предвосхищения представим в конечном автомате, то этот оператор реализуем в конечном автомате.

Замечание. Пусть бесконечная игра осуществляется с помощью макроавтомата $A = (X \times Y, Q, q_0, T, F)$, где $F \subseteq P(Q)$ есть множество макросостояний, благоприятных для белых. Если выигрывают белые, то белые имеют выигрывающую стратегию $x = \mathbb{O}'(y)$, униформизирующую сверхъязык $L(A, q_0, F)$, ибо стратегия $\mathbb{O}'$ «загоняет» фишку в предельное макросостояние из F . Если выигрывают черные, то черные имеют выигрывающую стратегию $y = \mathbb{O}''(x)$, униформизирующую дополнительный сверхъязык, ибо стратегия $\mathbb{O}''$ «загоняет» фишку в предельное макросостояние из $P(Q)$, лежащее вне F .

12.4.3. Пример построения выигрывающего автомата

Пусть конечный автомат для бесконечной игры задается диаграммой, приведенной на рис. 12.6. $X = \{a, b\}$; $Y = \{c, d\}$. Макросостояния $\{0\}, \{1\}; \{0, 1, 2\}$ благоприятны для белых; остальные возможные для этого автомата макросостояния $\{2\}, \{0, 1\}, \{1, 2\}$ благоприятны для черных. Начальное состояние есть 0. Строим конечную игру (рис. 12.7), вершины которой помечены порядковыми векторами (i_0, i_1, i_2) . Построение конечного дерева на некотором пути прекращается при появлении на этом пути пары одинаковых порядковых векторов. Множество состояний между одинаковыми порядковыми векторами на одном и том же пути составляет предельное макросостояние.



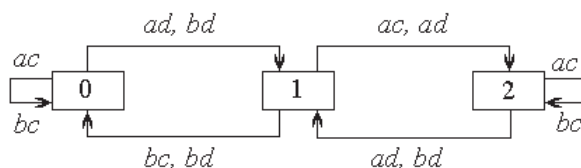


Рис. 12.6

Просматривая дерево игры на рис. 12.7, видим, что черные имеют выигрышную стратегию (рис. 12.8). Начальное состояние 012 расположено в корне дерева. Диаграмма автомата (Мили), ее вычисляющая, приведена на рис. 12.9. В скобках указаны выходы автомата. Входные символы автомата есть ходы белых; выходные (символы в скобках) есть ответы черных.

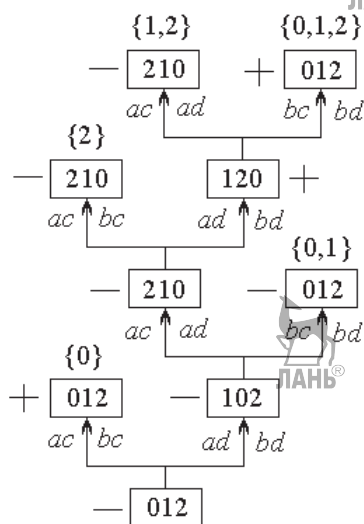


Рис.12.7

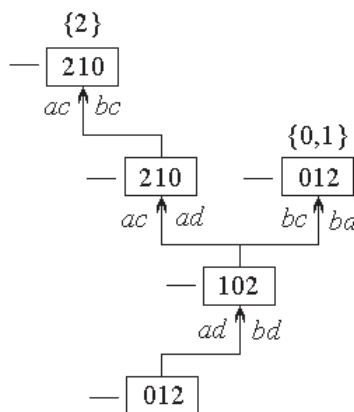


Рис.12.8

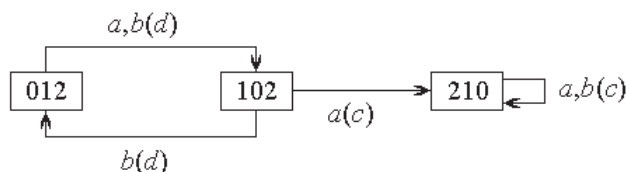


Рис. 12.9

Если для того же самого автомата, диаграмма которого изображена на рис. 12.6, к числу макросостояний, благоприятных для белых, добавить макросостояние $\{2\}$, то получим игру (рис. 12.10), в которой белые имеют выигрывающую стратегию (рис. 12.11). Диаграмма автомата (Мура), вычисляющего выигрывающую стратегию белых, приведена на рис. 12.12; начальное состояние 012; состояния помечены ходами белых; стрелки переходов – ходами черных.

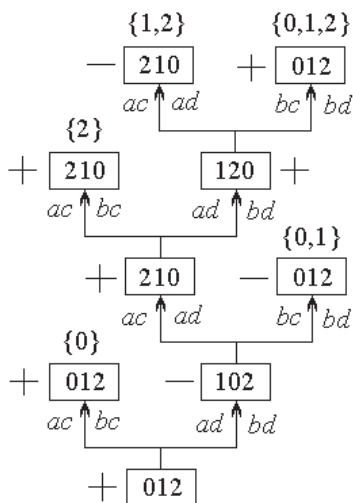


Рис. 12.10

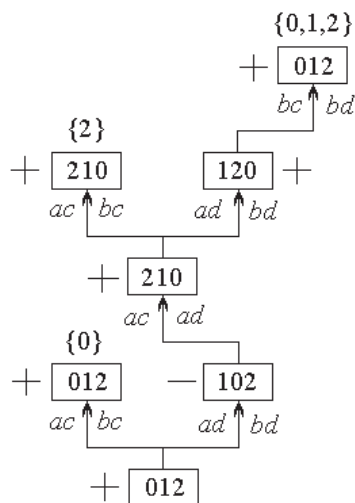


Рис. 12.11

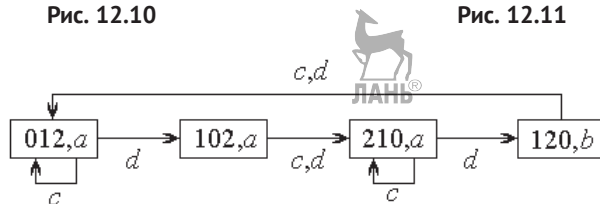


Рис. 12.12

Монадическая логика натуральных чисел



13.1. Монадическая логика

Опишем монадическую логику – интерпретируемый формализм, связанный с логикой одноместных предикатов (ЛОП) второго порядка на натуральных числах с функцией следования (т. е. прибавления единицы) $y = x + 1$ на натуральном ряду. ЛОП второго порядка называют также монадической логикой натуральных чисел второго порядка (МЛН).

Формальные символы

1. $0, 1, 2, \dots$ есть символы натуральных чисел.
2. $x, y, z, \dots$ есть символы предметных переменных, пробегающих натуральный ряд, возможно с индексами.
3. $'$ есть штрих, символ функции следования на натуральном ряду, $x' = x + 1$.
4. $a_1, a_2, \dots, a_n$ есть символы конечного алфавита $\Sigma = \{a_1, a_2, \dots, a_n\}$.
5. $X, Y, Z, \dots$ есть символы предикатных переменных (возможно, с индексами), аргументы которых пробегают натуральный ряд, а значения – множество символов алфавита Σ . Правильнее было бы сказать, что $X, Y, Z, \dots$ есть символы для конечнозначных функций, с аргументами, пробегающими натуральный ряд, и со значениями во множестве Σ . Сохраним все же традиционную предикатную терминологию.

6. $\&, \vee, \neg, \rightarrow, \exists, \forall$ есть логические символы: конъюнкция, дизъюнкция, отрицание, импликация, квантор существования, квантор общности.

7. $=$ есть знак равенства.

8. $(,)$ есть скобка левая, запятая, скобка правая.

Термы

1. Символ натурального числа есть терм.
2. Символ предметной переменной есть терм.
3. Если t есть терм, то t' есть терм.

Замечание. Вместо $t', t'', t''', \dots$ будем писать также $t+1, t+2, t+3, \dots$

Формулы

1. Если X есть предикатный символ, а t есть терм, содержащий предметную переменную или без нее, и a есть буква из Σ , то $X(t) = a$ есть атомарная формула с глубиной построения 1, в которой предикатная переменная X и предметная переменная x , если она есть, являются свободными переменными.

2. Если A и B есть формулы с глубиной построения r и s соответственно, то $(A \ \& \ B)$, $(A \vee B)$, $(A \rightarrow B)$ есть формулы с глубиной построения $\max(r, s) + 1$, свободные переменные которых – вместе взятые свободные переменные формул A и B . Выражение $(\neg A)$ – тоже формула с глубиной построения $r + 1$. Свободные переменные формулы $(\neg A)$ есть свободные переменные формулы A .

3. Если $A(x)$ есть формула со свободной предметной переменной x и с глубиной построения r , то $(\exists x)A(x)$, $(\forall x)A(x)$ есть формулы с глубиной построения $r + 1$, переменная x в которой связана (квантором). Свободные переменные формул $(\exists x)A(x)$, $(\forall x)A(x)$ есть все свободные переменные формулы $A(x)$, кроме x .

4. Если $A(X)$ есть формула со свободной предикатной переменной X и с глубиной построения r , то $(\exists X)A(X)$, $(\forall X)A(X)$ есть формулы с глубиной построения $r + 1$, переменная X в которой связана (квантором). Свободные переменные формул $(\exists X)A(X)$, $(\forall X)A(X)$ есть все свободные переменные формулы $A(X)$, кроме X .

Формула называется *замкнутой*, если она не имеет свободных переменных. Всякая замкнутая формула содержательно представляет собой высказывание о натуральном ряде и о его подмножествах, истинное или ложное.

Построенный интерпретируемый формализм составляет *монадическую логику* (МЛН) натуральных чисел, или логику одноместных предикатов (ЛОП) второго порядка натуральных чисел с функцией следования на натуральном ряду.

Вместо $X(t) = a$ для краткости будем писать также $X^a(t)$. Предикат $X(x)$, определенный на натуральном ряду, можно рассматривать как сверхслово $X(0)X(1)X(2)\dots$ в алфавите Σ .

Пусть $A(X_1, X_2, \dots, X_k)$ есть формула МЛН без свободных предметных переменных и со свободными предикатными переменными $X_1, X_2, \dots, X_k$. Набор фиксированных предикатов $X_1, X_2, \dots, X_k$ (назовем этот набор k -предикатом)

$$\begin{aligned} &X_1(0) X_1(1) \dots X_1(t) \dots \\ &X_2(0) X_2(1) \dots X_2(t) \dots \\ &\dots \\ &X_k(0) X_k(1) \dots X_k(t) \dots, \end{aligned}$$

написанных один под другим, можно рассматривать как «толстое» слово «толщины» k в алфавите $\Sigma \times \Sigma \times \dots \times \Sigma$ (k раз). Тогда множество истинности формулы $A(X_1, \dots, X_k)$ есть некоторый сверхязык в алфавите $\Sigma \times \Sigma \times \dots \times \Sigma$ (k раз).

Пусть $A(X, X_1, \dots, X_k)$ есть формула в МЛН без свободных предикатных переменных, и $X, X_1, \dots, X_k$ есть полный список ее свободных предикатных пере-

менных. Пусть $\lceil A \rceil = \{(X, X_1, \dots, X_k) : \text{высказывание } A(X, X_1, \dots, X_k) \text{ истинно}\}$ есть множество истинности формулы A . Пусть формула $B \ B(X_1, \dots, X_k) = (\exists X)A(X, X_1, \dots, X_k)$. Тогда множество истинности формулы B может быть получено из множества истинности формулы A вычеркиванием (удалением) во всех наборах $(X, X_1, \dots, X_k)$ из $\lceil A \rceil$ первой компоненты X . Если, например, в алфавите $\Sigma = \{0, 1, 2\}$ 4-предикат

$$\begin{aligned} X &= 1\ 0\ 1\ 1\ 2\ 2\ 0\ 1\ 2\ 3\ \dots \\ X_1 &= 2\ 1\ 0\ 1\ 1\ 2\ 0\ 0\ 1\ 1\ \dots \\ X_2 &= 0\ 1\ 2\ 2\ 1\ 2\ 0\ 1\ 2\ 0\ \dots \\ X_3 &= 1\ 2\ 2\ 1\ 1\ 0\ 1\ 2\ 0\ 1\ \dots \end{aligned}$$



принадлежит множеству истинности некоторой формулы $A(X, X_1, X_2, X_3)$, то 3-предикат

$$\begin{aligned} X_1 &= 2\ 1\ 0\ 1\ 1\ 2\ 0\ 0\ 1\ 1\ \dots \\ X_2 &= 0\ 1\ 2\ 2\ 1\ 2\ 0\ 1\ 2\ 0\ \dots \\ X_3 &= 1\ 2\ 2\ 1\ 1\ 0\ 1\ 2\ 0\ 1\ \dots \end{aligned}$$

принадлежит множеству истинности формулы $B(X_1, X_2, X_3) = (\exists X)A(X, X_1, X_2, X_3)$. Можно считать, что этот 3-предикат получается из исходного 4-предиката поразрядной заменой, т. е. проекцией $\{0, 1, 2\}^4 \rightarrow \{0, 1, 2\}^3$, при которой $(b_0, b_1, b_2, b_3)^T \rightarrow (b_1, b_2, b_3)^T$, где каждое b_i лежит в Σ , а буква T означает транспонирование.

Если $B(X_1, \dots, X_k)$ есть $(\exists X)A(X, X_1, \dots, X_k)$, то можно считать, что множество истинности формулы B получается из множества истинности формулы A проектированием последнего на оси $1, 2, \dots, k$ с помощью проекции $\Sigma^{k+1} \rightarrow \Sigma^k$, при которой $(b_0, b_1, \dots, b_k)^T \rightarrow (b_1, \dots, b_k)^T$, где все b_i лежат в Σ . Таким образом, $\lceil B \rceil = \text{Pr}_{1,2,\dots,k}(\lceil A \rceil)$. Переменная X в формуле A может стоять на любом аргументном месте формулы A .

13.2. Выразимость в монадической логике

Покажем, что монадическая логика второго порядка есть довольно гибкий инструмент для записи работы автоматов, источников, макроисточников, регулярных и общерегулярных языков. Анализ указанных объектов с помощью МЛН довольно прост и естественен. Синтез осуществляется сложнее. Рассмотрим его ниже.

В МЛН выразимы формулами следующие предикаты.

$$\begin{aligned} x &= y \leftrightarrow (\forall X)(X^a(x) \rightarrow X^a(y)), \ a \in \Sigma. \\ x &= 0 \leftrightarrow (\forall y)(x \neq y+1). \\ x &= 1 \leftrightarrow (\forall y)(x \neq y+2) \ \& \ x \neq 0. \\ x &= 2 \leftrightarrow (\forall y)(x \neq y+3) \ \& \ x \neq 0 \ \& \ x \neq 1. \\ &\dots \\ x &= k \leftrightarrow (\forall y)(x \neq y+k+1) \ \& \ x \neq 0 \ \& \ \dots \ \& \ x \neq k-1. \end{aligned}$$



Заметим, что

$$\begin{aligned} X(t) \neq a &\leftrightarrow \bigvee_{b \in \Sigma - \{a\}} X(t) = b; \\ X^a(x+1) &\leftrightarrow (\exists y)(x+1=y \ \& \ X^a(y)). \end{aligned} \quad (13.1)$$

В формуле, следующей за (13.1), переменная y в $X(y)$ не имеет штриха.

$$\begin{aligned} x \leq y &\leftrightarrow (\forall X)(X^a(x) \ \& \ (\forall x)(X^a(x) \rightarrow X^a(x')) \rightarrow X^a(y)), \ a \in \Sigma. \\ x < y &\leftrightarrow x \leq y \ \& \ x \neq y. \\ x \geq y &\leftrightarrow y \leq x. \\ x > y &\leftrightarrow y < x. \end{aligned}$$



Введем ограниченные кванторы:

$$\begin{aligned} (\exists y)_{\leq x} A(y) &\leftrightarrow (\exists y)(y \leq x \ \& \ A(y)). \\ (\forall y)_{\leq x} A(y) &\leftrightarrow (\forall y)(y \leq x \rightarrow A(y)). \\ (\exists t)_x^y A(t) &\leftrightarrow (\exists t)(x \leq t < y \ \& \ A(t)). \\ (\forall t)_x^y A(t) &\leftrightarrow (\forall t)(x \leq t < y \rightarrow A(t)). \end{aligned}$$

Введем также неограниченные кванторы:

$$\begin{aligned} (\exists^{\omega} x) A(x) &\leftrightarrow (\forall y)(\exists x)_{>y} A(x). \\ (\forall^{\omega} x) A(x) &\leftrightarrow (\exists y)(\forall x)_{>y} A(x). \end{aligned}$$

Определим формулами МЛН следующие предикаты:

$$\begin{aligned} a \in \lim X(x) &\leftrightarrow (\exists^{\omega} x)(X(x) = a); \\ \lim X(x) = \{a_{i1}, \dots, a_{is}\} &\leftrightarrow \\ &\bigwedge_{k=1}^s (a_{ik} \in \lim X(x)) \ \& \ \bigwedge_{a \in \Sigma} (a \in \lim X(x) \rightarrow \bigvee_{k=1}^s (a = a_{ik})). \end{aligned}$$

Теорема (анализа). По всякому источнику S можно построить формулу МЛН $A(X, x, y)$, для которой слово

$$X(x)X(x+1)\dots X(y-1) \in \text{Beh}(S) \leftrightarrow A(X, x, y).$$

Доказательство. Источнику без пустых ребер $B = (X, Q, Q_0, D, F)$, где $F \subseteq Q$, поставим в соответствие формулу МЛН.

$A(X, x, z)$, которая есть

$$(\exists Q) ((\bigvee_{q \in Q} Q(x) = q \ \& \ (\forall y)_{x, a, q' \in D}^z (\bigvee_{q \in F} (Q(y) = q \ \& \ X(y) = a \ \& \ Q(y+1) = q')) \rightarrow (\bigvee_{q \in F} Q(z) = q)).$$

Тогда слово $X(x)X(x+1)\dots X(z-1) \in \text{Beh}(S) \leftrightarrow A(X, x, z)$.

Заметим, что формула A фиктивна вне $[x, z]$ в том смысле, что значения предиката X вне $[x, z]$ не влияют на характер истинности формулы A . Если хотим иметь в источнике и пустые ребра, то алфавит Σ следует расширить пустым символом.



13.2.1. Макроисточники и монадическая логика

Теорема (анализа). По всякому макроисточнику MS можно построить формулу МЛН $A(X, x)$, для которой сверхслово $X(x)X(x+1)\dots \in Beh(MS) \leftrightarrow A(X, x)$. Значение предиката X вне $[0, x)$ не существенно.

Доказательство. Макроисточнику $MS = (X, Q, Q_0, D, F')$, где $F' \in P(Q)$, сопоставим формулу $A(X, x)$, которая есть

$$(\exists Q) ((\bigvee_{q \in Q_0} Q(x) = q) \ \& \ (\forall y)_{\geq x} (\bigvee_{q, a, q' \in D} (Q(y) = q \ \& \ X(y) = a \ \& \ Q(y+1) = q')) \ \& \ (\bigvee_{F \in F'} \lim Q(x) = F)).$$

Сверхслово $X(0)X(1)\dots X(x)X(x+1) \in Beh(MS) \leftrightarrow A(X, x)$. Формула $A(X, x)$ фиктивна вне $[0, x)$.

13.2.2. Регулярные языки и монадическая логика

Теорема (анализа). Для всякого регулярного языка R можно построить формулу МЛН $A(X, x)$, для которой слово $X(x)X(x+1)\dots X(y-1) \in R \leftrightarrow A(X, x, y)$. Значение предиката X вне $[x, y)$ не существенно.

Доказательство. Индукция по глубине k построения R .

Базис. $k = 1$. R есть a , где $a \in \Sigma$. Тогда $A(X, x, y)$ есть $X(x) = a \ \& \ y = x + 1$.

Предположение индукции. Допустим, что для всякого регулярного языка с глубиной построения k требуемая формула МЛН может быть построена.

Шаг индукции. Покажем, что для всякого регулярного языка с глубиной построения k требуемая формула может быть построена. Пусть регулярный язык R имеет глубину построения k . Возможны следующие случаи.

1. $R = R_1 * R_2$, где $*$ $\in \{\vee, \cdot\}$. Так как глубина построения регулярных языков R_1 и R_2 меньше k , то по предположению индукции существуют формулы $A_1(X, x, y)$ и $A_2(X, x, y)$, для которых слово

$$X(x)X(x+1)\dots X(y-1) \in R_i \leftrightarrow A_i(X, x, y), \ i = 1, 2.$$

Тогда формула A , равная $A_1 \vee A_2$, соответствует языку $R_1 \vee R_2$, а формула $A(X, x, y)$, равная $(\exists t)_x^y (A_1(X, x, t) \ \& \ A_2(X, t, y))$, соответствует языку $R_1 \cdot R_2$.

2. $R = R_1^*$. Так как R_1 имеет глубину построения меньше k , то по предположению индукции для R_1 существует формула МЛН $A_1(X, x, y)$, для которой $X(x)X(x+1)\dots X(y-1) \in R_1 \leftrightarrow A_1(X, x, y)$. Пусть формула МЛН $B(X, x, y)$ есть $Y(x) = a \ \& \ Y(y) = a \ \& \ (\forall t)(x < t < y \rightarrow Y(t) \neq a)$. Положим формулу $A(X, x, y)$ равной

$$x < y \ \& \ (\exists Y) (Y(x)=a \ \& \ Y(y)=a \ \& \ (\forall u)(\forall v)(x \leq u < v \leq y \ \& \ B(Y, u, v) \rightarrow A_1(X, u, v))).$$

Тогда слово $X(x)X(x+1)\dots X(y-1) \in R \leftrightarrow A(X, x, y)$. Значения предиката X вне $[x, y)$ не существенны.

Замечание. Сверхслово Y имеет вхождение буквы a на тех позициях отрезка $[x, y)$, которые разбивают слово $X(x)X(x+1)\dots X(y-1)$ на куски, принадлежащие итерируемому языку.

13.2.3. Общерегулярные языки и монадическая логика

Теорема (анализа). Для всякого общерегулярного языка Ω можно построить формулу МЛН $A(X, x)$, для которой сверхслово

$$X(x)X(x+1) \dots \in \Omega \leftrightarrow A(X, x).$$

Значения предиката X вне $[0, x)$ не существенны.

Доказательство. Индукция по глубине r построения Ω .

Базис. $k = 1$. Пусть R есть регулярный язык и $\Omega \subseteq R^\omega$ есть общерегулярный язык. Для R существует формула МЛН $A_1(X, x, y)$, для которой слово $X(x)X(x+1) \dots X(y-1) \in R \leftrightarrow A_1(X, x, y)$.

Пусть формула $A(X, x)$ есть

$$(\exists y)(Y(x) = a \ \& \ (\exists^\omega y)(Y(y) = a) \ \& \ (\forall u)(\forall v)(x \leq u < v \ \& \ B(Y, u, v) \rightarrow A_1(X, u, v))),$$

где B есть построенная в предыдущей теореме формула. Тогда сверхслово $X(x)X(x+1) \dots \in R^\omega \leftrightarrow A(X, x)$.

Значения предиката X вне $[0, x)$ не существенны.

Предположение индукции. Пусть для всякого общерегулярного сверхъязыка с глубиной построения меньше k требуемая формула существует.

Шаг индукции. Покажем, что для всякого общерегулярного сверхъязыка с глубиной построения k требуемая формула может быть построена. Пусть общерегулярный сверхъязык Ω имеет глубину построения k . Тогда $\Omega = R \cdot \Omega_1$ для некоторого регулярного языка R и общерегулярного сверхъязыка Ω_1 . Для R существует формула МЛН $C(X, x, y)$, для которой слово

$$X(x)X(x+1) \dots X(y-1) \in R \leftrightarrow C(X, x, y).$$

Так как глубина построения Ω_1 меньше k , то по предположению индукции для Ω_1 существует формула $D(X, x)$, для которой сверхслово

$$X(x)X(x+1) \dots \in \Omega_1 \leftrightarrow D(X, x).$$

Пусть формула $A(X, x)$ есть $(\exists y)(x < y \ \& \ C(X, x, y) \ \& \ D(X, y))$.

Тогда сверхслово $X(x)X(x+1) \dots \in \Omega \leftrightarrow A(X, x)$. Значения предиката X вне $[0, x)$ не существенны.

13.3. Специальная префиксная форма

Формула A в МЛН имеет специальную префиксную форму, если A находится в префиксной форме и при этом в кванторной приставке все предикатные кванторы предшествуют всем предметным кванторам. Покажем, что всякая формула МЛН эквивалентна некоторой формуле МЛН в специальной префиксной форме. Пусть A есть формула МЛН. В формуле A устраним все вхождения нуля, а именно если формула A имеет вид $A(0)$, то заменим ее на формулу $(\exists x)(x = 0 \ \& \ A(x))$, где $x = 0$ есть $(\forall y)(x \neq y + 1)$. В результате получим формулу A_1 , не имеющую нулей и эквивалентную формуле A .

В формуле A_1 устраним итерации штриха, заменяя вхождения атомарной формулы $X(x''...')$, где штрих встречается k раз, на формулу

$$(\exists y_1)(\exists y_2)...(\exists y_k) (y_1 = x+1 \ \& \ y_2 = y_1+1 \ \& \ ... \ \& \ y_k = y_{k-1}+1 \ \& \ X(y_k)).$$

В результате получим формулу A_2 , эквивалентную исходной формуле, при этом каждая предметная переменная в A_2 имеет не более одного штриха, а предметные переменные, являющиеся аргументами свободных предикатных переменных формулы A , вовсе не имеют штрихов.

Приведем формулу A_2 к префиксной форме. В результате получим формулу A_3 в префиксной форме, эквивалентную исходной формуле A .

Справедливы следующие эквивалентности:

$$(\exists x)(\forall X)C(X,x) \leftrightarrow (\exists Y)(\forall X)(\forall x)((\exists y)Y^a(y) \ \& \ (Y^a(x) \rightarrow C(X,x))); \quad (13.2)$$

$$(\forall x)(\exists X)C(X,x) \leftrightarrow (\forall Y)(\exists X)(\exists x)((\forall y)Y^a(y) \rightarrow Y^a(x) \ \& \ C(X,x)); \ a \in \Sigma. \quad (13.3)$$

Докажем первую из них. Пусть в формуле (13.2) истинна правая часть. Тогда имеем следующий ряд эквивалентных формул:

$$\begin{aligned} &(\exists Y)(\forall X)(\forall x)((\exists y)Y_0^a(y) \ \& \ (Y_0^a(x) \rightarrow C(X,x))); \\ &(\forall X)(\forall x)((\exists y)Y_0^a(y) \ \& \ (Y_0^a(x) \rightarrow C(X,x))); \\ &(\forall x)(Y_0^a(y_0) \ \& \ (Y_0^a(x) \rightarrow C(X,y_0))); \\ &Y_0^a(y_0) \ \& \ (Y_0^a(y_0) \rightarrow C(X,y_0)); \\ &Y_0^a(y_0); \ Y_0^a(y_0) \rightarrow C(X,y_0); \\ &C(X,y_0); \\ &(\forall X)C(X,y_0); \\ &(\exists x)(\forall X)C(X,x). \end{aligned}$$

Пусть теперь в (13.2) истинна левая часть $(\exists x)(\forall X)C(X,x)$. Тогда $(\forall X)C(X,y_0)$ истинно при некотором y_0 . Зададим предикат $Y_0(y)$ так, чтобы

$$Y_0^a(y_0) \ \& \ (\forall x)(x \neq y_0 \rightarrow Y_0(x) \neq a).$$

Следующие далее формулы последовательно эквивалентны друг другу (во всех этих формулах $a \in \Sigma$):

$$\begin{aligned} &Y_0^a(y_0); \ (\forall X)C(X,y_0); \\ &Y_0^a(y_0); \ Y_0^a(y_0) \rightarrow (\forall X)C(X,y_0); \\ &Y_0^a(y_0); \ (\forall x)(Y_0^a(x) \rightarrow (\forall X)C(X,x)); \\ &(\exists y)Y_0^a(y); \ (\forall x)(\forall X)(Y_0^a(x) \rightarrow C(X,x)); \\ &(\exists y)Y_0^a(y) \ \& \ (\forall X)(\forall x)(Y_0^a(x) \rightarrow C(X,x)); \\ &(\forall X)(\forall x)((\exists y)Y_0^a(y) \ \& \ (Y_0^a(x) \rightarrow C(X,x))); \\ &(\exists Y)(\forall X)(\forall x)((\exists y)Y^a(y) \ \& \ (Y^a(x) \rightarrow C(X,x))). \end{aligned}$$

Справедливость соотношения (13.2) установлена. Справедливость формулы (13.3) показывается аналогично. Левая и правая части в (13.3) двойственны соответствующим частям в (13.2).

Используем эквивалентности (13.2) и (13.3) для разделения предикатных и предметных кванторов в формуле A_3 . В результате получим формулу A_4 в специальной префиксной форме, эквивалентную исходной формуле A .

13.4. Синтез автомата по формуле монадической логики

Пусть $A(X_1, \dots, X_n)$ есть формула МЛН; $X_1, \dots, X_n$ есть полный список ее свободных предикатных переменных; свободных предметных переменных формула A не имеет. Покажем, что по формуле A можно построить макроавтомат, поведение которого совпадает со множеством истинности формулы A .

Так как для всякой формулы МЛН существует ей эквивалентная формула в специальной префиксной форме, то можно сразу считать, что формула A находится в специальной префиксной форме, при этом согласно предыдущему пункту свободные предикатные переменные $X_1, \dots, X_n$ формулы A имеют предметные переменные без штрихов. Формула A имеет следующий вид:

$$(Q_1 Y_1) \dots (Q_r Y_r) (Q'_1 x_1) \dots (Q'_k x_k) B,$$

где B есть бескванторная формула МЛН. Формулу B представим в виде СДНФ; далее согласно формуле (13.1) заменим отрицания вида $X(t) \neq a$ на эквивалентную формулу $\bigvee_{b \in \Sigma - \{a\}} (X(t) = b)$, не содержащую отрицаний; результат приведем к СДНФ. Тогда получим формулу A_1 , эквивалентную исходной формуле и имеющую вид

$$\begin{aligned} & (Q_1 Y_1) \dots (Q_r Y_r) (Q'_1 x_1) \dots (Q'_k x_k) \\ & (\bigvee_{i_1} A_{i_1 1}(x_1) \dots A_{i_1 r}(x_r) B_{i_1 1}(x'_1) \dots B_{i_1 r}(x'_r) C_{i_1 1}(x_1) \dots C_{i_1 n}(x_1) \\ & \dots \\ & A_{i_k 1}(x_k) \dots A_{i_k r}(x_k) B_{i_k 1}(x'_k) \dots B_{i_k r}(x'_k) C_{i_k 1}(x_k) \dots C_{i_k n}(x_k)), \end{aligned}$$

где каждые $A_{ijl}(x_l)$, $B_{ijl}(x'_l)$, $C_{ijl}(x_l)$ есть некоторые равенства

$$Y_j(x_l) = a_{ijl}, \quad Y_j(x'_l) = b_{ijl}, \quad X_j(x_l) = c_{ijl}; \quad a_{ijl}, b_{ijl}, c_{ijl} \in \Sigma.$$

Так как в бескванторную часть формулы A_1 входят только одноместные предикатные переменные, то можно использовать процедуру Бемана продвижения предметных кванторов в глубину бескванторной СДНФ в формуле A_1 . Продвигаем сначала квантор $(Q'_k x_k)$. Если Q' есть квантор $\exists$, то в A_1 продвигаем квантор $(Q'_k x_k)$ сначала по дизъюнкции, а потом по конъюнкции. В результате получим формулу

$$\begin{aligned} & (Q_1 Y_1) \dots (Q_r Y_r) (Q'_1 x_1) \dots (Q'_{k-1} x_{k-1}) \\ & (\bigvee_{i_1} A_{i_1 1}(x_1) \dots A_{i_1 r}(x_r) B_{i_1 1}(x'_1) \dots B_{i_1 r}(x'_r) C_{i_1 1}(x_1) \dots C_{i_1 n}(x_1) \\ & \dots \\ & A_{i_{k-1}, 1}(x_{k-1}) \dots A_{i_{k-1}, r}(x_k) B_{i_{k-1}, 1}(x'_k) \dots B_{i_{k-1}, r}(x'_k) C_{i_{k-1}, 1}(x_{k-1}) \dots C_{i_{k-1}, n}(x_{k-1})) \end{aligned}$$

$$\frac{(\exists x_k) A_{ik1}(x_k) \dots A_{ikr}(x_k) B_{ik1}(x'_k) \dots B_{ikr}(x'_k) C_{ik1}(x_k) \dots C_{ik1}(x_k))}{E_{ik} = (\exists x_k) D_{ik}(x_k)}$$

Формула E_{ik} выступает в дальнейших преобразованиях единым неизменным блоком. Квантор $(\exists x_k)$ в формулу A_1 продвинут.

Если Q'_k есть $\forall$, то справедлива следующая последовательность эквивалентных формул:

$$(\forall x_k)(\text{СДНФ}) \leftrightarrow \neg \neg (\forall x_k)(\text{СДНФ}) \leftrightarrow \neg (\exists x_k) \neg (\text{СДНФ}) \leftrightarrow \neg (\exists x_k)(\text{СДНФ}1).$$

Далее квантор $(\exists x_k)$ продвигаем в глубину СДНФ1, как это делалось выше, а внешнее отрицание распределяется над получившейся в результате продвижения квантора существования ДНФ. В обоих случаях продвижения квантора в глубину формулы A_1 и последующего превращения бескванторной части в СДНФ получим формулу A_2 вида

$$\begin{aligned} & ((Q_1 Y_1) \dots (Q_r Y_r) (Q'_1 x_1) \dots (Q'_{k-1} x_{k-1}) (\bigvee_i A_{i11}(x_1) \dots A_{i1r}(x_1) B_{i11}(x'_1) \dots B_{i1r}(x'_1) C_{i11}(x_1) \dots C_{i1n}(x_1) E'_{ij1} \\ & \dots \\ & A_{i,k-1,1}(x_{k-1}) \dots A_{i,k-1,r}(x_k) B_{i,k-1,1}(x'_k) \dots B_{i,k-1,r}(x'_k) C_{i,k-1,1}(x_{k-1}) \dots C_{i,k-1,n}(x_{k-1}) E'_{i,k,k-1}, \end{aligned}$$

где каждое E'_{ij} есть E_{ik} , взятое с отрицанием или без него.

Последовательно продвигая в глубину формулы A_2 все другие предметные кванторы, получим формулу A_3 вида

$$((Q_1 Y_1) \dots (Q_r Y_r) (\bigvee_i E'_{i1} E'_{i2} \dots E'_{ik}),$$

где каждое E'_{ij} есть E_{ij} с отрицанием или без него.

Пусть одна из E_{ij} есть формула E , равная

$$\begin{aligned} & (\exists x) (Y_1(x) = a_1 \ \& \ Y_1(x+1) = b_1 \ \& \\ & \quad Y_2(x) = a_2 \ \& \ Y_2(x+1) = b_2 \ \& \\ & \quad \dots \\ & \quad Y_r(x) = a_r \ \& \ Y_r(x+1) = b_r \ \& \\ & \quad X_1(x) = c_1 \ \& \\ & \quad X_2(x) = c_2 \ \& \\ & \quad \dots \\ & \quad X_n(x) = c_n). \end{aligned}$$

Формула E утверждает, что набор предикатов (сверхслов)

$$\begin{aligned} & Y_1(0)Y_1(1) \dots Y_1(x)Y_1(x+1) \dots \\ & Y_2(0)Y_2(1) \dots Y_2(x)Y_2(x+1) \dots \\ & \dots \\ & Y_r(0)Y_r(1) \dots Y_r(x)Y_r(x+1) \dots \\ & X_1(0)X_1(1) \dots X_1(x)X_1(x+1) \dots \\ & \dots \\ & X_n(0)X_n(1) \dots X_n(x)X_n(x+1) \dots \end{aligned} \tag{13.4}$$

принадлежит множеству истинности формулы E тогда и только тогда, когда два соседних разряда x и $x+1$ высекают в сверхсловах (13.4) двухбуквенное «толстое» слово

$$(\hat{a}, \hat{b}) = (a_1 \dots a_r c_1 \dots c_n)^T (b_1 \dots b_r d_1 \dots d_n)^T$$

в алфавите Σ^{r+n} (знак « T » означает транспонирование).

Пусть H есть множество пар $(\hat{a}, \hat{b})$ всех таких двухбуквенных слов в алфавите Σ^{r+n} . Пусть

$$H_1 = \{\hat{a} : \exists \hat{b} (\hat{a}, \hat{b}) \in H\}; H_2 = \{\hat{b} : \exists \hat{a} (\hat{a}, \hat{b}) \in H\},$$

т. е. H_1 и H_2 есть проекции множества H на первую и вторую оси соответственно.

Построим макроавтомат $MA = (\Sigma^{r+n}, Q, q_0, T, F)$, поведение которого совпадает со множеством истинности $\lceil E \rceil$ формулы E . В качестве состояний макроавтомата MA возьмем множество

$$Q = \{q_{\hat{a}, \hat{b}} : (\hat{a}, \hat{b}) \in H\} \cup \{q_{\hat{a}} : \hat{a} \in H_1\} \cup \{q_0\}.$$

Функция переходов $T : Q \times \Sigma^{r+n} \rightarrow Q$ строится следующим образом:

$$T(q_0, a) = \begin{cases} q_0, & \text{если } a \notin H_1, \\ q_{\hat{a}}, & \text{если } a \in H_1; \end{cases}$$

$$T(q_a, b) = \begin{cases} q_{\hat{a}, \hat{b}}, & \text{если } \hat{b} \in H_2, \\ q_{\hat{b}}, & \text{если } \hat{b} \notin H_2 \text{ \& } \hat{b} \in H_1, \\ q_0, & \text{если } \hat{b} \notin H_2 \text{ \& } \hat{b} \notin H_1; \end{cases}$$

$$T(q_{\hat{a}, \hat{b}}, c) = q_{\hat{a}, \hat{b}}, \text{ где } c \in \Sigma^{r+n}.$$

Множество выделенных состояний $F = \{q_{\hat{a}, \hat{b}} : (\hat{a}, \hat{b}) \in H\}$. Макроавтомат MA построен; для него $\lceil E \rceil = \text{Beh}(MA)$.

Аналогично строим макроавтоматы для остальных формул E_{ij} из формулы A_3 . Далее действуем по построению формулы A_3 . Используя теорему о замкнутости класса сверхязыков, определенных макроавтоматами, относительно булевых операций и проекции, строим макроавтомат, поведение которого совпадает со множеством истинности формулы A_3 , а потому и ей эквивалентной исходной формулы A .

Требуемый макроавтомат построен.

13.5. Алгоритмическая разрешимость монадической логики

Теорема. Монадическая логика натуральных чисел алгоритмически разрешима, т. е. по всякой замкнутой формуле МЛН можно установить, истинна она или ложна.

Доказательство. Пусть A есть замкнутая формула МЛН. Приведем A к специальной префиксной форме; она имеет вид $(QX)B(X)$, где формула $B(X)$ имеет единственную свободную переменную X . По формуле $B(X)$ построим макроавтомат MB , поведение которого совпадает со множеством истинности формулы B . Если квантор (QX) есть $(\exists X)$, то вопрос об истинности или ложности формулы $(\exists X)B(X)$ эквивалентен вопросу о непустоте или пустоте сверхъязыка, представляемого макроавтоматом MB . Проблема пустоты для макроавтомата алгоритмически разрешима. Следовательно, распознаваема и проблема истинности экзистенциальной формулы $(\exists X)B(X)$. Вопрос об истинности формулы $(\forall X)B(X)$ с квантором общности сводится к аналогичному вопросу для квантора существования.

Замечание. Проблему униформизации конечноавтоматного языка в алфавите $X \times Y$ можно решить с помощью указанного алгоритма. А именно по макроавтомату A , представляющему сверхъязык L , строим формулу МЛН $A(X, Y)$, множество истинности которой совпадает со множеством L . Если истинна формула $(\forall X)(\exists Y)A(X, Y)$, то существует конечноавтоматный оператор $Y = \oplus(X)$, униформизирующий сверхъязык L . Если истинно отрицание исходной формулы, то существует конечноавтоматный оператор $X = \oplus(Y)$, униформизирующий дополнение сверхъязыка L . Конечный автомат, реализующий оператор $\oplus$, строится согласно алгоритму, приведенному в главе 13.



Темпоральная логика



В области моделирования темпоральных свойств систем лидером являются язык Promela и инструмент SPIN (<https://lwn.net/Articles/243851/>). На понимание темпоральных формализмов, используемых в таких языках и инструментах, направлена данная глава.

14.1. Пропозициональная темпоральная логика

Пропозициональная темпоральная (временная) логика (ПТЛ) есть расширение пропозициональной логики темпоральными (временными) операторами.

Формальные символы (алфавит)

1. $P, Q, R, \dots$, пропозициональные символы, возможно с индексами. Специальный дополнительный символ *start* используется только в начальный момент времени.

2. $true, false, \&, \vee, \neg, \rightarrow, \equiv$, логические символы (связки, операторы).

3. $\circ$ (следующий), $\diamond$ (существует), $\square$ (всегда), $\mathcal{U}$ (пока) есть темпоральные символы (связки, операторы).

4. $(,)$, скобка левая, запятая, скобка правая.

Замечание. Знак $\leftrightarrow$ означает: если и только если.

Формулы

1. Каждый пропозициональный символ из 1 и 2 есть (элементарная) формула.

2. Если φ и ψ есть формулы, то $(\neg\varphi)$, $(\varphi \& \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$, $(\circ\varphi)$, $(\diamond\varphi)$, $(\square\varphi)$, $(\varphi \mathcal{U} \psi)$ есть формулы.

Подформулы

1. Подформула элементарной формулы есть она сама.

2. Пусть φ есть некоторая формула. Тогда φ есть подформула формулы $(\neg\varphi)$.

3. Пусть знак $*$ $\in \{\&, \vee, \rightarrow, \equiv\}$. Пусть φ и ψ есть некоторые формулы. Тогда φ и ψ есть подформулы формулы $(\varphi * \psi)$.

Пример. Пусть $PROP = \{P, Q, R\}$ есть множество элементарных пропозициональных формул. Следующая последовательность знаков есть формулы из ПТЛ.

P, Q, R есть элементарные формулы ПТЛ. Все другие формулы ПТЛ строятся из элементарных формул:

$(\neg P), (\neg R), (\circ(\neg P)), ((\neg R) \mathcal{U}(\circ(\neg P))), (\square((\neg R) \mathcal{U}(\circ(\neg P))))$.

Формулы ПТЛ можно построить заранее и получить для нее множество $PROP$ всех ее элементарных формул.

Приоритет. Логические операторы имеют приоритет: $\neg$, $\&$, $\vee$, $\equiv$, $\rightarrow$ (каждый следующий оператор слабее предыдущего). Унарные операторы связывают сильнее, чем бинарные. Унарные операторы $\neg$, $\bigcirc$, $\square$, $\diamond$ имеют один и тот же приоритет.

Интуитивный смысл темпоральных операций. Пусть φ и ψ есть формулы.

Формула	Интуитивный смысл
$\square\varphi$	φ истинна во все будущие моменты времени
$\diamond\varphi$	φ <i>гарантированно</i> истинна в некоторый будущий момент времени
$\bigcirc\varphi$	φ истинна в следующий момент времени
$\varphi \mathcal{U} \psi$	φ истинна до тех пор, пока <i>гарантированно</i> в некоторый будущий момент времени ψ не станет истинной

Пример. 1. Всегда если послан e-mail, то он будет доставлен.

$\square(\text{послан\_e-mail} \rightarrow \diamond\text{доставлен})$

2. Если автомобиль припаркован и время парковки истекло, то в следующий момент счетчик выпишет сообщение о нарушении.

$(\text{припаркован} \& \text{время истекло}) \rightarrow (\bigcirc\text{сообщение о нарушении})$

3. Библиотека университета никогда не закрывается.

$\square\text{библиотека открыта}$

4. Всегда после получения монеты и нажатия кнопки автомат выдает кофе или чай.

$\square(\text{монета} \& \bigcirc\text{нажать\_кнопку} \rightarrow \diamond(\text{выдать\_кофе} \vee \text{выдать\_чай}))$

5. Всегда после нажатия кнопки автомат сразу выдает кофе или чай.

$\square(\text{нажать\_кнопку} \rightarrow \text{выдать\_кофе} \vee \bigcirc\text{выдать\_чай})$

6. Шлагбум закрыт, пока поезд на перекрестке.

$\text{шлагбаум\_закрыт} \mathcal{U} \text{поезд\_на\_перекрестке}$

14.2. Интерпретация формул

Определение. Пусть $PROP$ есть непустое конечное множество элементарных формул ПТЛ. *Интерпретация* для ПТЛ на данном множестве $PROP$ есть объект $I = (\mathbb{N}, \pi)$, where $\pi: \mathbb{N} \rightarrow 2^{PROP}$ есть отображение, при котором $(\forall i \in \mathbb{N})$ значение $\pi(i) = s_i = s(i) \subseteq PROP$.

Замечание. Функцию s можно задать последовательностью

$s = (s_0, s_1, s_2, \dots) = (s_i)_{i=0}^{\infty}$, где всякое $s_i \subseteq PROP$.

Пример. Пусть $PROP = \{P, Q, R\}$. Интерпретация есть, например, последовательность $s(0) = s_0 = \{P, Q\}$, $s(1) = s_1 = \{Q\}$, $s(2) = s_2 = \{P, R\}$, ...

Индукцией по построению формулы определим понятия общезначимости, выполнимости, опровержимости, невыполнимости формул ПТЛ.

Пусть интерпретация $s : \mathbb{N} \rightarrow 2^{PROP}$ есть функция из $\mathbb{N}$ во множество всех подмножеств непустого конечного множества $PROP$. Знак $\models$ означает «выполняет».

1. Элементарные формулы.

$$s_i \models P \leftrightarrow P \in s_i; s_i \not\models P \leftrightarrow P \notin s_i.$$

$$s_0 \models P, \text{ ибо } P \in s_0 = \{P, Q\}, s_0 \models Q, \text{ ибо } Q \in s_0, s_0 \not\models R, \text{ ибо } R \notin s_0.$$

$$s_1 \not\models P, \text{ ибо } P \notin s_1 = \{Q\}, s_1 \models Q, \text{ ибо } Q \in s_1, s_1 \not\models R, \text{ ибо } R \notin s_1,$$

$$s_2 \models P, \text{ ибо } P \in s_2 = \{P, R\}, s_2 \not\models Q, \text{ ибо } Q \notin s_2, s_2 \models R, \text{ ибо } R \in s_2.$$

Пусть формула $\varphi = P$. Тогда истинностное значение формулы φ на s_0 есть

$$\varphi(s_0) = P(s_0) = \text{И}, \text{ ибо } P \in s_0 = \{P, Q\}.$$

$$\varphi(s_1) = P(s_1) = \text{Л}, \text{ ибо } P \notin s_1 = \{Q\}.$$

$$\varphi(s_2) = P(s_2) = \text{И}, \text{ ибо } P \in s_2 = \{P, R\}.$$

Пусть формула $\varphi = Q$.

$$\varphi(s_0) = Q(s_0) = \text{И}, \text{ ибо } Q \in s_0 = \{P, Q\}.$$

$$\varphi(s_1) = Q(s_1) = \text{И}, \text{ ибо } Q \in s_1 = \{Q\}.$$

$$\varphi(s_2) = Q(s_2) = \text{Л}, \text{ ибо } Q \notin s_2 = \{P, R\}.$$

Пусть формула $\varphi = R$.

$$\varphi(s_0) = R(s_0) = \text{Л}, \text{ ибо } R \notin s_0 = \{P, Q\}.$$

$$\varphi(s_1) = R(s_1) = \text{Л}, \text{ ибо } R \notin s_1 = \{Q\}.$$

$$\varphi(s_2) = R(s_2) = \text{И}, \text{ ибо } R \in s_2 = \{P, R\}.$$

Высказывание *start* истинно только в начальный момент времени.

$$s_i \models \text{start} \leftrightarrow (i = 0).$$

2. Формулы логики высказываний с операциями $\neg$, $\&$, $\vee$, $\rightarrow$, $\equiv$.

$$s_i \models \neg\varphi \equiv s_i \not\models \varphi.$$

$$s_i \models \varphi \& \psi \leftrightarrow (s_i \models \varphi) \& (s_i \models \psi).$$

$$s_i \models (\varphi \vee \psi) \leftrightarrow (s_i \models \varphi) \vee (s_i \models \psi).$$

$$s_i \models (\varphi \rightarrow \psi) \leftrightarrow \text{если } (s_i \models \varphi), \text{ то } (s_i \models \psi).$$

3. Формулы темпоральной логики с операциями $\bigcirc$, $\square$, $\diamond$, $\mathcal{U}$.

$$s_i \models \bigcirc\varphi \leftrightarrow s_{i+1} \models \varphi.$$

$$s_i \models \square\varphi \leftrightarrow \forall j(j \geq i \rightarrow s_j \models \varphi).$$

$$s_i \models \diamond\varphi \leftrightarrow \exists j(j \geq i \& s_j \models \varphi).$$

$$s_i \models (\varphi \mathcal{U} \psi) \leftrightarrow \exists j(j \geq i) \& (s_j \models \psi) \& \forall k(i \leq k < j \rightarrow (s_k \models \varphi)).$$

Существование j гарантировано.

14.3. Общезначимость, выполнимость, опровержимость, невыполнимость

Определение. Пусть φ есть формула в ПТЛ и интерпретация

$$s = (s_0, s_1, s_2, \dots).$$

s *выполняет* φ , если $(\forall i) s_i \models \varphi$. Интерпретация s *опровергает* φ (или $s \not\models \varphi$), если $(\exists i) s_i \not\models \varphi$.

Обозначение. s *выполняет* φ , $s \models \varphi$, $\varphi(s) = 1$.

Определение. Пусть φ есть формула в ПТЛ и конечное множество пропозициональных переменных $PROP$.

1. Формула φ *общезначима* (тождественно истинна) в $PROP$, если всякая интерпретация s в $PROP$ выполняет формулу φ .

2. Формула φ *выполнима* в $PROP$, если существует интерпретация s в $PROP$, которая выполняет формулу φ .

3. Формула φ *опровержима* в $PROP$, если существует интерпретация s в $PROP$, которая выполняет формулу φ .

4. Формула φ *невыполнима* (тождественно ложна, противоречива) в $PROP$, если всякая интерпретация s в $PROP$ опровергает формулу φ .

Замечание. Пусть φ есть формула в ПТЛ.

1. Формула φ *общезначима* (тождественно истинна), если $(\forall s) s \models \varphi$ (или $(\forall s)(\forall i) s_i \models \varphi$).

2. Формула φ *выполнима*, если $(\exists s) s \models \varphi$ (или $(\exists s)(\forall i) s_i \models \varphi$).

3. Формула φ *опровержима*, если $(\exists s) s \not\models \varphi$ (или $(\exists s)(\exists i) s_i \not\models \varphi$).

4. Формула φ *невыполнима* (тождественно ложна, противоречива), если $(\forall s) s \not\models \varphi$ (или $(\forall s)(\forall i) s_i \not\models \varphi$).

Замечание. Пусть $PROP$ есть конечное множество пропозициональных переменных с n переменными и 2^{PROP} есть множество всех подмножеств множества $PROP$. Мощность $PROP$ есть число n элементов в $PROP$. Мощность 2^{PROP} равна 2^n . Мощность $2^{(2^{PROP})} = 2^{(2^n)}$.

Пример. Пусть $PROP = \{p, q, r\}$, $n = |PROP| = 3$, множество 2^{PROP} всех $2^n = 2^3 = 8$ подмножеств множества $PROP$ есть множество

$$2^{PROP} = \{\emptyset, \{p\}, \{q\}, \{r\}, \{p, q\}, \{p, r\}, \{q, r\}, \{p, q, r\}\}.$$

Обозначим значения *true* и *false* через 1 и 0 соответственно.

1. Найдём значения формулы

$$\varphi = p \ \& \ \neg q \rightarrow \neg r$$

на всех множествах из 2^{PROP} .

Если c_i есть некоторое множество из 2^{PROP} , то значение

$$\varphi(c_i) = (c_i \models \varphi) = (p \in c_i) \ \& \ \neg(q \in c_i) \rightarrow \neg(r \in c_i).$$

Например,

$$\varphi(\emptyset) = (p \in \emptyset) \& \neg(q \in \emptyset) \rightarrow \neg(r \in \emptyset) = 0 \& \neg 0 \rightarrow \neg 0 = 0 \& 1 \rightarrow 1 = 1,$$

$$\varphi(\{p, r\}) = (p \in \{p, r\}) \& \neg(q \in \{p, r\}) \rightarrow \neg(r \in \{p, r\}) = 1 \& \neg 0 \rightarrow \neg 1 = 1 \& 1 \rightarrow 0 = 0.$$

Все значения формулы φ на множествах из 2^{PROP} приведены в следующей табл. 13.1.



Таблица 13.1

	pqr	2^{PROP}	$p \in c_i$	$q \in c_i$	$r \in c_i$	$c_i \models \varphi$
0	000	$c_0 = c_\emptyset = \emptyset$	0	0	0	1
1	001	$c_1 = c_r = \{r\}$	0	0	1	1
2	010	$c_2 = c_q = \{q\}$	0	1	0	1
3	011	$c_3 = c_{qr} = \{q, r\}$	0	1	1	1
4	100	$c_4 = c_p = \{p\}$	1	0	0	1
5	101	$c_5 = c_{pr} = \{p, r\}$	1	0	1	0
6	110	$c_6 = c_{pq} = \{p, q\}$	1	1	0	1
7	111	$c_7 = c_{pqr} = \{p, q, r\}$	1	1	1	1

Всякая последовательность $s = (s_0, s_1, \dots, s_i, s_{i+1}, \dots)$ состоит из подмножеств множества 2^{PROP} . Число таких подмножеств равно $2^{(2^n)} = 2^{(2^3)} = 2^8 = 256$.

2. Найдем значения формулы

$$\varphi = p \& \neg q \rightarrow \neg r$$

на некоторых последовательностях множеств из 2^{PROP} .

Пусть последовательность $s1 = (s1_0, s1_1, \dots, s1_i, s1_{i+1}, \dots)$ построена из множеств c_q, c_{qr} и каждое из них представлено в $s1$. Это значит, что для всякого i элемент $s1_i \in \{c_q, c_{qr}\}$. Так как $(c_q \models \varphi) = 1, (c_{qr} \models \varphi) = 1$, то $(s1 \models \varphi) = 1$, ибо $\forall i \in \mathbb{N} (s_i \models \varphi) = 1$. Так как $s1 \models \varphi, \varphi(s1) = 1$, то последовательность $s1$ выполняет φ , формула φ выполняется.

Пусть последовательность $s2 = (s2_0, s2_1, \dots, s2_i, s2_{i+1}, \dots)$ построена из множеств c_p, c_{pr}, c_{pr} и каждое из них представлено в $s2$. Это значит, что для всякого i элемент $s2_i \in \{c_p, c_{pr}, c_{pr}\}$. Так как $(c_p \models \varphi) = 1, (c_{pr} \models \varphi) = 0, (c_{qr} \models \varphi) = 1$, то $(s2 \models \varphi) = 0$, ибо $(\exists s2_i = s_{pr})((s2_i \models \varphi) = 0)$.

Так как $s2 \not\models \varphi$, то последовательность $s2$ опровергает (не выполняет) φ , и формула φ опровержима.

О формуле φ можно сказать следующее.

- 1) формула φ выполнима, ибо $s1 \models \varphi$;
- 2) формула φ опровержима, ибо $s2 \not\models \varphi$;
- 3) формула φ не общезначима, ибо $s2 \not\models \varphi$;
- 4) формула φ не тождественно ложна, ибо $s1 \models \varphi$.

3. Найдем значения формулы

$$\psi = \bigcirc \varphi = \bigcirc(p \& \neg q \rightarrow \neg r)$$

на некоторых последовательностях множеств из 2^{PROP} .

Напомним, что для последовательности $s = (s_0, s_1, \dots, s_i, s_{i+1}, \dots)$ множество

$$s_i \models \bigcirc \varphi \leftrightarrow s_{i+1} \models \varphi.$$

Пусть последовательность $s1 = (s1_0, s1_1, \dots, s1_i, s1_{i+1}, \dots)$ построена из множеств c_q, c_{qr} и каждое из них представлено в $s1$. Множество

$$s1_i \models \psi \leftrightarrow s1_i \models \bigcirc \varphi \leftrightarrow s1_{i+1} \models \varphi.$$

Так как $(\forall i)(s1_i \models \varphi) = 1$, то $(\forall i)(s1_i \models \psi) = 1$. Поэтому последовательность $s1$ выполняет ψ , то есть $s1 \models \psi$, $\psi(s1) = 1$.

Пусть последовательность $s2 = (s2_0, s2_1, \dots, s2_i, s2_{i+1}, \dots)$ построена из множеств c_p, c_{qr}, c_{pr} , каждое из них представлено в $s2$, и c_{pr} представлено в $s2$ не менее двух раз. Для формулы φ значение $(c_{pr} \models \varphi) = 0$, то есть $c_{pr} \not\models \varphi$. Последовательность $s2$ имеет такое $i > 0$, для которого $s2_i = c_{pr}$, и потому

$$s2_i \not\models \varphi \leftrightarrow s2_{(i-1)+1} \not\models \varphi \leftrightarrow s2_{(i-1)} \not\models \bigcirc \varphi \leftrightarrow s2_{(i-1)} \not\models \psi, \psi(s2_{(i-1)}) = 0,$$

$$(\exists i)(s2_i \not\models \psi), (\exists s)(\exists i)(s_i \not\models \psi),$$



последовательность $s2$ опровергает (не выполняет) ψ , то есть формула ψ опровержима.

О формуле ψ можно сказать следующее.

- 1) формула ψ выполнима, ибо $s1 \models \psi$;
 - 2) формула ψ опровержима, ибо $s2 \not\models \psi$;
 - 3) формула ψ не общезначима, ибо $s2 \not\models \psi$;
 - 4) формула ψ не является невыполнимой, ибо $s2 \models \psi$.
4. Найдем значения формулы

$$\psi = \Box \varphi = \Box(p \ \& \ \neg q \rightarrow \neg r)$$

на некоторых последовательностях множеств из 2^{PROP} .

Напомним, что если последовательность $s = (s_0, s_1, \dots, s_i, \dots, s_j, \dots)$, то

$$s_i \models \Box \varphi \leftrightarrow (\forall j)_{j \geq i}(s_j \models \varphi).$$

Пусть последовательность $s1 = (s1_0, s1_1, \dots, s1_i, \dots, s1_j, \dots)$ построена из множеств c_q, c_{qr} и каждое из них представлено в $s1$. Множество

$$s1_i \models \psi \leftrightarrow s1_i \models \Box \varphi \leftrightarrow (\forall j)_{j \geq i}(s1_j \models \varphi).$$

Так как $(\forall i)(s1_i \models \varphi)$, то $(\forall i)(\forall j)_{j \geq i}(s1_j \models \varphi)$, $(\forall i)(s1_i \models \Box \varphi)$, $s1 \models \Box \varphi$, $s1 \models \psi$, $\varphi(s1) = 1$, $s1$ выполняет ψ , формула ψ выполнима.

Пусть последовательность $s2 = (s2_0, s2_1, \dots, s2_i, \dots, s2_j, \dots)$ построена из множеств c_p, c_{qr}, c_{pr} и каждое из них представлено в $s2$. Ранее было получено, что $(c_{pr} \models \varphi) = 0$, то есть $c_{pr} \not\models \varphi$. Множество

$$s2_i \not\models \psi \leftrightarrow s2_i \not\models \Box \varphi \leftrightarrow \neg (\forall j)_{j \geq i}(s2_j \models \varphi) \leftrightarrow (\exists j)_{j \geq i}(s2_j \not\models \varphi).$$

Последовательность $s2$ имеет i , для которого $s2_i = c_{pr}$, и потому $s2_i \not\models \varphi$. Тогда $(\exists j)_{j \geq i}(s2_j \not\models \varphi)$, откуда $(\exists j)_{j \geq i}(s2_j \not\models \varphi)$. Поэтому $s2_i \not\models \Box \varphi$, $s2_i \not\models \psi$, $\psi(s2_i) = 0$, $(\exists i)(s2_i \not\models \psi)$, $s2 \not\models \psi$, $\psi(s2) = 0$, $s2$ не выполняет ψ , формула ψ опровержима.

О формуле ψ можно сказать следующее.

- 1) формула ψ выполнима, ибо $s1 \models \psi$;
- 2) формула ψ опровержима, ибо $s2 \not\models \psi$;
- 3) формула ψ не общезначима, ибо $s2 \not\models \psi$;
- 4) формула ψ не является невыполнимой, ибо $s2 \models \psi$.
5. Найдем значения формулы

$$\psi = \Diamond \varphi = \Diamond(p \ \& \ \neg q \rightarrow \neg r)$$

на некоторых последовательностях множеств из 2^{PROP} .

Напомним, что если последовательность $s = (s_0, s_1, \dots, s_i, \dots, s_j, \dots)$, то

$$s_i \models \Diamond \varphi \leftrightarrow (\exists j)_{j \geq i} (s_j \models \varphi).$$

$$s_i \not\models \Diamond \varphi \leftrightarrow (\forall j)_{j \geq i} (s_j \not\models \varphi).$$



Пусть последовательность $s1 = (s1_0, s1_1, \dots, s1_i, \dots, s1_j, \dots)$ построена из множеств c_p, c_{qr} и каждое из них представлено в $s1$. Множество

$$s1_i \models \psi \leftrightarrow s1_i \models \Diamond \varphi \leftrightarrow (\exists j)_{j \geq i} (s_j \models \varphi).$$

Так как $(\forall i)(s_j \models \varphi)$, то $(\forall i)(\forall j)_{j \geq i} (s_j \models \varphi)$, $(\forall i)(\exists j)_{j \geq i} (s_j \models \varphi)$, $(\forall i)(s1_i \models \Diamond \varphi)$, $s1 \models \Diamond \varphi$, $s1 \models \psi$, $\varphi(s1) = 1$. Последовательность $s1$ выполняет формулу ψ , то есть формула ψ выполнима.

Пусть последовательность $s2 = (s2_0, s2_1, \dots, s2_i, \dots, s2_j, \dots)$ построена из множеств c_p, c_{qr}, c_{pr} и каждое из них представлено в $s2$. Ранее мы имели $(c_{pr} \models \varphi) = 0$, то есть $c_{pr} \not\models \varphi$. Тогда $s2_i = c_{pr}$, $s2_i \not\models \varphi$ для некоторого i . Поэтому $(\exists j)_{j \geq i} (s2_j \not\models \varphi)$. Так как $s2_i \not\models \psi \leftrightarrow s2_i \not\models \Diamond \varphi \leftrightarrow \neg (\exists j)_{j \geq i} (s2_j \models \varphi) \leftrightarrow (\exists j)_{j \geq i} (s2_j \not\models \varphi)$ и последнее верно, ибо $s2_i \not\models \psi$.

Последовательность $s2$ имеет такое i , для которого $s2_i = c_{pr}$ и потому для некоторого $s2_i \not\models \varphi$. Тогда $(\exists j)_{j=i} (s_j \not\models \varphi)$, и потому $(\exists j)_{j \geq i} (s_j \not\models \varphi)$. Поэтому $s2_i \not\models \Diamond \varphi$, $s2_i \not\models \psi$, $(\exists i)(s2_i \not\models \psi)$, $s2 \not\models \psi$, $\psi(s2) = 0$.

О формуле ψ можно сказать следующее.

- 1) формула ψ выполнима, ибо $s1 \models \psi$;
- 2) формула ψ опровержима, ибо $s2 \not\models \psi$;
- 3) формула ψ не общезначима, ибо $s2 \not\models \psi$;
- 4) формула ψ не является тождественно ложной, ибо $s2 \models \psi$.

Замечание. Пусть φ и ψ есть формулы в ПТЛ. Напомним, что если последовательность $s = (s_0, s_1, \dots, s_i, \dots, s_j, \dots)$, то

$$s_i \models \varphi \ \mathcal{U} \ \psi \leftrightarrow (\exists j)_{j \geq i} (s_j \models \psi) \ \& \ (\forall k)_{i \leq k < j} (s_i \models \varphi).$$

14.4. Другие темпоральные операторы

Unless оператор $\mathcal{W}$ (негарантированное пока). Формула $\varphi \ \mathcal{W} \ \psi$ означает, что φ истинна до тех пор, пока негарантированно в некоторый будущий момент времени ψ не станет истинной (то есть возможен случай, когда ψ никогда не ста-

нет истинной). Разница между операторами *Until* и *Unless* в том, что в первом случае ψ гарантированно будет истинной в будущем, а в последнем случае ψ может никогда не стать истинной. Поэтому при $\varphi \mathcal{W} \psi$ возможно: 1) ψ станет истинной в некоторый будущий момент, и тогда $\mathcal{W}$ идентичен оператору *Until*, и 2) формула ψ никогда не станет истинной, и тогда φ всегда истинна. Оператор $\mathcal{W}$ определяется через ранее определенные операторы $\mathcal{U}$ и $\Box$ с помощью исключающего «или» следующим образом:

$$s_i \models \varphi \mathcal{W} \psi \leftrightarrow s_i \models \varphi \mathcal{U} \psi \text{ XOR } s_i \models \Box \varphi.$$

Оператор $\mathcal{W}$ выражается через операторы $\mathcal{U}$, $\Box$, XOR.

Since оператор $\mathcal{S}$ (с тех пор как). Формула $\varphi \mathcal{S} \psi$ (φ since ψ) означает, что формула ψ стала в прошлом истинной, и формула φ с тех пор тоже стала истинной. Тогда

$$s_i \models \varphi \mathcal{S} \psi \leftrightarrow \exists j(0 \leq j < i \text{ \& } s_j \models \psi \text{ \& } \forall k(k \geq j \rightarrow s_k \models \Box \varphi)).$$

Оператор $\mathcal{S}$ выражается через операторы $\Box$, $\&$, $\rightarrow$.

Release оператор $\mathcal{R}$ (отпускает, освобождает). Формула $\varphi \mathcal{R} \psi$ (φ отпускает ψ) означает, что ψ гарантированно становится истинной, когда φ впервые гарантированно стала истинной. Если φ никогда не станет истинной, то ψ должна остаться истинной навсегда. Если φ становится истинной в состоянии s_i , то ψ не истинна от состояния s_{i+1} , но мы не можем что-то утверждать о ψ после состояния s_i . Отметим, что оператор *Release* обратен оператору *Unless*.

$$s_i \models \varphi \mathcal{R} \psi \leftrightarrow \exists i(\forall k(0 \leq k < i \rightarrow (s_k \not\models \varphi) \text{ \& } (s_k \models \psi))) \vee \forall j(s_j \models \psi).$$

Пример. Следующие формулы поясняют вышеприведенную семантику.

1) *уровень_воды_высок* $\rightarrow$ *звонит_звонок*: если уровень воды высок в состоянии s_i , $i \geq 0$, то в состоянии s_{i+1} звонит предупреждающий звонок;

2) $\neg(\Box(\text{много\_знать} \text{ \& } \text{отличаться\_в\_спорте}))$: не всегда возможно иметь высокий уровень знаний, и отличаться в спорте не всегда возможно;

3) *система_упала* $\rightarrow$ *система_поднялась*: если система упала до состояния s_i , то в некотором будущем состоянии s_j , $j \geq i$, система возвращается в прежнее состояние;

4) *начало_файла* $\&$ *принтер_включен* $\rightarrow$ (*старт_печати* $\&$ *печать* $\mathcal{U}$ *конец_файла*): если принтер включен и поступает начало файла в состоянии s_i , то принтер начинает печатать файл при некотором будущем состоянии s_j , $j \geq i$, и продолжит печать, пока не достигается конец файла при некотором состоянии k , $k > j$;

5) *исполнять_программу* $\mathcal{W}$ *стек_не_переполнится*: исполнение программы продолжается, пока не переполнится стек;

6) *пересекать_шлагбаум* $\mathcal{S}$ *шлагбаум_открыт*: шлагбаум остается открытым с тех пор, как поезд его прошел;

7) *прибытие_на_вокзал* $\mathcal{R}$ *машинист_локомотива*: после прибытия поезда на вокзал машинист локомотива был отпущен.



14.5. Аксиоматическая система

Зададим формальную аксиоматическую систему описания тождественно истинных формул в ПТЛ.

Формальные символы

1. $P, Q, R, \dots$, пропозициональные символы, возможно с индексами. Специальный дополнительный символ *start* используется только в начальный момент времени.

2. *true, false*, $\&, \vee, \neg, \rightarrow$ есть логические символы.

3. $\circ, \diamond, \square, \mathcal{U}$ есть темпоральные символы.

4. $(,)$ есть скобка левая, запятая, скобка правая.

Формулы

1. Каждый пропозициональный символ из 1 и 2 есть (элементарная) формула.

2. Если ϕ и ψ есть формулы, то $(\neg\phi)$, $(\phi \& \psi)$, $(\phi \vee \psi)$, $(\phi \rightarrow \psi)$, $(\circ\phi)$, $(\diamond\phi)$, $(\square\phi)$, $(\phi \mathcal{U} \psi)$ есть формулы.

Схемы аксиом

Пусть ϕ и ψ есть произвольные формулы из ПТЛ. Множество аксиом для ПТЛ задается группами следующих формул.

1. Аксиомы абсорбции для удаления некоторых символов.

$$\diamond\square\diamond\phi \equiv \diamond\square\phi; \square\diamond\square\phi \equiv \diamond\square\phi.$$

2. Аксиомы дистрибутивности.

$$\diamond(\phi \vee \psi) \equiv \diamond\phi \vee \diamond\psi; \square(\phi \& \psi) \equiv \square\phi \& \square\psi;$$

$$\circ(\phi \& \psi) \equiv \circ\phi \& \circ\psi; \circ(\phi \mathcal{U} \psi) \equiv \circ\phi \mathcal{U} \circ\psi.$$

3. Аксиомы двойственности.

$$\neg\square\phi \equiv \diamond\neg\phi; \neg\diamond\phi \equiv \square\neg\phi; \neg\circ\phi \equiv \circ\neg\phi.$$

4. Аксиомы распространения (*expansion*).

$$\square\phi \equiv \phi \& \circ\square\phi; \diamond\phi \equiv \phi \vee \circ\diamond\phi; \phi \mathcal{U} \psi \equiv \psi \vee (\phi \& \circ(\phi \mathcal{U} \psi)).$$

5. Аксиомы идемпотентности.

$$\square\square\phi \equiv \square\phi; \diamond\diamond\phi \equiv \diamond\phi; \phi \mathcal{U} (\phi \mathcal{U} \psi) \equiv \phi \mathcal{U} \psi; (\phi \mathcal{U} \psi) \mathcal{U} \psi \equiv \phi \mathcal{U} \psi.$$

6. Аксиома индукции.

$$\vdash \square(\phi \rightarrow \circ\phi) \rightarrow (\phi \square\phi).$$

Замечание. Каждая аксиома есть тождественно истинная формула.

Правила вывода

1. *Правило заключения* (ПЗ). Если A и $A \rightarrow B$ есть доказуемые формулы, то формула B тоже доказуема.

2. *Правило эквивалентности* (ПЭ). Если $A(B)$ и $B \equiv C$ есть доказуемые формулы, то формула $A(C)$ тоже доказуема.

Определение. Аксиоматическая система для ПТЛ состоит из формальных символов, формул, аксиом и правил вывода.

Определение. Доказательство есть конечная последовательность формул, каждая из которых есть либо аксиома, либо получена из предыдущих формул последовательности по правилу заключения или по правилу эквивалентности.

Определение. Формула A доказуема, если существует доказательство, последняя формула которого есть A .

Обозначение. $\vdash A$ означает, что формула A доказуема.

Теорема. (О. Lichtenstein, A. Pnueli.) Пусть φ есть формула из ПТЛ. Тогда $\vdash \varphi \leftrightarrow \varphi$ тождественно истинна.

14.6. Спецификация свойств формулами

Покажем удобство пропозициональной темпоральной логики для характеристики свойств безопасности, живучести, справедливости на следующих примерах.

Пример. Рассмотрим проблему взаимной исключаемости для двух процессов P_1 и P_2 . Процесс состоит из 1) критической области, 2) некритической области, 3) области ожидания вхождения в критическую область. Пусть $wait_i$ означает, что процесс P_i находится в фазе ожидания, а $crit_i$ означает, что P_i находится в критической области. Свойство *безопасности*, утверждающее, что P_1 и P_2 никогда одновременно не имеют доступа в их критические области, выражается формулой

$$\Box(\neg crit_1 \vee \neg crit_2).$$

Свойство *живучести*, утверждающее, что каждый процесс P_i , бесконечно часто попадающий в критическую область, выражается формулой

$$(\Box \Diamond crit_1) \& (\Box \Diamond crit_2).$$

Свойство *справедливости* (fairness) состоит в том, что каждый ожидающий процесс со временем войдет в его критическую область. Он может быть выражен следующей формулой:

$$(\Box \Diamond wait_1 \rightarrow \Box \Diamond crit_1) \& (\Box \Diamond wait_2 \rightarrow \Box \Diamond crit_2).$$

Пример. Описываются темпоральные свойства перемены огней светофоров уличного движения R (красный), G (зеленый), Y (желтый, yellow). Мы можем рассмотреть «мгновенные» (атомарные) фазы. То есть огонь светофора в фазе R перейдет к следующей фазе G , и между R и G нет промежуточных огней. Нас не заботит продолжительность горящего красного света перед переходом к желтому свету. С этим допущением формула, выражающая переход от состояния (фазы) R к состоянию (фазе) G , имеет вид

$$\Box(R \rightarrow \bigcirc G).$$

Однако если мы хотим учесть *продолжительности*, то огонь светофора должен на некоторое время оставаться красным, прежде чем стать зеленым. Так как неизвестно, как долго свет остается красным, можно лишь утверждать, что заключительно (в конце концов) свет снова станет зеленым.

$$\Box(R \rightarrow \Diamond G).$$

Если требуются подробности переключений света, то можно сказать: «будучи зеленым, свет, наконец, становится красным, прежде пребывания некоторое время желтым».

$$\Box(G \rightarrow \bigcirc(G \cup (Y \& \bigcirc(Y \cup R)))).$$

Зеленый свет светофора повторяется бесконечно много раз.

$$\Box \Diamond G.$$

Красный свет светофора возможен лишь в конечном числе случаев.

$$\Diamond \Box \neg R.$$

14.7. Спецификация взаимодействия и параллелизма

Рассмотрим методы спецификации для взаимодействия и параллелизма и проиллюстрируем их при описании реактивных систем. Рассмотрим систему с несколькими объектами (компонентами, или агентами), которые могут взаимодействовать между собой при достижении своих целей. Существенно, что каждый объект независим от другого, исключая случай, когда этого требует ресурс или информация от другого объекта в порядке продолжения его активности. В дискретно-временной модели все компоненты имеют общие глобальные часы и одно и то же определение следующего момента.

Положим, $Spec_A$ и $Spec_B$ соответственно означают спецификации компонент A и B , $(\mathbb{N}, \pi_1)$ означает интерпретацию для $Spec_A$, и $(\mathbb{N}, \pi_2)$ означает интерпретацию для $Spec_B$.

Параллелизм. Исполнение компонент идет независимо и одновременно, проходя одну и ту же последовательность состояний. Этот тип параллелизма называется *синхронным*. Нет никакого взаимодействия между высказываниями/формулами в $Spec_A$ и высказываниями/формулами $Spec_B$. Спецификация объединенных систем, которые параллельны и синхронны, есть $Spec_A \& Spec_B$, интерпретация которой есть $(\mathbb{N}, \pi)$, where $\pi(i) = \pi_1(i) \& \pi_2(i)$, для $i \in \mathbb{N}$.

Пример.

$$Spec_A = \Box[(start \rightarrow P) \& (P \rightarrow \bigcirc \bigcirc Q) \& (Q \rightarrow \bigcirc P)],$$

$$Spec_B = \Box[(start \rightarrow A) \& (A \rightarrow \bigcirc B) \& (B \rightarrow \bigcirc \bigcirc A)].$$

Интерпретация для $Spec_A$ есть $(\mathbb{N}, \pi_1)$, интерпретация для $Spec_B$ is $(\mathbb{N}, \pi_2)$, где

$$\pi_1(i) = \begin{cases} P, & \text{если } i = 3k, k \geq 0; \\ Q, & \text{если } i = 3k + 2, k \geq 0, \end{cases} \quad \pi_2(i) = \begin{cases} A, & \text{если } i = 3k, k \geq 0; \\ B, & \text{если } i = 3k + 1, k \geq 0. \end{cases}$$

Интерпретация для $Spec_A$ & $Spec_B$ есть $(\mathbb{N}, \pi)$, где

$$\pi(i) = \begin{cases} A \& P, & \text{если } i = 3k, k \geq 0; \\ B, & \text{если } i = 3k + 1, k \geq 0; \\ Q, & \text{если } i = 3k + 2, k \geq 0. \end{cases}$$



Чередующиеся исполнения. Два компонента исполняются независимо; однако только один компонент может исполняться в течение некоторого периода времени. Интерпретация $(\mathbb{N}, \pi)$ этого поведения задается равенством:

$$\pi(i) = \begin{cases} \pi_1(i), & \text{если } A \text{ выполняется;} \\ \pi_2(i), & \text{если } B \text{ выполняется.} \end{cases}$$

Исполнения, вызываемые сообщениями. Прохождение сообщения можно специфицировать одним из трех способов.

1. *Пиринговое взаимодействие:* объект A посылает сообщение *непосредственно* к объекту B , минуя центральный сервер.

2. *Радиоподобное взаимодействие:* некоторый объект посылает сообщение нескольким объектам, не зная получателя сообщения.

3. *Мультирадиоподобное взаимодействие:* ограниченная форма широковещания, которая специфицирует ограничитель, базируемое на котором множество получателей характеризуется. Посылатель фактически не знает получателей сообщения. Сообщения посылаются через каналы со следующими характеристиками:

- (а) канал *однонаправленный*,
- (б) канал не сохраняет сообщений,
- (с) сообщения доставляются в порядке поступления.

Чтобы специфицировать взаимодействие между компонентами A и B с помощью проходящих сообщений, необходимы два канала, один – для компонента A посылать сообщения компоненту B , а другой – для компонента A получать сообщения от компонента B . Вы увидите, что множество высказываний в каждой спецификации может быть поделенным на три множества: *входное, выходное, внешнее*. Высказывание $P \in output_A$ и высказывание $\bar{P} \in input_B$ соединяются связкой $\Rightarrow$, чтобы показать направление потока сообщений от компонента A к компоненту B . Аналогично высказывание $Q \in output_B$ и высказывание $\bar{Q} \in input_A$ соединяются связкой $\Rightarrow$, чтобы показать направление потока сообщений от компонента B к компоненту A . В зависимости от типа взаимодействия критерий качества обслуживания (QoS, Quality of Service) и среды взаимодействия сообщение доставляется одним из следующих способов.

1. Сообщение доставлено немедленно:

$$\Box[P \Rightarrow \bar{P}].$$

2. Сообщение доставлено в следующий момент:

$$\Box[P \Rightarrow \bigcirc \bar{P}].$$

3. Сообщение доставлено в некоторый будущий момент:

$$\Box[P \Rightarrow \Diamond \bar{P}].$$

Существуют другие возможности, например такие, как «сообщение доставлено в некоторое *особое* время» или как «сообщение доставлено с некоторым интервалом времени», которые здесь не рассматриваются.

В следующем примере, $send\_msg \in output_A$ и $rcv\_msg \in input_B$, спецификация взаимодействия утверждает, что сообщение $send\_msg$ послано компонентом A к компоненту B . Когда оно получено в следующий момент времени, высказывание rcv_msg будет истинным.

Пример.

$$Spec_A = \Box[(start \rightarrow P) \ \& \ (P \rightarrow \bigcirc\bigcirc Q) \ \& \ (Q \rightarrow \bigcirc P) \ \& \ (Q \rightarrow send\_msg)],$$

$$Spec_B = \Box[(rcv\_msg \rightarrow A) \ \& \ (A \rightarrow \bigcirc B) \ \& \ (B \rightarrow \bigcirc\bigcirc A)].$$

Спецификация взаимодействия есть

$$Spec_{AtoB} : \Box[send\_msg \rightarrow \bigcirc rcv\_msg].$$

Спецификация работы системы есть

$$Spec_A \ \& \ Spec_B \ \& \ Spec_{AtoB}.$$

В связи с используемой моделью эта спецификация работы системы лучшая. Компонент A *делит совместно используемую память, совместно используемое запоминающее устройство (например, связанное одновременно с несколькими процессорами)* исполняется, стартуя в начальном состоянии. В начальном состоянии P истинно. Вторая строка в $Spec_A$ обеспечивает то, что Q становится истинным во временной точке 2. Третья и четвертая строки в $Spec_A$ утверждают, что высказывание $send\_msg$ делается истинным во временных точках 2 и P делается истинным во временной точке 3. С этого момента P истинно во временных точках 6, 9, ...; Q истинно во временных точках 5, 8, ..., $send\_msg$ истинно, когда Q истинно. Спецификация взаимодействия $send\_msg \rightarrow \bigcirc rcv\_msg$ обеспечивает то, что высказывание rcv_msg становится истинным в момент после того, как $send\_msg$ становится истинным. С тех пор как высказывание $send\_msg$ стало истинным в моменты времени 2, 5, ..., высказывание rcv_msg становится истинным в моменты времени 3, 6, ... Первая строка в $Spec_B$ обеспечивает то, что высказывание A истинно всякий раз, когда высказывание rcv_msg истинно, вторая строка в $Spec_B$ утверждает, что высказывание B истинно в момент после того, как высказывание A стало истинным. Последняя строка в $Spec_B$ утверждает, что описанное выше поведение повторяется. Поэтому высказывание A истинно в моменты времени 3, 6, ..., и высказывание B истинно в моменты времени 4, 7, ... Заметим, что в объединенной спецификации высказывания $send\_msg$ и rcv_msg не появляются. То есть высказывания во множествах $output_A$

и $input_B$ «сбалансированы», оставляя объединенное поведение описывающим лишь высказывания в $innerA$ и $innerB$. Интерпретации спецификаций $Spec_A$, $Spec_B$, $Spec_A \& Spec_B$ & $Spec_{AtoB}$ приводятся ниже.

$$\text{Модель для } Spec_A \text{ есть } \pi_A(i) = \begin{cases} A, & \text{если } i = 3k, k \geq 0; \\ Q & \text{если } i = 3k + 2, k \geq 0; \\ send\_msg, & \text{если } i = 3k + 2, k \geq 0. \end{cases}$$

$$\text{Модель для } Spec_B \text{ есть } \pi_B(i) = \begin{cases} A, & \text{если } i = 3k, k \geq 1; \\ B, & \text{если } i = 3k + 1, k \geq 1; \\ rcv\_msg, & \text{если } i = 3k + 1, k \geq 1. \end{cases}$$

$$\text{Модель для } (Spec_A \& Spec_B \& Spec_{AtoB}) \text{ есть } \pi(i) = \begin{cases} P, & \text{если } i = 0; \\ P \& A, & \text{если } i = 3k, k \geq 1; \\ Q, & \text{если } i = 3k - 1, k \geq 1; \\ B, & \text{если } i = 3k + 1, k \geq 1. \end{cases}$$



Глава 15

.....

Аксиоматический язык программирования OBJ3

OBJ3 есть функциональный язык программирования высокого уровня и широкого спектра. Развитая в OBJ3 модульная система программ внедрена в языках программирования массового использования Ada, ML, C++, Lotoc. Параметризация из OBJ имплементирована в C#. В основе OBJ3 лежат аксиоматически заданные алгебраические теории. Иногда OBJ3 называют аксиоматическим языком программирования. Аксиоматический подход в языках программирования значительно облегчает задачу верификации программ. Аксиоматические языки программирования могут служить платформой для создания других языков программирования.

Аксиоматические языки программирования проходят период количественных накоплений в том новом направлении, которое в будущем обещает привести к качественным изменениям в теории и практике аксиоматизации компьютерных программ.

15.1. Описание языка

OBJ3 базируется на порядковой сортовой логике равенства, построенной на формализме логики предикатов первого порядка. OBJ3 допускает модульное развитие. Эта цель достигается наличием трех программных единиц языка: *объект (object)*, *теория (theory)*, *вью (view, проекция)*. Объект инкапсулирует исполняемый код. Теория не исполнима. Она лишь (аксиоматически) определяет свойства, которые могут удовлетворяться другой теорией или объектом. *Модуль* есть объект или теория. Программная единица вью обеспечивает связь между теорией и модулем. Работа системы OBJ3 основана на выводе алгебраических термов из аксиом. Правила вывода есть аксиомы программы. Каждая аксиома есть равенство двух термов.

Основная программная единица в OBJ3 есть *объект (object)*. Синтаксически определение объекта начинается с ключевого слова `obj` и заканчивается ключевым словом `endo`. Идентификатор объекта появляется сразу после ключевого слова `obj`. Ключевое слово `is` следует за именем. Затем следует тело объекта.

15.2. Спецификация объекта

Спецификация объекта включает пять следующих определений.

1. *Оъявление сортов.* Объект (*module*) с именем `FLAVORS` и сорт (тип) с именем `Flavor` вводятся в первой строке. Имя объекта пишется большими буквами. Имя сорта начинается с большой буквы. Объявление объекта

```
obj NUMBER is sorts Nat Rat .
```

вводит объект `NUMBER` и два сорта с именами `Nat` и `Rat`. Пробелы (перед последней точкой тоже) обязательны.

2. *Оъявление операторов (функций).* Операторы могут определяться тремя следующими способами:

а) *функция* (стандартная префиксная форма записи функции):

```
op first : Flavor Flavor -> Flavor .
```

б) *mixfix операция* (инфиксная форма записи функции):

```
op _second_ : Flavor Flavor -> Flavor .
```

Символ подчеркивания есть место для аргумента, как, например, в выражении `X second Y`. Определение

```
op _ : Bit Bits -> Bits .
```

используется для создания бит стринга. Результат применения этой операции к паре стрингов `1 011` есть стринг `1011`;

в) *константы.* Константа есть операция местности (arity) 0. Например:

```
ops Chocolate Vanilla Strawberry : -> Flavor .
```

Ключевое слово `ops` используется для введения более чем одной операции. Для разделения операций в сложных случаях удобно использовать скобки. Например:

```
ops (_+_ ) (_-_) : IntExp IntExp -> IntExp .
```

3. *Оъявление переменных.* Переменная объявляется ключевым словом `var`. Более одной переменной одного и того же сорта можно ввести ключевым словом `vars`:

```
vars X Y : Flavor .
```

Имена переменных начинаются с большой буквы.

4. *Оъявление аксиом.* Аксиома есть равенство двух термов. Например:

```
eq first (X, Y) = X .
```

```
eq X second Y = Y .
```

5. *Конец спецификации.* Ключевое слово `endo` маркирует конец спецификации объекта. После слова `endo` точки нет.

15.3. Сорта и подсорта

OBJ3 базируется на строгом сортировании. Каждый символ имеет сорт или ассоциируется с сортом. Сорта вводятся в OBJ3 синтаксисом `sorts <SortIds>`, например как в

```
sorts Nat Int .
```

В OBJ3 действует система подсортов, которая поддерживает обработку частичных операторов, кратную наследственность и обработку ошибок. OBJ3 можно использовать для формальной спецификации иерархии объектно ориентированных программных компонент.

Сорт (множество) s есть подсорт (подмножество) сорта s , запись $s' < s$ верна, если домен для s включает домен для s' и операторы для s доступны для s' . Подсорт в OBJ3 объявляется командой

```
subsort  $s' < s$  .
```

Подмножество частичного упорядочения может быть установлено между локально определенными и импортированными сортами. Например:

```
subsort MyInt < MyRat < MyReal .
```

где `MyInt` есть подмножество в `MyRat`, а `MyRat` есть подмножество в `MyReal`. Всякая операция, определенная для сорта `MyReal`, доступна для переменных из сортов `MyInt` и `MyRat`. Следующие примеры иллюстрируют отношение быть подсортом некоторого сорта.

Подсортность гарантирует корректное применение функций к переменным соответствующих подсортов и индуцирует процесс работы с исключительными операциями должным образом.

Упорядоченная сортовая алгебра поддерживает также кратную наследственность, так что подсорт может иметь более чем один надсорт.

15.4. Импорт модулей

OBJ3 имеет четыре способа импорта модулей: `protecting` (защита), `extending` (расширение), `using` (использование), `including` (включение). Вместо ключевых слов `protecting`, `extending`, `using`, `including` можно использовать их аббревиатуры `pr`, `ex`, `us`, `inc`. Если модуль X импортирует модуль Y , который, в свою очередь, импортирует модуль Z , то модуль X тоже импортирует модуль Z , то есть отношение *импортирует* транзитивно.

Способ импорта `protecting` осуществляется следующим образом:

```
obj NATTREE is sorts Nebtree Btree .
  protecting NAT .
  protecting BOOL .
  subsorts Nat < Nebtree < Btree .
  ...
endo
```

Если модуль X импортирует модуль Y , то модуль Y защищен. Никакие новые элементы сортов из модуля Y не могут быть определены в этом модуле. Сигнатура модуля Y не может быть изменена, не могут быть введены никакие новые операции с сортами модуля Y . Уже определенная в модуле Y функция не может быть переопределена. Однако сигнатура модуля Y может быть использована в определении операций в X .

Способ импорта `extending` осуществляется, например, в модуле `ORDLIST` следующим образом:

```
obj ORDLIST is sort List .
  extending LIST .
  op insert : List Nat -> List .
  vars I J : Nat. var L : List.
  eq insert(null, I) = (I null) .
  cq insert(I L, J) = if I > J then (J I L)
                        else (I insert(L, J))
endо
```



Если модуль X импортирует модуль Y и модуль Y расширяется, то новые пункты данных сорта из Y могут быть определены в модуле X . Однако операции в модуле X не переопределяют никакую функцию, уже определенную в модуле Y . Из этого следует, что новые операции могут быть добавлены, чтобы расширить поведение модуля Y в модуле X .

Способ импорта `using` осуществляется следующим образом:

```
obj X is using Y .
  ...
endо
```



Если модуль X импортирует модуль Y способом `using`, то нет гарантии в том смысле, что новые (data) элементы данных сортов из модуля Y могут быть созданы, как и старые (data) элементы данных сортов из модуля Y могут быть переопределены. Этот способ импорта аналогичен повторному использованию (to code reuse) в объектно ориентированной парадигме. OBJ3 имплементирует использование копированием импортируемых модульных верхних уровней структур, делящих (sharing) все из модулей (all of the modules), которые это импортируют. Поэтому от сортов, определяемых данным модулем, требуется иметь различные имена, и все копируемые операции должны быть уникально (uniquely) идентифицированы их именем и рангом.

Способ импорта `including` осуществляется следующим образом:

```
obj X is including Y .
  ...
endо
```

Если модуль X включает модуль Y , то модуль Y инкорпорируется в модуль X без копирования. В этом единственная разница между способами `using` и `including`.

15.5. Встроенные сорта

Система OBJ3 включает библиотеку, содержащую несколько встроенных сортов следующих часто используемых абстрактных типов данных.

Встроенная теория TRUTH-VALUE выдает константу `truth` как значения `true` и `false`.

Встроенная теория TRUTH есть `TRUTH-VALUE` с операторами `==`, `!=`, `if _ then _ else _ fi`. Инфиксный полиморфный оператор `_==_` применим к любому сорту. Оператор `_==_` проверяет, равны ли синтаксически два основных (без переменных) терма. Если оператор `_==_` применяется к двум термам несравнимых сортов, то парсинг выдает ошибку. Для термов сорта `Bool` оператор `_==_` превращается в `_iff_`. Операторы `!=`, `if _ then _ else _ fi` обеспечиваются модулем `BOOL`. Они тоже полиморфны.

Встроенная теория BOOL выдает булевы инфиксные операторы `and`, `or`, `xor`, префиксный `not`, инфиксный `implies`. Поддерживаются операции из `TRUTH`. Квотированные идентификаторы не поддерживаются. Операции `==`, `!=`, `if _ then _ else _ fi` поддерживаются. Они полиморфны, то есть применимы к любым объявленным сортам.

Встроенная теория IDENTICAL может использоваться вместо `BOOL`. Операции `===`, `===` проверяют посимвольное (графическое) равенство термов без их вычисления.

Встроенная теория NAT выдает натуральные числа и операции над ними: сложение, умножение, меньше, больше, предшествование, следование.

Встроенная теория obj NZNAT выдает ненулевые натуральные числа.

Встроенная теория INT выдает целые числа и операции над ними, в том числе частное и остаток при делении целых чисел. Импортируется `NAT`.

Встроенная теория RAT выдает рациональные числа.

Встроенная теория FLOAT выдает числа с плавающей точкой, в том числе арифметические операции, линейный порядок, экспонента, квадрат, логарифм, тригонометрические функции.

Встроенная теория ID выдает идентификаторы (identifiers, слова), включает лексикографическое упорядочение и все операции, доступные в `BOOL`.

Встроенная теория QID работает так же, как `ID`. Различие с `ID` в том, что в `QID` идентификаторы котируются (начинаются с символа апостроф). Например, `'a`, `'b`, `'1300`, `'anyidentifier`. Импортируется `BOOL`.

Встроенная теория QIDL выдает идентификаторы с апострофами и включает все операции, доступные в `BOOL`. `QIDL` дополнительно включает лексикографическое упорядочение.

15.6. Декларация атрибутов

Некоторые свойства операций удобно рассматривать как атрибуты и объявлять их совместно с объявлением операций. Это ассоциативность, коммута-

тивность, равенство. Объявление атрибутов операций влияет на порядок вычислений и синтаксический разбор (parsing).

Ассоциативность и коммутативность. Следующие примеры иллюстрируют объявление ассоциативных операций.

```
op _or_ : Bool Bool -> Bool [assoc] .
op __ : NeList List -> NeList [assoc] .
```

Выражение, включающее ассоциативный оператор, не требует скобок.

Коммутативные инфиксные операции могут быть объявлены атрибутом `comm`, который семантически есть аксиома коммутативности. Бинарная операция может иметь коммутативные и ассоциативные атрибуты одновременно. Атрибут тождества может быть объявлен для бинарной операции. Например, команда

```
op _or_ : Bool Bool -> Bool [assoc id : false] .
```

содержит атрибут `assoc` (ассоциативность) и атрибут `id : false`, который логически эквивалентен команде

```
(B or false = B) and (false or B = B) .
```

Атрибуты тождества могут быть константы, такие как 0 для + и 1 for *, равно как и `nil` при объединении множества и пустого множества.

15.7. Приоритет

Приоритет (precedence) оператора есть натуральное число в пределах от нуля до 127. При этом меньшее число имеет больший приоритет и связывает сильнее большего числа в качестве приоритета. Например, встроенный объект `INT` имеет команды:

```
op _+ : Int Int -> Int [assoc comm id: 0 prec 33] .
op *_ : Int Int -> Int [assoc comm id: 1 prec 31] .
```

Приоритет по умолчанию для оператора стандартного вида (то есть префикс со скобками) есть 0. Если оператор не начинается и не заканчивается символом подчеркивания, то по умолчанию его приоритет есть тоже 0. Унарный префиксный оператор имеет по умолчанию приоритет 15. Во всех других случаях приоритет по умолчанию есть 41.

15.8. Параметризованное программирование

Параметризованное программирование осуществляется в OBJ3 с помощью следующих программных единиц: `object` (объект), `theory` (теория), `view` (вью, проекция), модульное выражение. Цель параметризации есть максимизация возможностей повторного использования программных единиц, которые, оставаясь замкнутыми в себе, недоступными для редактирования, готовы для повторного использования в других единицах программы.

Для абстрактных типов данных, например таких, как множество, могут быть написаны спецификации для операций и аксиом, которые затем могут быть использованы в качестве составной части таких спецификаций, как, например, конкретное множество целых чисел, множество вещественных чисел, множество последовательностей. Независимые от типа элементов множественные операции (объединение, пересечение, дополнение) могут быть специфицированы отдельно и вызываться потом в другие программные единицы, описывающие конкретные множества. Удобно отдельно специфицировать множество элементов сорта E , а затем использовать параметризованный модуль спецификации $SET[T]$, где T есть формальный параметр, который может быть отображен на сорт E . OBJ3 обеспечивает механизм для такой параметрической спецификации. Параметризованное программирование делит код на параметризованные компоненты. На уровне спецификации программные единицы `object`, `theory`, `view`, модульное выражение обеспечивают поддержку для написания параметрических спецификаций. Программная единица `Theory` используется для определения интерфейса и свойств параметризованного модуля. Программная единица `view` привязывает фактические параметры некоторого модуля к некоторой теории. Инстанциация параметризованного модуля с использованием некоторого конкретного `view` в качестве фактического параметра дает новый модуль. Модульные выражения, содержащие интерактивные (взаимодействующие) модули, порождают новый модуль, который состоит из интерактивных единиц в соответствии с модульным выражением.

15.9. Теории

Теории в OBJ3 имеют ту же структуру, что и объекты. Они описывают сорта, операции, переменные, равенства. Теории могут импортировать другие теории и объекты, они также могут быть параметризованы. Разница между объектами и теориями есть то, что объекты исполнимы, в то время как теории лишь определяют свойства. Ниже следуют примеры теорий и объектов.

Встроенная теория TRIV вводит новый сорт `Elt`.

```
th TRIV is sort Elt .
endth
```

15.9.1. Программная спецификация FIELD алгебраического поля

```
th FIELD is
  sorts Field NzField .
  subsorts NzField < Field .
  protecting BOOL .
  op 0 : -> Field .
  op 1 : -> NzField .
  op +_ : Field Field -> Field [assoc comm id: 0].
  op *_ : Field Field -> Field [assoc comm id: 1].
  op *_ : NzField NzField -> NzField [assoc comm id: 1].
```




```

op _- : Field -> Field .
op _^-1 : NzField -> NzField .
op nz : Field -> Bool .
vars X Y Z : Field .
vars X' Y' : NzField .
*** as NzField : X if nz(X) .
eq X + (- X) = 0 .
eq X' * (X' ^-1) = 1 .
eq X * (Y + Z) = (X * Y) + (X * Z) .
cq X = 0 if not nz(X) .
endth

```

Определение. Поле F есть алгебра $(F, \{+, \cdot\})$, где F есть множество, на котором определены две операции (функции): сложение $f(x, y) = x + y : F \times F \rightarrow F$, умножение $g(x, y) = x \cdot y : F \times F \rightarrow F$, удовлетворяющие следующим аксиомам. $\forall x \in F \forall y \in F \forall z \in F$:

1. $(x + y) + z = x + (y + z)$.
2. $x + y = y + x$.
3. $\exists 0 \in F \ x + 0 = x$.
4. $\forall x \in F \ \exists (-x) \in F \ x + (-x) = 0$.
5. $(x \cdot y) \cdot z = x \cdot (y \cdot z)$.
6. $x \cdot y = y \cdot x$.
7. $\exists e \in F \ \forall x \in F \ x \cdot e = x$.
8. $\forall x \in F - \{0\} \ \exists x^{-1} \in F \ x \cdot x^{-1} = e$.
9. $(x + y) \cdot z = x \cdot z + y \cdot z$.

Сравнивая теорию `th FIELD` с математическим определением поля, видим, что OBJ3-программа есть запись аксиом поля в терминах языка OBJ3.

15.9.2. Программная спецификация PROPC пропозиционального исчисления

Программа преобразует булеву формулу в полином Жегалкина. Слагаемые полинома располагаются в лексикографическом порядке.

```

obj PROPC is
  sort Prop .
  extending TRUTH .
  protecting QID .
  subsorts Id Bool < Prop .
  op _and_ : Prop Prop -> Prop [assoc comm
                                idem idr: true prec 2] .
  op _xor_ : Prop Prop -> Prop [assoc comm
                                idr: false prec 3] .

  vars p q r : Prop .
  eq p and false = false .
  eq p xor p = false .
  eq p and (q xor r) = (p and q) xor (p and r) .
  op _or_ : Prop Prop -> Prop [assoc prec 7] .

```

```

op not_ : Prop -> Prop [prec 1] .
op _implies_ : Prop Prop -> Prop [prec 9] .
op _iff_ : Prop Prop -> Prop [assoc prec 11] .
eq p or q = (p and q) xor p xor q .
eq not p = p xor true .
eq p implies q = (p and q) xor p xor true .
eq p iff q = p xor q xor true .
endo

```

Следующие примеры дают представление о программном представлении в OBJ3 множеств, стеков, списков.



15.9.3. Программная спецификация SET-NAT множества натуральных чисел

Модуль SET-NAT защищает импортированные модули NAT и BOOL.

```

obj SET-NAT is sort Set .
  protecting BOOL .
  protecting NAT .
  op empty : -> Set .
  op insert : Nat Set -> Set .
  op member : Nat Set -> Bool .
  vars S : Set .
  vars N M : Nat .
  eq member(N, empty) = false .
  eq member(N, insert(M, S)) = (M == N) or member(N, S) .
  eq insert(N, insert(M, S)) = insert(M, insert(N, S)) .
endo

```

*** Ниже следуют запросы к программе и ответы программы.

```

red insert(10, insert(11,empty)) .
red member(11, insert(10, insert(11,empty))) .
red member(15, insert(10, insert(11,empty))) .
let S = insert(10, insert(11,empty)) . red member(11,S)
let S = insert(10, insert(11,empty)) . red member(15,S) .
=====
reduce in SET-NAT : insert(10, insert(11, empty))
result Set: insert(11, insert(10, empty))
rewrite time: 141ms      parse time: 0ms
=====
reduce in SET-NAT : member(11, insert(10, insert(11, empty)))
result Bool: true
rewrite time: 16ms      parse time: 63ms
=====
reduce in SET-NAT : member(15, insert(10, insert(11, empty)))
result Bool: false
rewrite time: 15ms      parse time: 0ms
opening module SET-NAT.

```



```

=====
reduce in SET-NAT : member(11, S)
result Bool: true
rewrite time: 31ms      parse time: 0ms
=====
reduce in SET-NAT : member(15, S)
result Bool: false
rewrite time: 16ms      parse time: 0ms
=====

```

Аналогичную программу можно написать для множества слов.

15.9.4. Программная спецификация obj WORDTREE дерева слов

```

obj WORDTREE is sort Tree .
  protecting BOOL .
  protecting QID .
  op empty : -> Tree .
  op node : Tree Id Tree -> Tree .
  op left : Tree -> Tree .
  op right : Tree -> Tree .
  op content : Tree -> Id .
  op isempty : Tree -> Bool .
  op isfound : Tree Id -> Bool .
  vars X Y : Tree .
  vars N M : Id .
  eq isempty(empty) = true .
  eq isempty(node(X, N, Y)) = false .
  eq left(node(X, N, Y)) = X .
  eq right(node(X, N, Y)) = Y .
  eq content(node(X, N, Y)) = N .
  eq isfound(empty, M) = false .
  eq isfound(node(X, N, Y), M) =
    (M == N) or (isfound(X, M) or isfound(Y, M)) .
endo

let n5 = node(empty, '3, empty) .
let n6 = node(empty, '7, empty) .
let n3 = node(n5, '10, n6) .
let n4 = node(empty, '15, empty) .
let n1 = node(empty, '1, empty) .
let n2 = node(n3, '1, n4) . let n0 = node(n1, '4, n2) .
red n0 . red content(n0) . red left(n0) . red right(n0) .
red isempty(n2) . red isfound(n0, '3) . red isfound(n3, '10) .
=====

reduce in WORDTREE : n0
result Tree: node(node(empty, '1, empty), '4, node(node(empty,
  '3, empty), '10, node(empty, '7, empty)), '1, node(empty, '15,
  empty)))
rewrite time: 187ms      parse time: 0ms
=====

```



```

reduce in WORDTREE : content(n0)
result Id: '4
rewrite time: 0ms      parse time: 16ms
=====
reduce in WORDTREE : left(n0)
result Tree: node(empty, '1, empty)
rewrite time: 15ms     parse time: 16ms
=====
reduce in WORDTREE : right(n0)
result Tree: node(node(node(empty, '3, empty), '10, node(empty, '7,
    empty)), '1, node(empty, '15, empty))
rewrite time: 16ms     parse time: 0ms
=====
reduce in WORDTREE : isempty(n2)
result Bool: false
rewrite time: 31ms     parse time: 0ms
=====
reduce in WORDTREE : isfound(n0, '3)
result Bool: true
rewrite time: 47ms     parse time: 0ms
=====
reduce in WORDTREE : isfound(n3, '10)
result Bool: true
rewrite time: 0ms      parse time: 0ms
=====

```

Аналогичную программу можно написать для бинарного дерева натуральных чисел.

15.9.5. Программная спецификация WORDSTACK словарного стека

```

obj WORDSTACK is sort Stack .
  protecting QID .
  op empty : -> Id .
  op newstack : -> Stack .
  op push : Stack Id -> Stack .
  op pop : Stack -> Stack .
  op top : Stack -> Id .
  vars S : Stack .
  vars N : Id .
  eq pop(newstack) = newstack .
  eq top(newstack) = empty .
  eq pop(push(S, N)) = S .
  eq top(push(S, N)) = N .
endo
red push(push(push(newstack, '5), '7), '10) .
red pop(push(push(push(newstack, '5), '7), '10)) .
red top(newstack) .
let st1 = push(push(push(newstack, '5), '7), '10) .
red st1 .

```



Аналогичную программу можно написать для стека натуральных чисел.

15.9.6. View

View (отображение, проекция) описывает связь между теорией и объектом таким образом, что сорта теории отображаются на сорта объекта с сохранением отношения подсортности. Операции теории отображаются на операции объекта. *View* есть гомоморфизм из алгебры, описанной теорией, в алгебру, описанную объектом. Ниже следуют примеры *view*.

```
view TRIV-TO-FLAVORS from TRIV to FLAVORS is
  sort Elt to Flavor .
  op newop to first .
endv
```

Теории параметризованных модулей должны быть определены раньше в последовательности модулей, которые используют эти теории. Параметризованные модули (объекты или теории) представляются в системе OBJ3 следующим образом:

`obj NAME[X :: THEORY1]` для объекта,
`th NAME[X :: THEORY1]` для теории.

Таким объявлением все сорта, операции, равенства из *THEORY1* становятся видимы в модуле *NAME*.

15.9.7. Инстанциация

Инстанциация параметризованного модуля замещает его формальные параметры фактическими параметрами. Каждая теория замещается соответствующим фактическим модулем с использованием *view* для связывания фактических параметров с формальными параметрами.

Когда алгебра инстанциации используется в нескольких контекстах, удобно именовать алгебру, используя команду *make*. Например,

```
make BAR-FLAVORS is BAR[TRIV-TO-FLAVORS] endm .
```

где *BAR-FLAVORS* есть имя, данное замещенному объекту *BAR[TRIV-TO-FLAVORS]*.

Приведем далее примеры параметризации.

15.9.8. Параметризованная теория линейного векторного пространства

```
th VECTOR-SP[F :: FIELD] is
  sort Vector .
  op 0 : -> Vector .
  op _+_ : Vector Vector -> Vector [assoc comm id: 0] .
  op *__ : Field Vector -> Vector .
  vars F F1 F2 : Field .
  vars V V1 V2 : Vector .
  eq (F1 + F2) * V = (F1 * V) + (F2 * V) .
  eq (F1 * F2) * V = F1 * (F2 * V) .
  eq F * (V1 + V2) = (F * V1) + (F * V2) .
endth
```

Определение. *Линейное векторное пространство* L над полем F есть множество L элементов произвольной природы (векторов) $x, y, z, \dots$, на котором определены две (линейные) операции: сложение векторов $f(x, y) = x + y: L \times L \rightarrow L$, умножение $g(a, x) = a \cdot x: F \times L \rightarrow L$ элемент a из F на вектор x (при этом $ax = xa$), удовлетворяют следующим аксиомам. $\forall x, y, z \in L, \forall a, b, c \in F$:

1. $x + (y + z) = (x + y) + z$.
2. $x + y = y + x$.
3. $\exists 0 \in L \ x + 0 = x$.
4. $\exists (-x) \in L \ x + (-x) = 0$.
5. $a \cdot (x + y) = a \cdot x + a \cdot y$.
6. $(a \oplus b) \cdot x = a \cdot x + b \cdot x$.
7. $(a \odot b) \cdot x = a \cdot (b \cdot x)$.
8. $1 \cdot x = x, 1 \in F$.

Вектор 0 из L есть нулевой вектор в L . Вектор $-x$ из L называется вектором, противоположным к x . В линейном пространстве существуют объекты двух типов (сортов): векторы и элементы поля.

Сравнивая теорию `th VECTOR-SP[F :: FIELD]` с математическим определением линейного векторного пространства, видим, что OBJ3-программа `th VECTOR-SP[F :: FIELD]` есть запись аксиом линейного векторного пространства в терминах языка OBJ3. При этом теория поля `th FIELD` и ее аксиомы видны из теории линейного векторного пространства `th VECTOR-SP[F :: FIELD]`.

15.9.9. Параметризованный модуль `obj ORD-PAIR` пар вида (натуральное число, слово)

Модули могут иметь более одного параметра. Двупараметризованный модуль имеет следующую сигнатуру:

```
obj NAME[X :: THEORY1, Y :: THEORY2]
```

Если две теории одинаковы, то можно писать:

```
obj NAME[X Y :: THEORY1]
```

Параметризованный модуль `obj ORD-PAIR` имеет два параметра S и T , являющихся двумя различными именами для теории `TRIV`. Параметр $S :: TRIV$ с помощью проекции `view TRIV-TO-NAT` связывает первую переменную `Elt` пары с сортом натуральных чисел. Параметр $T :: TRIV$ с помощью проекции `view TRIV-TO-QID` связывает вторую переменную `Elt` пары с сортом котируемых слов. Команда `make NATWORD-ORD-PAIR` создает замещенный модуль `ORD-PAIR[TRIV-TO-NAT, TRIV-TO-QID]`, обрабатывающий пару с первым элементом – натуральным числом и со вторым элементом пары – котируемым словом.

```
obj ORD-PAIR[S :: TRIV, T :: TRIV] is sort OrdPair .
  protecting NAT .
  protecting BOOL .
  protecting IDENTICAL .
```

```

protecting QID .
op pair : Elt.S Elt.T -> OrdPair .
op first : OrdPair -> Elt.S .
op second : OrdPair -> Elt.T .
op eqp : OrdPair OrdPair -> Bool [comm] .
var Es : Elt.S . var Et : Elt.T . vars P Q : OrdPair .
eq first(pair(Es, Et)) = Es .
eq second(pair(Es, Et)) = Et .
eq eqp(P, Q) = (first(P) === first(Q)) and
(second(P) === second(Q)) .
endo
view TRIV-TO-NAT from TRIV to NAT is
  sort Elt to Nat .
endv
view TRIV-TO-QID from TRIV to QID is
  sort Elt to Id .
endv
make NATWORD-ORD-PAIR is
  ORD-PAIR[TRIV-TO-NAT, TRIV-TO-QID]
endm
red first(pair(1, '2)) . red second(pair(1, '2)) .
red eqp(pair(1, '2), pair(1, '2)) .
red eqp(pair(1, '2), pair(1, '3)) .

```

15.9.10. Параметризованный модуль SEQUENCE[X :: ELEMS] списков натуральных чисел и списков слов

```

th ELEMS is
  sort Elems .
endth
obj SEQ[X :: TRIV] is
  protecting BOOL .
endo
obj SEQUENCE[X :: ELEMS] is sort Seq .
  protecting SEQ[X] . protecting NAT .
  protecting BOOL . protecting QID .
  subsort Elems < Seq .
  op empty : -> Seq .
  op __ : Seq Seq -> Seq [assoc id: empty] .
  op tail : Seq -> Seq .
  op head : Seq -> Elems .
  op length : Seq -> Nat .
  op equ : Seq Seq -> Bool [comm] .
  vars U V : Seq . vars X Y : Elems .
  eq length(empty) = 0 .
  eq length(X U) = 1 + (length(U)) .
  eq head(X U) = X .
  eq tail(empty) = empty .

```



```
eq tail(X U) = U .
eq equ(empty, empty) = true .
cq equ(U, empty) = false if U /= empty .
cq equ(U, V) = equ(tail(U), tail(V))
  if U /= empty and V /= empty and head(U) == head(V) .
cq equ(U, V) = false
  if U /= empty and V /= empty and head(U) /= head(V) .
endv
view ELEM3-T0-NAT from ELEM3 to NAT is
  sort ELEM3 to Nat .
endv
make NAT-SEQUENCE is
  SEQUENCE[ELEM3-T0-NAT]
endm
red length(1 2 3) . red head(1 2 3) . red tail(1 2 3) .
red equ((1 2 3), (1 2 3)) . red equ((1 2 3), (2 3 4)) .
view ELEM3-T0-QID from ELEM3 to QID is
  sort ELEM3 to Id .
endv
make WORD-SEQUENCE is
  SEQUENCE[ELEM3-T0-QID]
endm
red length('1 '2 '3) . red head('1 '2 '3) .
red tail('1 '2 '3) .
red equ(('1 '2 '3), ('1 '2 '3)) .
red equ(('1 '2 '3), ('2 '3 '4)) .
```



Приложение 1

Логика высказываний и предикатов. Пролог

Задача 1. Заданную функцию $f(x_1, x_2, x_3, x_4)$ представить: 1) таблицей своих значений; 2) множеством M_1 десятичных эквивалентов двоичных наборов, на которых f принимает значение 1; 3) множеством M_0 десятичных эквивалентов двоичных наборов, на которых f принимает значение 0; 4) картой Карно; 5) двоичным единичным кубом.

1.1. 0111001011110001.	1.2. 0001110000111011.
1.3. 1100111001110010.	1.4. 0101000111000101.
1.5. 1100010110100110.	1.6. 1001110100011010.
1.7. 0100110000011110.	1.8. 1111000100111011.
1.9. 0000110101110110.	1.10. 1011101011000101.
1.11. 0011101100011110.	1.12. 0111011001011010.
1.13. 0001111010111010.	1.14. 0101101010011101.
1.15. 1011101011011100.	1.16. 1011000101111100.
1.17. 1001110101111100.	1.18. 0011011101111100.
1.19. 1101110001110111.	1.20. 0111110010001101.
1.21. 0111011111100010.	1.22. 1000110101000101.
1.23. 1110001010111001.	1.24. 0100010101000111.
1.25. 1011100110000110.	1.26. 0100011101110011.
1.27. 1000011001110011.	1.28. 0101011001110011.
1.29. 0111010001010110.	1.30. 0101011001010110.

Задача 2. Для данных формул построить таблицу истинностных значений и определить, является ли формула: а) общезначимой; б) выполнимой; в) опровержимой; г) невыполнимой.

- 2.1. $(x \rightarrow (y \rightarrow z)) \rightarrow ((x \rightarrow y) \rightarrow (x \rightarrow z))$,
 $\neg(((x \rightarrow y) \rightarrow (\neg z \rightarrow u)) \rightarrow w) \rightarrow w \rightarrow (x \rightarrow (u \rightarrow x))$,
 $(x \vee \neg x) \equiv \neg x$, $(x \& \neg x) \equiv x$.
- 2.2. $(x \rightarrow y) \rightarrow ((x \rightarrow z) \rightarrow (x \rightarrow yz))$,
 $\neg((x \rightarrow y) \rightarrow ((x \rightarrow (y \rightarrow z)) \rightarrow (x \rightarrow z)))$,
 $(x \rightarrow y) \rightarrow ((\neg(x \rightarrow z) \rightarrow (x \rightarrow yz)))$.

- 2.3. $(x \rightarrow z) \rightarrow ((y \rightarrow z) \rightarrow (x \equiv (y \rightarrow z))), \neg((xy \rightarrow z) \rightarrow (x \rightarrow (y \rightarrow z))),$
 $(x \rightarrow (y \rightarrow z)) \rightarrow (\neg(x \rightarrow y) \rightarrow (x \rightarrow z)).$
- 2.4. $(x \equiv x) \vee x, (x \& x) \equiv x, (x \vee x) \equiv x, \neg((x \rightarrow (y \rightarrow z)) \rightarrow (x \& y \rightarrow z)),$
 $(x \vee \neg y) \equiv \neg(x \vee y), x \& y \equiv y \& x.$
- 2.5. $x \vee y \equiv y \vee x, x \& y \equiv y \& x, ((\neg x \rightarrow \neg y) \rightarrow ((\neg y \rightarrow x) \rightarrow y)),$
 $(xy \rightarrow z) \rightarrow (x \rightarrow (\neg(y \rightarrow z))).$
- 2.6. $((x \rightarrow y) \rightarrow (\neg z \rightarrow u)) \rightarrow w \rightarrow ((w \rightarrow x) \rightarrow (u \rightarrow x)),$
 $\neg((x \vee y) \equiv (y \vee x)), \neg((x \& y) \equiv (y \& x)), (\neg x \rightarrow \neg y) \rightarrow ((\neg y \rightarrow x) \rightarrow \neg y).$
- 2.7. $(x \rightarrow y) \rightarrow ((x \rightarrow (y \rightarrow z)) \rightarrow (x \rightarrow z)),$
 $\neg((x \rightarrow z) \rightarrow ((y \rightarrow z) \rightarrow (x \equiv (y \rightarrow z)))), (x \rightarrow (y \rightarrow z)) \rightarrow (\neg(xy \rightarrow z)).$
- 2.8. $(x \rightarrow y) \rightarrow ((y \rightarrow z) \rightarrow (x \rightarrow z)), \neg((x \rightarrow y) \rightarrow ((x \rightarrow z) \rightarrow (x \rightarrow yz))),$
 $((x \rightarrow y) \rightarrow (\neg z \rightarrow \neg u)) \rightarrow \neg w \rightarrow ((w \rightarrow x) \rightarrow (z \rightarrow x)).$
- 2.9. $(\neg y \rightarrow x) \rightarrow ((y \rightarrow x) \rightarrow x),$
 $\neg((x \rightarrow (y \rightarrow z)) \rightarrow ((x \rightarrow y) \rightarrow (x \rightarrow z))), x \vee yz \equiv (x \vee y) \& (\neg x \& \neg y).$
- 2.10. $(xy \rightarrow z) \rightarrow (x \rightarrow (y \rightarrow z)), \neg((x \vee x) \equiv x), \neg(((x \& x) \equiv x),$
 $x \& \neg(yz) \equiv (xy)z.$
- 2.11. $(x \rightarrow (y \rightarrow z)) \rightarrow (xy \rightarrow z), \neg((\neg y \rightarrow x) \rightarrow ((y \rightarrow x) \rightarrow x)),$
 $(x \rightarrow y) \rightarrow ((\neg y \rightarrow z) \rightarrow (\neg x \rightarrow z)).$
- 2.12. $(\neg x \rightarrow \neg y) \rightarrow ((\neg y \rightarrow x) \rightarrow y), \neg((x \rightarrow y) \rightarrow ((y \rightarrow z) \rightarrow (x \rightarrow z))),$
 $(x \vee z) \rightarrow (\neg(y \rightarrow z) \rightarrow ((x \vee y) \rightarrow z)).$
- 2.13. $(x \vee yz) \equiv (x \vee y)(x \vee z), \neg((x \rightarrow y) \equiv (\neg y \rightarrow \neg x)),$
 $(x \rightarrow y) \rightarrow ((x \rightarrow (\neg y \rightarrow z)) \rightarrow (x \rightarrow z)).$
- 2.14. $(x \vee (y \vee z)) \equiv ((x \vee y) \vee z), \neg((x \rightarrow y) \equiv (\neg x \vee y)), x(\neg y \vee z) \equiv (xy \vee xz).$
- 2.15. $x \rightarrow (y \rightarrow x), \neg(\neg x \rightarrow (\neg y \rightarrow \neg(x \vee y))), \neg(x \vee y) \equiv (x \& \neg y).$
- 2.16. $xy \rightarrow x, \neg(x(y \vee z) \equiv (xy \vee xz)), (\neg y \rightarrow \neg x) \rightarrow ((y \rightarrow x) \rightarrow \neg x).$
- 2.17. $xy \rightarrow y, \neg((x(yz) \equiv (xy)z)), (x \rightarrow y) \equiv (y \rightarrow \neg x).$
- 2.18. $x \rightarrow (x \vee y), \neg(x(x \vee y) \equiv x), \neg x \equiv \neg\neg x.$
- 2.19. $y \rightarrow (x \vee y), \neg(y \rightarrow (x \vee y)), (x \vee (y \vee \neg z)) \equiv ((x \vee y) \vee z).$
- 2.20. $(x \rightarrow y) \equiv (\neg y \rightarrow \neg x), \neg(x \rightarrow (y \rightarrow x)), x \rightarrow (\neg y \rightarrow (x \vee y)).$
- 2.21. $x \equiv \neg x, \neg(\neg(x \vee y) \equiv \neg x \& \neg y), ((x \vee \neg y) \equiv \neg(x \& y)).$
- 2.22. $(x \& \neg y) \equiv (\neg x \vee y), \neg((x \vee xy) \equiv x), xy \rightarrow \neg y.$
- 2.23. $x(yz) \equiv (xy)z, \neg(\neg(x \vee y) \equiv (\neg x \& \neg y)), x(x \vee \neg y) \equiv x.$
- 2.24. $(x \vee xy) \equiv x, \neg(x \vee (y \vee z) \equiv (x \vee y) \vee z), xy \rightarrow x.$
- 2.25. $x(x \vee y) \equiv x, \neg(x \equiv \neg\neg x), \neg x \vee xy \equiv x.$
- 2.26. $x(y \vee z) \equiv (xy \vee xz), \neg((x \& y) \rightarrow y), \neg x \rightarrow (y \rightarrow x).$
- 2.27. $x \vee \neg x \& y, \neg(x \rightarrow (x \vee y)), (x \rightarrow \neg y) \equiv (\neg x \vee y).$
- 2.28. $(x \vee y) \equiv (\neg x \& \neg y), \neg(x \vee \neg y), x \rightarrow (\neg x \vee y).$
- 2.29. $(x \& y) \equiv (\neg x \vee \neg y), \neg(x \& y \rightarrow x), \neg x \vee \neg x.$
- 2.30. $\neg x \rightarrow (\neg y \rightarrow x \vee y), \neg((x \vee yz) \equiv (x \vee y)(x \vee z)), y \rightarrow x \vee \neg y.$

Задача 3. Для данных формул построить таблицу истинностных значений и определить, является ли формула: а) общезначимой; б) выполнимой; в) опровержимой; г) невыполнимой; д) упростить формулу. Для обеих формул

построить схемы из функциональных элементов для дизъюнкции, конъюнкции, отрицания, импликации (рис. П1.1).

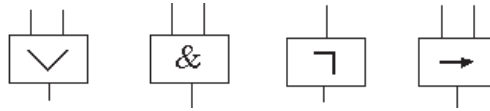


Рис. П1.1

- 3.1. $x \rightarrow \neg(\neg(x \& \neg y) \vee (x \rightarrow y))$.
- 3.2. $\neg x \vee (\neg y \rightarrow x) \vee \neg(x \rightarrow y)$.
- 3.3. $\neg(x \rightarrow \neg(xy)) \vee \neg(y \rightarrow \neg x \& \neg y)$.
- 3.4. $\neg(xy) \rightarrow (y \rightarrow \neg(xy))$.
- 3.5. $(x \rightarrow y) \rightarrow \neg(\neg y \rightarrow \neg(\neg(xy) \vee x))$.
- 3.6. $\neg(x \rightarrow y) \rightarrow (\neg y \vee \neg(xy))$.
- 3.7. $\neg(\neg x \rightarrow y) \rightarrow \neg(\neg x \vee \neg(xy))$.
- 3.8. $\neg(\neg(x \vee xy) \rightarrow x) \rightarrow y$.
- 3.9. $\neg((xy \vee x \& \neg y) \rightarrow x) \rightarrow y$.
- 3.10. $\neg(\neg(x \vee y) \vee \neg(x \rightarrow y)) \rightarrow x$.
- 3.11. $\neg(xy \vee \neg x \& y) \vee \neg(\neg(x \vee xy) \vee \neg x) \vee x$.
- 3.12. $\neg(x \rightarrow y) \vee \neg(\neg x \rightarrow y \vee \neg(x \rightarrow xy) \vee (y \rightarrow x))$.
- 3.13. $\neg(x \vee y) \rightarrow \neg(x \rightarrow y) \vee y$.
- 3.14. $(\neg(xy \vee y) \rightarrow \neg(xy \rightarrow x)) \rightarrow x$.
- 3.15. $\neg(\neg x \& y \vee x) \vee \neg(xy \rightarrow y) \rightarrow (x \vee y)$.
- 3.16. $(\neg(x \rightarrow y) \vee x) \rightarrow ((x \vee y) \rightarrow y)$.
- 3.17. $(\neg(xy) \rightarrow y) \rightarrow ((xy \rightarrow \neg x \& y) \rightarrow x)$.
- 3.18. $\neg(x \vee y) \vee \neg(y \rightarrow ((xy \rightarrow (x \vee y))))$.
- 3.19. $\neg(xy \rightarrow y) \vee ((\neg(x \rightarrow y) \vee xy) \rightarrow x)$.
- 3.20. $\neg(xy \rightarrow y \rightarrow \neg((x \rightarrow y) \rightarrow xy))$.
- 3.21. $\neg(x \vee y) \vee \neg(y \rightarrow (xy \rightarrow \neg x \& y) \rightarrow y)$.
- 3.22. $(\neg(xy \vee y) \rightarrow x) \rightarrow \neg x$.
- 3.23. $((x \& \neg y \vee \neg(\neg x \vee y)) \rightarrow x) \rightarrow y$.
- 3.24. $(\neg y \vee x) \rightarrow (x \& \neg y \vee y)$.
- 3.25. $\neg(x \rightarrow \neg x \& y) \rightarrow \neg(y \rightarrow (x \rightarrow y))$.
- 3.26. $\neg(x \& \neg y) \vee \neg(xy) \rightarrow x$.
- 3.27. $\neg((\neg x \& \neg y \rightarrow x) \rightarrow (yx \vee \neg x))$.
- 3.28. $\neg(\neg x \rightarrow y) \vee \neg(\neg x \& y \rightarrow x \& \neg y)$.
- 3.29. $(x \vee \neg(xy)) \rightarrow \neg(y \rightarrow xy)$.
- 3.30. $(\neg(xy) \vee x \& \neg y) \rightarrow \neg(x \vee y)$.

Задача 4. Построить СДНФ, СКНФ, полином Жегалкина для функции $f(x_1, x_2, x_3)$, заданной множеством M_1 десятичных эквивалентов двоичных наборов, на которых f принимает значение 1.

4.1. {4,5,6,7}.	4.2. {3,4,5,6}.	4.3. {2,3,5,6}.	4.4. {1,3,5,6}.
4.5. {0,1,2,3}.	4.6. {0,1,2,7}.	4.7. {0,1,4,7}.	4.8. {0,2,4,7}.

- 4.9. {4,5,7}. 4.10. {4,6,7}. 4.11. {2,3,7}. 4.12. {0,1,4,5,6}.
 4.13. {1,3,7}. 4.14. {0,1,2,3,6}. 4.15. {0,5,7}. 4.16. {2,6,7}.
 4.17. {0,5,6}. 4.18. {0,1,2,3,5}. 4.19. {0,3,6}. 4.20. {0,3,5}.
 4.21. {1,2,3,4,6}. 4.22. {1,2,3}. 4.23. {1,4,6}. 4.24. {0,2,4,5,6}.
 4.25. {0,6,7}. 4.26. {0,1,5,6,7}. 4.27. {2,4,5,6}. 4.28. {3,4,5,7}.
 4.29. {1,4,6,7}. 4.30. {4,5,7}.

Задача 5. Найти все тупиковые и все минимальные ДНФ и КНФ для всюду определенной функции. Одну из минимальных форм реализовать схемой с элементами для $\&$, $\vee$, $\neg$, $\rightarrow$ (см. рис. П1.1).

- 5.1. 1001001110011011. 5.2. 0010100011011111.
 5.3. 1101111100100010. 5.4. 1001100110111001.
 5.5. 1110110011001100. 5.6. 1101110110001010.
 5.7. 1010100011011101. 5.8. 1110110011001100.
 5.9. 1101001000111011. 5.10. 1010000011011111.
 5.11. 1010100001110111. 5.12. 1010101001011101.
 5.13. 0110111011000110. 5.14. 1110010011101100.
 5.15. 0111110100101010. 5.16. 0010100011111101.
 5.17. 1100011011101100. 5.18. 1111001000111011.
 5.19. 0011011111100111. 5.20. 1010001101110011.
 5.21. 1110011111100001. 5.22. 0010001001010111.
 5.23. 1101110110001010. 5.24. 0111001001111010.
 5.25. 1011011100001011. 5.26. 1010001111011011.
 5.27. 1101101011010010. 5.28. 1010100001111111.
 5.29. 0111110110001010. 5.30. 0101100011110010.

Задача 6. Для заданной всюду определенной функции $f(x_1, x_2, x_3, x_4)$ построить минимальную ДНФ методом Квайна–МакКласки. Каждая функция задана множеством M_1 десятичных эквивалентов двоичных наборов, на которых функция принимает значение 1.

- 6.1. {1,3,5,7,9,10,11,12,13}. 6.2. {2,3,6,7,9,10,11,12,14}.
 6.3. {4,5,6,7,9,10,12,13,14}. 6.4. {1,2,5,6,7,9,10,11,13}.
 6.5. {2,3,5,6,7,9,10,11,14}. 6.6. {1,3,5,6,7,9,11,12,13}.
 6.7. {2,3,5,6,7,10,11,12,14}. 6.8. {3,4,5,6,7,10,12,13,14}.
 6.9. {3,4,5,6,7,9,12,13,14}. 6.10. {1,3,5,6,7,9,10,11,13}.
 6.11. {5,6,8,9,10,11,12,13,14}. 6.12. {3,6,8,9,10,11,12,13,14}.
 6.13. {3,5,8,9,10,11,12,13,14}. 6.14. {1,3,5,7,9,10,11,12,13,14}.
 6.15. {1,2,5,6,7,9,10,11,13,14}. 6.16. {1,3,5,6,7,9,11,12,13,14}.
 6.17. {1,3,5,6,7,9,10,11,13,14}. 6.18. {0,1,3,5,7,9,10,11,12,13}.
 6.19. {0,2,3,6,7,9,10,11,12,14}. 6.20. {0,4,5,6,7,9,10,12,13,14}.
 6.21. {0,1,2,5,6,7,9,10,11,13}. 6.22. {0,2,3,5,6,7,9,10,11,14}.
 6.23. {0,1,3,5,6,7,9,11,12,13}. 6.24. {0,2,3,5,6,7,10,11,12,14}.
 6.25. {2,6,12,13,14,15}. 6.26. {0,1,4,5,8,9,11,12,13,14,15}.

6.27. {3,6,7,11,12,13,14,15}.

6.28. {2,3,7,10,11,12,13,14,15}.

6.29. {0,1,4,5,7,10,11,12,13,15}.

6.30. {0,2,3,4,6,7,8,14,15}.

Задача 7. Найти все тупиковые и все минимальные ДНФ и КНФ для частично определенной функции. Одну из минимальных форм реализовать схемой с элементами $\&$, $\vee$, $\neg$, $\rightarrow$ (см. рис. П1.1).

7.1. 1---010010--1--1.

7.2. 1---111100--0--0.

7.3. 1---011110--0--0.

7.4. 1---101110--0--0.

7.5. 1---110110--0--0.

7.6. 1---111100--0--0.

7.7. 0---111100--0--1.

7.8. 0---11100--0--10.

7.9. 0---011111--0--0.

7.10. 0-1-1-010-110---.

7.11. 1-1-0010-01-100.

7.12. -1-1010101--0---.

7.13. --1-01-0001-1-1-.

7.14. -1-1-10-00110---.

7.15. 0-11011--1--0--0.

7.16. 1-010-0-01-1--1.

7.17. -1--00001-1--1-1.

7.18. 1-1-10-01010---.

7.19. --1-1-01010-01--.

7.20. --1-1-010-010-1.

7.21. --1--1110-0-010-.

7.22. 11-11--1-0-0-0-0.

7.23. 0-01010--1--1--1.

7.24. -10-010-0--1--1-.

7.25. 1-1-0-01--0-1--1.

7.26. -1--1010-0-01-1.

7.27. 1-1-----0-010101.

7.28. 010-1-01--01--1-.

7.29. 1--0--101--010-1.

7.30. -01-10-10-0-1--1.

Задача 8. Минимизировать всюду определенную функцию алгебры логики из задачи 5 и частично определенную функцию из задачи 6 с помощью карт Карно.

Задача 9. Построить минимальную ДНФ системы функций $f_1(x_1, x_2, x_3)$, $f_2(x_1, x_2, x_3)$, $f_3(x_1, x_2, x_3)$ и реализовать ее с помощью ПЛИМ. Совместную минимизацию функций проводить с помощью карт Карно.

Каждая функция задана множеством M_i десятичных эквивалентов двоичных наборов, на которых функция принимает значение 1. (Пример см. в следующей задаче 10.)

9.1. {2,3,4,5,7}; {0,4,5}; {3,4,5,7}.

9.2. {1,3,4,6,7}; {0,4,6}; {3,4,5,7}.

9.3. {2,3,4,5,7}; {0,2,6}; {2,3,5,7}.

9.4. {1,3,4,6,7}; {0,1,3}; {1,3,6,7}.

9.5. {1,2,5,6,7}; {0,1,5}; {1,5,6,7}.

9.6. {1,2,5,6,7}; {0,2,5}; {2,5,6,7}.

9.7. {1,3,5,7}; {1,2,3}; {4,5,7}.

9.8. {2,3,6,7}; {1,2,3}; {4,6,7}.

9.9. {1,3,5,7}; {1,4,5}; {2,3,7}.

9.10. {2,3,6,7}; {2,4,6}; {1,3,7}.

9.11. {4,5,6,7}; {2,4,6}; {1,5,7}.

9.12. {4,5,6,7}; {1,4,5}; {1,6,7}.

9.13. {3,4,5,7}; {4,5,6}; {1,2,3}.

9.14. {3,4,6,7}; {4,5,6}; {1,2,3}.

9.15. {2,3,5,7}; {2,3,6}; {1,4,5}.

9.16. {1,3,6,7}; {1,3,5}; {2,4,6}.

9.17. {1,5,6,7}; {1,3,5}; {2,4,6}.

9.18. {2,5,6,7}; {2,3,6}; {1,4,5}.

9.19. {0,2,4}; {0,2,3}; {1,2,3}.

9.20. {0,1,4}; {0,1,3}; {1,2,3}.

9.21. {0,2,4}; {0,4,5}; {1,4,5}.

9.22. {0,1,4}; {0,4,6}; {2,4,6}.

- 9.23. $\{0,1,2\}; \{0,2,6\}; \{2,4,6\}$. 9.24. $\{0,1,2\}; \{0,1,5\}; \{1,4,5\}$.
 9.25. $\{0,2,6,7\}; \{0,1,6\}; \{0,2,7\}$. 9.26. $\{0,2,6,7\}; \{2,5,7\}; \{1,3,7\}$.
 9.27. $\{0,1,4,6,7\}; \{3,4,5,6,7\}; \{4,5,7\}$. 9.28. $\{0,1,2,3,5,7\}; \{4,5,7\}; \{0,1,3\}$.
 9.29. $\{0,1,2\}; \{0,1,4,7\}; \{6,7\}$. 9.30. $\{4,5,7\}; \{0,3,4,5,7\}; \{2,3,6\}$.

Задача 10. Провести приближенную совместную минимизацию четырех функций алгебры логики. В качестве заданий взять из задачи 5 три последние функции и функцию своего варианта. Результат минимизации реализовать с помощью программируемых логических матриц (ПЛИС). Минимизацию проводить с помощью карт Карно. Минимизировать каждую функцию в отдельности (с помощью карт Карно) и результат из четырех функций реализовать на ПЛИС. Сравнить две реализации и указать, какая из них экономнее.

Алгоритм совместной минимизации системы из k функций (жадный алгоритм приближенной минимизации)

1. Найти все простые импликанты функции f_1 & f_2 & ... & f_k . Выбрать минимальное покрытие ее единиц простыми импликантами. Перевести область общих единиц в область неопределенности для каждой из данных функций.
2. Применять пункт 1, пока это возможно, по всем возможным произведениям $f_{i_1} \& f_{i_2} \& \dots \& f_{i_t}$, где $\{i_1, i_2, \dots, i_t\} \subseteq \{1, 2, \dots, k\}$, $t = k-1, \dots, 3, 2, 1$.

Пример.

$$\begin{aligned} f_1 &= 0011\ 0110\ 0110\ 1111, \\ f_2 &= 0011\ 0111\ 0011\ 0011, \\ f_3 &= 0000\ 0011\ 0111\ 1100. \end{aligned}$$

Решение. Для функций f_1, f_2, f_3 заполняем карты Карно (рис. П1.2).

		$\bar{z}$				z				
		$z\ t$		$z\ t$		$z\ t$		$z\ t$		
		00		01		11		10		
$\bar{x}$	xy	$\bar{x}$	00	0	0	1	1	$\bar{y}$	$\bar{y}$	
	01		0	1	1	0				
	11		0	1	1	0				
	10		1	1	1	1				
		x		$\bar{t}$		t		$\bar{t}$		

Строим функцию $f_1 \& f_2 \& f_3$ и находим ее карту Карно (рис. П1.3).

		$\bar{z}$		z	
		$z\ t$	$z\ t$	$z\ t$	$z\ t$
		00	01	11	10
$x\ y$	$\bar{x}$	0	0	0	0
		0	0	1	0
	x	0	0	1	0
		0	0	0	0
		$\bar{t}$		t	$\bar{t}$

Рис. П1.3

Две единицы функции $f_1 \& f_2 \& f_3$ покрывает ДНФ $d_{f_1 f_2 f_3} = yzt$.

Переводим единицы функции $f_1 \& f_2 \& f_3$ в область неопределенности (она обозначается знаком «—») функций f_1, f_2, f_3 и получаем функции f'_1, f'_2, f'_3 , задаваемые следующими (рис. П1.4) картами Карно.

		$\bar{z}$		z	
		$z\ t$	$z\ t$	$z\ t$	$z\ t$
		00	01	11	10
$x\ y$	$\bar{x}$	0	0	1	1
		0	1	—	0
	x	0	1	—	0
		1	1	1	1
		$\bar{t}$		t	$\bar{t}$

0	0	1	1
0	1	—	1
0	0	—	1
0	0	1	1

0	0	0	0
0	0	—	1
0	1	—	0
1	1	0	1

Рис. П1.4

Строим функцию $f'_1 \& f'_2$ и находим ее карту Карно (рис. П1.5).

Конъюнкция значений функций проводится в соответствии с операцией $\&$, определяемой следующей таблицей (рис. П1.6).



		$\bar{z}$		z		
		$z\ t$	$z\ t$	$z\ t$	$z\ t$	
		00	01	11	10	
$x\ y$	$\bar{x}$	0	0	1	1	$\bar{y}$
	01	0	0	—	0	y
	11	0	0	—	0	
	10	0	0	1	1	
		$\bar{t}$		t	$\bar{t}$	

Рис. П1.5

&	0	1	—
0	0	0	0
1	0	1	1
—	0	1	—

Рис. П1.6

Единицы функции f'_1 & f'_2 покрывает ДНФ $d_{f'_1 f'_2} = yzt$. $d_{f'_1 f'_2} = \bar{y}z \vee xyt$.

Переводим единицы функции f'_1 & f'_2 в область неопределенности функций f'_1, f'_2 и получаем функции f''_1, f''_2, f''_3 ($f''_3 = f'_3$), задаваемые следующими (рис. П1.7) картами Карно.

		$\bar{z}$		z			
		$z\ t$	$z\ t$	$z\ t$	$z\ t$		
		00	01	11	10		
x	$\bar{x}$	00	0	0	—	—	$\bar{y}$
	01	0	—	—	0		
	11	0	1	—	0		
	10	1	1	—	—		
		$\bar{t}$		t	$\bar{t}$		

0	0	—	—
0	—	—	1
0	0	—	1
0	0	—	—

f_1''

0	0	0	0
0	0	—	1
0	1	—	0
1	1	0	1

Рис. П1.7

Строим функцию f''_1 & f''_3 и находим ее карту Карно (рис. П1.8).

Единицы функции f''_1 & f''_3 покрывает ДНФ

$$d_{f''_1 f''_3} = x\bar{z}t \vee x\bar{y}z.$$

Переводим единицы функции f''_1 & f''_3 в область неопределенности функций f''_1, f''_3 и получаем функции f'''_1, f'''_2, f'''_3 ($f'''_2 = f''_2$), задаваемые следующими (рис. П1.9) картами Карно.

		$\bar{z}$		z		
		$z\ t$	$z\ t$	$z\ t$	$z\ t$	
		00	01	11	10	
x	$\bar{x}$	0 0	0	0	0	$\bar{y}$
	0 1	0	0	—	0	y
	1 1	0	1	—	0	
	1 0	1	1	0	0	
		$\bar{t}$		t	$\bar{t}$	

Рис. П1.8

		$\bar{z}$		z			
		$z\ t$	$z\ t$	$z\ t$	$z\ t$		
		00	01	11	10		
x	$\bar{x}$	0 0	0 0	—	—	y	$\bar{y}$
	0 1	0	—	—	0		
	1 1	0	1	—	0		
	1 0	1	1	—	0		
		$\bar{t}$		t	$\bar{t}$		

0	0	—	—
0	—	—	1
0	0	—	1
0	0	—	—

0	0	0	0
0	0	—	1
0	1	—	0
1	1	0	1

Рис. П1.9

Строим функцию f_2''' & f_3''' и находим ее карту Карно (рис. П1.10).

Единицы функции f_2''' & f_3''' покрывает ДНФ

$$d_{f_2''' f_3'''} = yz.$$

Переводим единицы функции f_2''' & f_3''' в область неопределенности функций f_2''' , f_3''' и получаем функции $f_1^{(4)}$, $f_2^{(4)}$, $f_3^{(4)}$, ($f_1^{(4)} = f_1'''$), задаваемые следующими (рис. П1.11) картами Карно

		$\bar{z}$		z			
		$z\ t$	$z\ t$	$z\ t$	$z\ t$		
		00	01	11	10		
x	$\bar{x}$	0 0	0	0	0	y	$\bar{y}$
	0 1	0	0	—	1		y
	1 1	0	0	—	1		$\bar{y}$
	1 0	0	0	0	0		y
		$\bar{t}$		t	$\bar{t}$		

Рис. П1.10

		$\bar{z}$		z			
		$z\ t$	$z\ t$	$z\ t$	$z\ t$		
		00	01	11	10		
x	$\bar{x}$	0 0	0	—	—	y	$\bar{y}$
	0 1	0	—	—	1		
	1 1	0	—	—	1		
	1 0	—	—	—	—		
		$\bar{t}$		t	$\bar{t}$		

0	0	—	—
0	—	—	0
0	0	—	0
0	0	—	—

0	0	0	0
0	0	—	—
0	—	—	—
—	—	0	0

Рис. П1.11

Единицы исчерпаны. Строим ДНФ-представления функций.

$$f_1(x, y, z, t) = d_{f_1 f_2 f_3} \vee d_{f_1' f_2'} \vee d_{f_1'' f_3''},$$

$$f_2(x, y, z, t) = d_{f_1 f_2 f_3} \vee d_{f_1' f_2'} \vee d_{f_2'' f_3''},$$

$$f_3(x, y, z, t) = d_{f_1 f_2 f_3} \vee d_{f_1'' f_3''} \vee d_{f_2'' f_3''},$$

откуда

$$f_1(x, y, z, t) = yzt \vee \bar{y}z \vee xy\bar{t} \vee x\bar{z}t \vee \bar{x}yz,$$

$$f_2(x, y, z, t) = yzt \vee \bar{y}z \vee xy\bar{t} \vee \bar{x}yz.$$

$$f_3(x, y, z, t) = yzt \vee \bar{y}z \vee xy\bar{t} \vee \bar{x}yz.$$

Ниже следует программируемая логическая матрица (ПЛМ), реализующая функции f_1, f_2, f_3 одновременно (рис. П1.12).

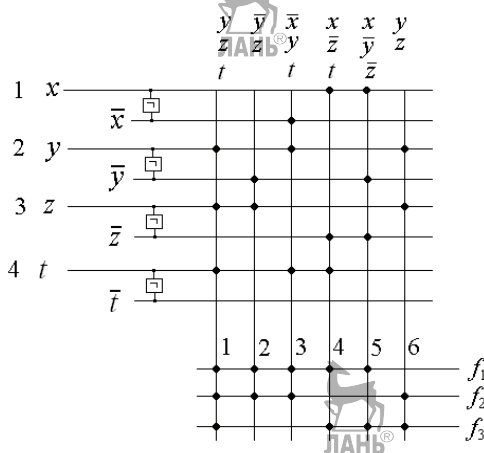


Рис. П1.12

ПЛМ имеет размер (4,6,3), где 4 есть местность функций, 6 есть ширина ПЛМ, 3 есть число реализуемых функций.

Замечание. Возможна совместная минимизация нескольких частично определенных функций. Конъюнкция значений функций проводится в соответствии с операцией $\&$, определяемой следующей таблицей.

$\&$	0	1	–
0	0	0	0
1	0	1	1
–	0	1	–

Задача 11. Заданную систему булевых функций исследовать на полноту с помощью теоремы Поста.

11.1. $(x \equiv y) + yz, x \& \neg y.$

11.2. $(x \rightarrow y) + (x \vee z), 0, 1.$

11.3. $x \equiv (y + z), \neg(x \rightarrow y) \equiv z.$

11.4. $(x + yz) \& \neg x \rightarrow z, xy.$

$$11.5. (x \equiv \neg y) \rightarrow (\neg x \equiv z), x \vee \neg y.$$

$$11.7. x + \neg(y \equiv z), \neg x \equiv y.$$

$$11.9. (x \rightarrow z) \mid y, \neg x \vee yz.$$

$$11.11. (x \equiv y) \rightarrow \neg z, x \vee \neg y.$$

$$11.13. (x \rightarrow y) + (y \rightarrow z), \neg x \& y.$$

$$11.15. (x + y) + (y \equiv \neg z), \neg x \rightarrow y.$$

$$11.17. \neg x \rightarrow y, 00, 11.$$

$$11.19. (x \rightarrow y) \vee \neg z, x \& \neg y.$$

$$11.21. (x \vee \neg y) \equiv z, (x \rightarrow y) \rightarrow y.$$

$$11.23. (x \mid y) \equiv (y \mid z), 00, 11.$$

$$11.25. (x \equiv \neg y) \rightarrow z, x \& \neg y.$$

$$11.27. (x + y \& \neg z) \rightarrow z, \neg x \& y.$$

$$11.29. (x \rightarrow yz) \vee \neg z, \neg x \vee yz.$$

$$11.6. (x \equiv \neg y) + xz, xy.$$

$$11.8. \neg x \equiv (y + z), xy.$$

$$11.10. (x \equiv y) \rightarrow (x \not\equiv z), 0.$$

$$11.12. (x \mid z) + y, x \equiv y \& \neg z.$$

$$11.14. (x \rightarrow y) \mid (y \rightarrow z), x + y.$$

$$11.16. x \equiv (y + \neg z), x.$$

$$11.18. (x \rightarrow y) + \neg z, x \vee \neg y.$$

$$11.20. (\neg x \equiv \neg y) \mid z, \neg x \equiv y.$$

$$11.22. x \mid z \rightarrow y, x \equiv y, 00.$$

$$11.24. (x \rightarrow \neg(yz)) \vee z, \neg x \rightarrow \neg y.$$

$$11.26. (\neg x \vee \neg yz) + z, x \rightarrow y.$$

$$11.28. x \vee \neg y \& z, \neg x \rightarrow y.$$

$$11.30. (x \vee \neg y) \equiv z, \neg x \rightarrow y.$$

Задача 12. Заданную систему булевых функций исследовать на полноту с помощью теоремы Поста.

$$12.1. 10110111, 01010100, 00100111.$$

$$12.3. 01010101, 0111, 00, 01010001.$$

$$12.5. 11101000, 1010, 00.$$

$$12.7. 10110001, 0011, 00.$$

$$12.9. 01001100, 1001.$$

$$12.11. 10101011, 1100, 11.$$

$$12.13. 01011000, 0101, 11.$$

$$12.15. 00011111, 1011, 00.$$

$$12.17. 10111000, 1011.$$

$$12.19. 01101101, 1001, 00.$$

$$12.21. 1011001, 1000, 00.$$

$$12.23. 11000111, 00011111, 00.$$

$$12.25. 10100011, 1001, 00.$$

$$12.27. 00110111, 1111, 00.$$

$$12.29. 01001011, 0001, 11.$$

$$12.2. 00110100, 11010101, 0111.$$

$$12.4. 11101110, 1100.$$

$$12.6. 10110001, 0001, 0000.$$

$$12.8. 10110001, 0010.$$

$$12.10. 00101011, 1100, 11.$$

$$12.12. 10010010, 0010, 11.$$

$$12.14. 01101110, 0000, 11.$$

$$12.16. 01101101, 0001, 11, 00.$$

$$12.18. 00111101, 1111, 00.$$

$$12.20. 00110011, 0101, 0011.$$

$$12.22. 10110001, 1001, 01.$$

$$12.24. 10100011, 0110.$$

$$12.26. 01001101, 1001, 00.$$

$$12.28. 00101001, 1101, 01.$$

$$12.30. 00001010, 1010, 11.$$

Задача 13. Реализовать функции из задач 5 и 6 с помощью мультиплексора (в базисе $\&, \vee, \neg, \text{MUX}(2)$).

Задача 14. Построить простую непересекающуюся декомпозицию функции $f(x_1, x_2, x_3, x_4, x_5) = f_1(x_1, x_2, x_3, f_2(x_4, x_5))$ и реализовать ее с помощью мультиплексора. Каждая функция задана множеством M_1 десятичных эквивалентов двоичных наборов, на которых функция принимает значение 1.

$$14.1. \{3, 8, 9, 10, 11, 20, 21, 22, 27\}.$$

$$14.2. \{4, 5, 6, 11, 19, 24, 25, 26, 27\}.$$

$$14.3. \{0, 1, 2, 3, 11, 19, 28, 29, 30\}.$$

$$14.4. \{7, 12, 13, 14, 15, 16, 17, 18, 31\}.$$



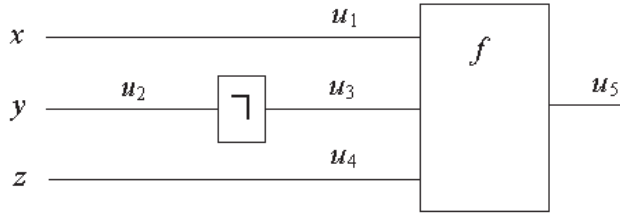
- 14.5. {1,8,9,10,11,20,22,23,25}.
 14.6. {2,8,9,10,11,20,21,23,26}.
 14.7. {3,12,13,14,16,17,18,19,27}.
 14.8. {0,1,2,15,23,28,29,30,31}.
 14.9. {4,6,7,9,17,24,25,26,27}.
 14.10. {4,9,10,11,20,21,22,23,28}.
 14.11. {4,5,7,10,18,24,25,26,27}.
 14.12. {4,5,6,7,15,23,24,25,26}.
 14.13. {0,1,2,3,9,17,28,30,31}.
 14.14. {0,1,2,3,10,18,28,29,31}.
 14.15. {5,12,13,14,15,16,18,19,29}.
 14.16. {6,12,13,14,15,16,17,19,30}.
 14.17. {0,8,9,10,11,21,22,23,24}.
 14.18. {7,8,9,10,20,21,22,23,31}.
 14.19. {5,8,10,11,20,21,22,23,29}.
 14.20. {2,12,13,15,16,17,18,19,26}.
 14.21. {4,5,6,7,13,21,24,26,27}.
 14.22. {4,5,6,7,14,22,24,25,27}.
 14.23. {0,1,2,3,8,16,29,30,31}.
 14.24. {4,12,13,14,15,17,18,19,28}.
 14.25. {1,2,3,4,9,10,11,16,20,21,22,23}.
 14.26. {1,2,9,10,12,14,20,21,22,23}.
 14.27. {11,12,13,14,28,29,30,31}.
 14.28. {12,13,14,15,20,25,26,27,28,29,30,31}.
 14.29. {12,13,14,16,17,18,27,31}.
 14.30. {8,9,10,11,17,18,20,23,25,26}.

Задача 15. Для булевой функции из задачи 4 построить минимальные проверяющие и полные тесты относительно указанных классов ошибок (s_{ij} – слипание каналов i и j , 0_i – обрыв канала i , 1_i – замыкание канала i).

15.1. $0_1, s_{24}$.	15.2. $0_3, s_{14}$.	15.3. $1_2, s_{34}$.
15.4. s_{13}, s_{24} .	15.5. $0_2, 1_3, s_{14}$.	15.6. $0_3, 1_1, s_{24}$.
15.7. $0_3, s_{14}$.	15.8. $1_3, s_{12}$.	15.9. s_{13}, s_{24} .
15.10. $0_1, 1_2, s_{34}$.	15.11. $0_1, 1_3, s_{24}$.	15.12. $1_3, s_{14}$.
15.13. $0_2, s_{14}$.	15.14. s_{23}, s_{24} .	15.15. $0_3, 1_2, s_{13}$.
15.16. $0_2, 1_1, s_{24}$.	15.17. $0_4, s_{14}$.	15.18. $0_3, s_{12}$.
15.19. $0_3, s_{13}, s_{24}$.	15.20. $0_1, 1_2, s_{14}$.	15.21. $0_3, 1_1, s_{24}$.
15.22. $1_1, s_{14}$.	15.23. $1_2, s_{14}$.	15.24. $0_1, s_{23}, s_{24}$.
15.25. $0_2, 1_3, s_{13}$.	15.26. $0_4, 1_1, s_{23}$.	15.27. $0_1, 1_3, s_{14}$.
15.28. $0_3, 1_2, s_{12}$.	15.29. $1_4, s_{13}, s_{24}$.	15.30. $0_3, s_{24}$.

Задача 16. Для данной схемы из функциональных элементов (СФЭ) найти:

- а) минимальный проверяющий тест;
 б) минимальный диагностический (полный) тест.



В вариантах заданий знак $\neg$ означает штрих Шеффера.

В данной схеме возможны следующие неисправности:

- 0_i , обрыв контакта u_i , т. е. $u_i \equiv 0$;
- 1_i , замыкание контакта u_i , т. е. $u_i \equiv 1$;
- $\neg_i$, вместо u_i реализуется $\neg u_i$.

Для данных трех неисправностей остальных двух неисправностей нет. Например, для задания 16.1. $f = x \& (y \rightarrow z)$, $0_1, 0_2, \neg_3$ в данной схеме присутствуют неисправности u_1, u_2, u_3 ; неисправностей u_4, u_5 нет.

16.1. $f = x \& (y \rightarrow z)$, $0_1, 0_2, \neg_3$.	16.2. $f = (x \& y) \rightarrow z$, $0_1, 0_2, 1_3$.
16.3. $f = (x \& y) z$, $0_1, 0_2, 1_4$.	16.4. $f = (x \& y) + z$, $0_1, 0_2, 1_4$.
16.5. $f = x \& y \& z$, $0_1, \neg_2, 1_4$.	16.6. $f = (x + y) \rightarrow z$, $1_1, 0_3, 0_4$.
16.7. $f = (x + y) z$, $1_1, \neg_3, 1_4$.	16.8. $f = (x + y) \& z$, $1_1, 1_3, 0_4$.
16.9. $f = (x \rightarrow y) \& z$, $1_1, 1_3, \neg_5$.	16.10. $f = (x \rightarrow y) z$, $0_1, 0_3, 1_4$.
16.11. $f = (x \rightarrow y) + z$, $0_1, 0_2, \neg_3$.	16.12. $f = (x y) \& z$, $0_1, 0_2, 0_4$.
16.13. $f = (x y) z$, $0_1, 1_2, 1_4$.	16.14. $f = (x y) + z$, $0_1, 1_2, 1_4$.
16.15. $f = x \& (y \rightarrow z)$, $1_1, \neg_3, 0_4$.	16.16. $f = (x \& y) \rightarrow z$, $1_1, 0_3, 1_4$.
16.17. $f = (x \& y) z$, $1_1, 1_3, 0_4$.	16.18. $f = (x \& y) + z$, $1_1, 1_3, 1_4$.
16.19. $f = x \& y \& z$, $0_1, 0_2, \neg_5$.	16.20. $f = (x + y) \rightarrow z$, $0_1, 1_3, \neg_4$.
16.21. $f = (x + y) z$, $0_1, 0_2, 0_4$.	16.22. $f = (x + y) \& z$, $0_1, 0_2, 1_4$.
16.23. $f = (x \rightarrow y) \& z$, $0_1, 1_2, 1_4$.	16.24. $f = (x \rightarrow y) z$, $1_1, 0_3, 0_4$.
16.25. $f = x \vee (y \& z)$, $1_1, 0_2, 3_4$.	16.26. $f = x \vee (y z)$, $1_1, \neg_3, 0_4$.
16.27. $f = x \vee (y \rightarrow z)$, $0_1, \neg_2, 1_5$.	16.28. $f = x \vee (y + z)$, $0_1, \neg_3, 1_4$.
16.29. $f = x \& (y \vee z)$, $1_1, \neg_4, 0_5$.	16.30. $f = x + (y \vee z)$, $1_1, 1_3, \neg_5$.

Задача 17. Заданы формула логики предикатов A и двухэлементное множество $M = \{1, 2\}$. Привести формулу A к префиксной нормальной форме. Является ли формула A на множестве M : 1) выполнимой; 2) опровержимой; 3) общезначимой; 4) невыполнимой? Вычислить значение истинности формулы A на множестве M со следующими предикатами, определенными на M .

x	1	2
$P(x)$	1	0
$R(x)$	0	1

$Q(x, y)$	1	2
1	1	0
2	0	0

- 17.1. $(\forall x)(P(x) \& R(x) \rightarrow (\exists y)Q(x, y))$.
 17.2. $(\forall x)(P(x) \rightarrow (R(x) \rightarrow (\exists y)Q(x, y)))$.

- 17.3. $(\forall x)(P(x) \& \neg R(x) \rightarrow (\exists y)Q(x,y)).$
 17.4. $(\forall x)(\neg P(x) \rightarrow (\neg R(x) \rightarrow (\exists y)\neg Q(x,y))).$
 17.5. $(\forall x)(\neg P(x) \vee \neg R(x) \rightarrow (\exists y)Q(x,y)).$
 17.6. $(\exists x)(P(x) \& R(x) \rightarrow (\forall y)Q(x,y)).$
 17.7. $(\exists x)(P(x) \rightarrow (R(x) \rightarrow (\forall y)Q(x,y))).$
 17.8. $(\exists x)(P(x) \vee \neg(R(x) \rightarrow (\forall y)Q(x,y))).$
 17.9. $(\exists x)(\neg P(x) \rightarrow (\neg R(x) \rightarrow (\forall y)Q(x,y))).$
 17.10. $(\exists x)(\neg P(x) \vee \neg R(x) \rightarrow (\exists y)\neg Q(x,y)).$
 17.11. $(\forall y)(P(y) \& R(y) \rightarrow (\exists x)Q(x,y)).$
 17.12. $(\forall y)(P(y) \rightarrow (R(y) \rightarrow (\exists x)Q(x,y))).$
 17.13. $(\forall y)(P(y) \vee \neg R(y) \rightarrow (\exists x)Q(x,y)).$
 17.14. $(\forall y)(\neg P(y) \rightarrow (\neg R(y) \rightarrow (\exists x)Q(x,y))).$
 17.15. $(\forall y)(P(y) \rightarrow (\neg R(x) \rightarrow (\exists x)Q(x,y))).$
 17.16. $(\forall x)(P(x) \& R(x) \rightarrow (\forall y)Q(x,y)).$
 17.17. $(\forall x)(P(x) \rightarrow (R(x) \rightarrow (\forall y)Q(x,y))).$
 17.18. $(\forall x)(P(x) \vee \neg P(x) \rightarrow (\forall y)Q(x,y)).$
 17.19. $(\forall x)((P(x) \rightarrow R(x)) \rightarrow (\forall y)Q(x,y)).$
 17.20. $(\forall x)(\neg P(x) \vee \neg R(x) \rightarrow (\forall y)Q(x,y)).$
 17.21. $(\exists y)(\forall x)(Q(x,y) \& R(x) \rightarrow (\forall y)P(y)).$
 17.22. $(\exists y)((\exists x)(Q(x,y) \rightarrow P(x) \vee Q(x,y))).$
 17.23. $(\exists y)((\forall x)(Q(x,y) \rightarrow \neg P(x) \vee \neg Q(x,y))).$
 17.24. $(\forall y)((\exists x)(Q(x,y) \rightarrow (P(x) \rightarrow Q(x,y))).$
 17.25. $(\forall x)((\exists y)Q(x,y) \rightarrow (R(x) \rightarrow P(x))).$
 17.26. $(\forall x)(P(x) \rightarrow (\exists y)(Q(x,y) \rightarrow R(x))).$
 17.27. $(\exists x)(P(x) \rightarrow (\exists y)(Q(x,y) \rightarrow \neg R(x))).$
 17.28. $(\exists y)(P(y) \rightarrow (\forall x)Q(x,y) \rightarrow R(y)).$
 17.29. $(\forall y)(P(y) \rightarrow (\forall x)(Q(y,x) \rightarrow \neg R(x))).$
 17.30. $(\forall y)(P(y) \rightarrow (\forall x)(Q(x,y) \vee \neg R(x))).$



Задача 18. Установить правильность или неправильность правил вывода, установив общезначимость соответствующей формулы.

$$18.1. \frac{P \rightarrow \neg M, S \& M}{S \& \neg P}.$$

$$18.3. \frac{M \rightarrow P, S \& M}{S \& P}.$$

$$18.5. \frac{P \rightarrow \neg M, M \& S}{S \& \neg P}.$$

$$18.7. \frac{P \& M, M \rightarrow S}{S \& P}.$$

$$18.9. \frac{M \rightarrow \neg P, M \rightarrow S, M}{S \& \neg P}.$$

$$18.2. \frac{P \rightarrow \neg M, M \rightarrow S, M}{S \& \neg P}.$$

$$18.4. \frac{P \rightarrow M, M \rightarrow S, P}{S \& P}.$$

$$18.6. \frac{P \rightarrow M, M \rightarrow \neg S}{S \rightarrow \neg P}.$$

$$18.8. \frac{P \rightarrow M, M \rightarrow \neg S, P}{\neg S \& P}.$$

$$18.10. \frac{M \rightarrow P, M \rightarrow S, M}{P \& S}.$$

$$18.11. \frac{M \rightarrow \neg P, M \& S}{S \& \neg P}.$$

$$18.13. \frac{M \& P, M \rightarrow S}{S \& P}.$$

$$18.15. \frac{P \rightarrow \neg M, S \rightarrow M, S}{S \& \neg P}.$$

$$18.17. \frac{P \rightarrow \neg M, S \& M}{S \& \neg P}.$$

$$18.19. \frac{P \rightarrow \neg M, S \rightarrow M}{S \rightarrow \neg P}.$$

$$18.21. \frac{M \rightarrow \neg P, S \rightarrow M, S}{S \& \neg P}.$$

$$18.23. \frac{M \rightarrow P, S \& M}{S \& P}.$$

$$18.25. \frac{M \rightarrow P, S \rightarrow M}{S \rightarrow P}.$$

$$18.27. \frac{P \rightarrow \neg M, S \rightarrow M, S}{S \& \neg P}.$$

$$18.29. \frac{M \rightarrow \neg P, \neg M \rightarrow \neg S}{S \rightarrow \neg P}.$$

$$18.12. \frac{M \rightarrow P, M \& S}{S \rightarrow P}.$$

$$18.14. \frac{M \& \neg P, M \rightarrow S}{S \rightarrow \neg P}.$$

$$18.16. \frac{P \rightarrow M, S \rightarrow \neg M, S}{S \& P}.$$

$$18.18. \frac{P \rightarrow M, S \rightarrow \neg M}{S \rightarrow \neg P}.$$

$$18.20. \frac{P \rightarrow M, S \& \neg M}{S \& \neg P}.$$

$$18.22. \frac{M \rightarrow P, S \rightarrow M, S}{S \& P}.$$

$$18.24. \frac{M \rightarrow \neg P, S \rightarrow M}{S \rightarrow \neg P}.$$

$$18.26. \frac{M \rightarrow \neg P, M \rightarrow \neg S}{\neg M \rightarrow \neg P}.$$

$$18.28. \frac{\neg P \rightarrow \neg M, S \rightarrow M, S}{S \& P}.$$

$$18.30. \frac{\neg M \rightarrow \neg P, S \& \neg M}{S \& \neg P}.$$

Задача 19. Установить правильность или неправильность правил вывода, установив общезначимость соответствующей формулы.

$$19.1. \frac{P \rightarrow \neg M, \neg S \& M}{\neg S \& \neg P}.$$

$$19.3. \frac{M \rightarrow \neg P, \neg S \& M}{\neg S \& P}.$$

$$19.5. \frac{\neg P \rightarrow \neg M, \neg M \& S}{\neg S \& \neg P}.$$

$$19.7. \frac{\neg P \& M, \neg M \rightarrow S}{S \& \neg P}.$$

$$19.9. \frac{M \rightarrow \neg P, M \rightarrow S, M}{\neg S \& \neg P}.$$

$$19.11. \frac{M \rightarrow \neg P, \neg M \& S}{\neg S \& \neg P}.$$

$$19.13. \frac{M \& P, \neg M \rightarrow S}{S \& P}.$$

$$19.15. \frac{P \rightarrow \neg M, \neg S \rightarrow M, \neg S}{S \& \neg P}.$$

$$19.2. \frac{P \rightarrow \neg M, \neg M \rightarrow S, M}{\neg S \& \neg P}.$$

$$19.4. \frac{P \rightarrow M, \neg M \rightarrow S, \neg P}{\neg S \& P}.$$

$$19.6. \frac{\neg P \rightarrow M, M \rightarrow \neg S}{\neg S \rightarrow \neg P}.$$

$$19.8. \frac{\neg P \rightarrow M, \neg M \rightarrow \neg S, \neg P}{S \& P}.$$

$$19.10. \frac{\neg M \rightarrow P, M \rightarrow \neg S, \neg P}{\neg P \& S}.$$

$$19.12. \frac{M \rightarrow \neg P, \neg M \& S}{\neg S \rightarrow P}.$$

$$19.14. \frac{\neg M \& \neg P, \neg M \rightarrow S}{S \rightarrow \neg P}.$$

$$19.16. \frac{\neg P \rightarrow M, \neg S \rightarrow \neg M, S}{S \& \neg P}.$$





$$19.17. \frac{P \rightarrow \neg M, \neg S \rightarrow M}{\neg S \& \neg P}.$$

$$19.19. \frac{P \rightarrow \neg M, \neg S \rightarrow M}{\neg S \rightarrow \neg P}.$$

$$19.21. \frac{\neg M \rightarrow \neg P, \neg S \rightarrow M, S}{\neg S \& \neg P}.$$

$$19.23. \frac{\neg M \rightarrow \neg P, S \& M}{S \& \neg P}.$$

$$19.25. \frac{\neg M \rightarrow P, S \rightarrow \neg M}{\neg S \rightarrow \neg P}.$$

$$19.27. \frac{P \rightarrow \neg M, S \rightarrow M, \neg S}{S \& \neg P}.$$

$$19.29. \frac{\neg M \rightarrow \neg P, \neg M \rightarrow \neg S}{S \rightarrow \neg P}.$$

$$19.18. \frac{\neg P \rightarrow M, S \rightarrow \neg M}{\neg S \rightarrow \neg P}.$$

$$19.20. \frac{\neg P \rightarrow M, S \& \neg M}{\neg S \& \neg P}.$$

$$19.22. \frac{\neg M \rightarrow P, S \rightarrow \neg M, \neg S}{\neg S \& \neg P}.$$

$$19.24. \frac{\neg M \rightarrow \neg P, \neg S \rightarrow M}{\neg S \rightarrow \neg P}.$$

$$19.26. \frac{\neg M \rightarrow \neg P, M \rightarrow \neg S}{\neg S \rightarrow \neg P}.$$

$$19.28. \frac{\neg P \rightarrow \neg M, S \rightarrow \neg M, S}{S \& P}.$$

$$19.30. \frac{\neg M \rightarrow \neg P, \neg S \& \neg M}{S \& \neg P}.$$

Задача 20. Установить правильность или неправильность правил вывода, используя естественный вывод Генцена. Задание взять из задачи 18.

Задача 21. Установить правильность или неправильность правил вывода, используя естественный вывод Генцена. Задание взять из задачи 19.

Задача 22. Установить правильность или неправильность правил вывода, используя метод резолюций. Задание взять из задачи 18.

Задача 23. Установить правильность или неправильность правил вывода, используя метод резолюций. Задание взять из задачи 19.

Задача 24. Доказать или опровергнуть невыполнимость множества дизъюнктов S путем построения обрезанного семантического дерева и построить вывод пустого дизъюнкта из S в случае невыполнимости S .

$$24.1. p \vee q \vee r, p \vee q \vee r, \neg q \vee \neg r, q, r.$$

$$24.2. p \vee q \vee r, p \vee \neg q \vee r, \neg q \vee \neg r, q, r.$$

$$24.3. p \vee q \vee r, p \vee \neg q \vee \neg r, \neg q \vee \neg r, q, r.$$

$$24.4. p \vee q \vee r, \neg p \vee q \vee r, \neg q \vee \neg r, q, r.$$

$$24.5. p \vee q \vee r, \neg p \vee q \vee \neg r, \neg q \vee \neg r, q, r.$$

$$24.6. p \vee q \vee r, \neg p \vee \neg q \vee r, \neg q \vee \neg r, q, r.$$

$$24.7. p \vee q \vee r, \neg p \vee \neg q \vee \neg r, \neg q \vee \neg r, q, r.$$

$$24.8. p \vee q \vee \neg r, p \vee \neg q \vee r, \neg q \vee \neg r, q, r.$$

$$24.9. p \vee q \vee \neg r, p \vee \neg q \vee \neg r, \neg q \vee \neg r, q, r.$$

$$24.10. p \vee q \vee \neg r, \neg p \vee q \vee r, \neg q \vee \neg r, q, r.$$

$$24.11. p \vee q \vee \neg r, \neg p \vee q \vee \neg r, \neg q \vee \neg r, q, r.$$

$$24.12. p \vee q \vee \neg r, \neg p \vee \neg q \vee r, \neg q \vee \neg r, q, r.$$

$$24.13. p \vee q \vee \neg r, \neg p \vee \neg q \vee \neg r, \neg q \vee \neg r, q, r.$$

$$24.14. p \vee \neg q \vee r, p \vee \neg q \vee \neg r, \neg q \vee \neg r, q, r.$$

- 24.15. $\neg p \vee \neg q \vee \neg r, \neg p \vee q \vee r, \neg q \vee \neg r, q, r.$
 24.16. $\neg p \vee \neg q \vee \neg r, \neg p \vee q \vee \neg r, \neg q \vee \neg r, q, r.$
 24.17. $\neg p \vee \neg q \vee \neg r, \neg p \vee \neg q \vee r, \neg q \vee \neg r, q, r.$
 24.18. $p \vee \neg q \vee r, \neg p \vee \neg q \vee \neg r, \neg q \vee \neg r, q, r.$
 24.19. $p \vee \neg q \vee \neg r, \neg p \vee q \vee r, \neg q \vee \neg r, q, r.$
 24.20. $p \vee \neg q \vee \neg r, \neg p \vee q \vee \neg r, \neg q \vee \neg r, q, r.$
 24.21. $p \vee \neg q \vee \neg r, \neg p \vee \neg q \vee r, \neg q \vee \neg r, q, r.$
 24.22. $p \vee \neg q \vee \neg r, \neg p \vee \neg q \vee \neg r, \neg q \vee \neg r, q, r.$
 24.23. $\neg p \vee q \vee r, \neg p \vee q \vee \neg r, \neg q \vee \neg r, q, r.$
 24.24. $\neg p \vee q \vee r, \neg p \vee \neg q \vee r, \neg q \vee \neg r, q, r.$
 24.25. $\neg p \vee q \vee r, \neg p \vee \neg q \vee \neg r, \neg q \vee \neg r, q, r.$
 24.26. $\neg p \vee q \vee \neg r, \neg p \vee \neg q \vee r, \neg q \vee \neg r, q, r.$
 24.27. $\neg p \vee q \vee \neg r, \neg p \vee \neg q \vee \neg r, \neg q \vee \neg r, q, r.$
 24.28. $\neg p \vee \neg q \vee r, \neg p \vee \neg q \vee \neg r, \neg q \vee \neg r, q, r.$
 24.29. $p \vee q \vee r, \neg p \vee \neg r, \neg q \vee \neg r, q, r.$
 24.30. $p \vee q \vee \neg r, \neg q \vee r, \neg q \vee \neg r, q, r.$



Задача 25. Доказать или опровергнуть невыполнимость множества дизъюнктов S путем построения обрезанного семантического дерева и построить вывод пустого дизъюнкта из S в случае невыполнимости S .

- 25.1. $\neg p \vee \neg q \vee \neg r, \neg p \vee \neg q \vee r, \neg p \vee q \vee \neg r.$
 25.2. $\neg p \vee \neg q \vee \neg r, \neg p \vee \neg q \vee r, \neg p \vee q \vee r.$
 25.3. $\neg p \vee \neg q \vee \neg r, \neg p \vee \neg q \vee r, p \vee \neg q \vee \neg r.$
 25.4. $\neg p \vee \neg q \vee \neg r, \neg p \vee \neg q \vee r, p \vee \neg q \vee r.$
 25.5. $\neg p \vee \neg q \vee \neg r, \neg p \vee \neg q \vee r, p \vee q \vee \neg r.$
 25.6. $\neg p \vee \neg q \vee \neg r, \neg p \vee \neg q \vee r, p \vee q \vee r.$
 25.7. $\neg p \vee \neg q \vee \neg r, \neg p \vee q \vee \neg r, \neg p \vee q \vee r.$
 25.8. $\neg p \vee \neg q \vee \neg r, \neg p \vee q \vee \neg r, p \vee \neg q \vee \neg r.$
 25.9. $\neg p \vee \neg q \vee \neg r, \neg p \vee q \vee \neg r, p \vee \neg q \vee r.$
 25.10. $\neg p \vee \neg q \vee \neg r, \neg p \vee q \vee \neg r, \neg p \vee \neg q \vee r.$
 25.11. $\neg p \vee \neg q \vee \neg r, \neg p \vee q \vee \neg r, p \vee q \vee r.$
 25.12. $\neg p \vee \neg q \vee \neg r, \neg p \vee q \vee r, p \vee \neg q \vee \neg r.$
 25.13. $\neg p \vee \neg q \vee \neg r, \neg p \vee q \vee r, p \vee \neg q \vee r.$
 25.14. $\neg p \vee \neg q \vee \neg r, \neg p \vee q \vee r, p \vee q \vee \neg r.$
 25.15. $\neg p \vee \neg q \vee \neg r, \neg p \vee q \vee r, p \vee q \vee r.$
 25.16. $\neg p \vee \neg q \vee \neg r, p \vee \neg q \vee \neg r, p \vee \neg q \vee r.$
 25.17. $\neg p \vee \neg q \vee \neg r, p \vee \neg q \vee \neg r, p \vee q \vee \neg r.$
 25.18. $\neg p \vee \neg q \vee \neg r, p \vee \neg q \vee \neg r, p \vee q \vee r.$
 25.19. $\neg p \vee \neg q \vee \neg r, p \vee \neg q \vee r, p \vee q \vee \neg r.$
 25.20. $\neg p \vee \neg q \vee \neg r, p \vee \neg q \vee r, p \vee q \vee r.$
 25.21. $\neg p \vee \neg q \vee \neg r, p \vee q \vee \neg r, p \vee q \vee r.$
 25.22. $\neg p \vee \neg q \vee r, \neg p \vee q \vee \neg r, \neg p \vee q \vee r.$
 25.23. $\neg p \vee \neg q \vee r, p \vee q \vee r, p \vee \neg q \vee \neg r.$

- 25.24. $\neg p \vee \neg q \vee r, p \vee q \vee r, p \vee \neg q \vee r.$
 25.25. $\neg p \vee \neg q \vee r, p \vee q \vee r, p \vee \neg q \vee \neg r.$
 25.26. $\neg p \vee \neg q \vee r, \neg p \vee q \vee \neg r, p \vee q \vee r.$
 25.27. $\neg p \vee \neg q \vee r, \neg p \vee q \vee r, p \vee \neg q \vee \neg r.$
 25.28. $\neg p \vee \neg q \vee r, \neg p \vee q \vee r, p \vee \neg q \vee r.$
 25.29. $\neg p \vee \neg q \vee r, p \vee q \vee r, p \vee q \vee \neg r.$
 25.30. $\neg p \vee \neg q \vee r, \neg p \vee q \vee r, p \vee q \vee r.$

Задача 26. Доказать правильность правил вывода, установив общезначимость соответствующей формулы.

Пример.

$$\begin{aligned}
 & \frac{(\forall x)(M(x) \rightarrow \overline{P(x)}), (\forall x)(M(x) \rightarrow S(x)), (\exists x)M(x)}{(\exists x)(S(x) \& \overline{P(x)})}. A = A(M, P, S) = \\
 & \frac{(\forall x)(M(x) \rightarrow \overline{P(x)}) \& (\forall x)(M(x) \rightarrow S(x)) \& (\exists x)M(x)}{(\exists x)(S(x) \& \overline{P(x)})} = \\
 & \frac{1 \& 2 \& 3}{\rightarrow 4} = \overline{1 \& 2 \& 3} \vee 4 = \overline{1} \vee \overline{2} \vee \overline{3} \vee 4 = \overline{1} \vee \overline{2} \vee 4 \vee \overline{3} = \\
 & \frac{(\forall x)(M(x) \rightarrow \overline{P(x)}) \vee (\forall x)(M(x) \rightarrow S(x)) \vee (\exists x)(S(x) \& \overline{P(x)}) \vee (\exists x)M(x)}{\overline{1} \quad \quad \quad \overline{2} \quad \quad \quad 4 \quad \quad \quad \overline{3}} = \\
 & (\exists x)(\overline{M(x) \vee P(x)}) \vee (\exists x)(\overline{M(x) \vee S(x)}) \vee (\exists x)(S(x) \& \overline{P(x)}) \vee \overline{3} = \\
 & (\exists x)(\overline{M(x)} \& \overline{P(x)}) \vee (\exists x)(\overline{M(x)} \& \overline{S(x)}) \vee (\exists x)(S(x) \& \overline{P(x)}) \vee \overline{3} = \\
 & (\exists x)(M(x) \& P(x) \vee M(x) \& \overline{S(x)} \vee S(x) \& \overline{P(x)}) \vee \overline{3} = \\
 & (\exists x)(MP \vee (M \vee S)(M \vee \overline{P})(\overline{S} \vee S)(\overline{S} \vee \overline{P})) \vee \overline{3} = \\
 & (\exists x)(MP \vee (M \vee S)(M \vee \overline{P})(\overline{S} \vee \overline{P})) \vee \overline{3} = \\
 & \quad \quad \quad \text{умножить} \\
 & (\exists x)(MP \vee (M \vee M\overline{P} \vee SM \vee S\overline{P})(\overline{S} \vee \overline{P})) \vee \overline{3} = \\
 & (\exists x)(MP \vee (M \vee S\overline{P})(\overline{S} \vee \overline{P})) \vee \overline{3} = \\
 & (\exists x)(MP \vee M\overline{S} \vee M\overline{P} \vee S\overline{P}\overline{S} \vee S\overline{P}) \vee \overline{3} = \\
 & (\exists x)(M(P \vee \overline{P}) \vee M\overline{S} \vee S\overline{P}) \vee \overline{3} = (\exists x)(M \vee M\overline{S} \vee S\overline{P}) \vee \overline{3} = \\
 & (\exists x)(M \vee S\overline{P}) \vee \overline{3} = (\exists x)M \vee (\exists x)(S\overline{P}) \vee \overline{3} = \\
 & (\exists x)M(x) \vee (\exists x)(S(x) \& \overline{P(x)}) \vee (\exists x)M(x) \equiv 1. \text{ Правило вывода верно.} \\
 26.1. & \frac{(\forall x)(P(x) \rightarrow \neg M(x)), (\exists x)(S(x) \& M(x))}{(\exists x)(S(x) \& \neg P(x))}. \\
 26.2. & \frac{(\forall x)(P(x) \rightarrow \neg M(x)), (\forall x)(M(x) \rightarrow S(x)), (\exists x)M(x)}{(\exists x)(S(x) \& \neg P(x))}.
 \end{aligned}$$

- 26.3. $\frac{(\forall x)(M(x) \rightarrow P(x)), (\exists x)(S(x) \& M(x))}{(\exists x)(S(x) \& P(x))}$.
- 26.4. $\frac{(\forall x)(P(x) \rightarrow M(x)), (\forall x)(M(x) \rightarrow S(x)), (\exists x)P(x)}{(\exists x)(S(x) \& P(x))}$.
- 26.5. $\frac{(\forall x)(P(x) \rightarrow \neg M(x)), (\exists x)(M(x) \& S(x))}{(\exists x)(S(x) \& \neg P(x))}$.
- 26.6. $\frac{(\forall x)(P(x) \rightarrow M(x)), (\forall x)(M(x) \rightarrow \neg S(x))}{(\forall x)(S(x) \rightarrow \neg P(x))}$.
- 26.7. $\frac{(\exists x)(P(x) \& M(x)), (\forall x)(M(x) \rightarrow S(x))}{(\exists x)(S(x) \& P(x))}$.
- 26.8. $\frac{(\forall x)(P(x) \rightarrow M(x)), (\forall x)(M(x) \rightarrow \neg S(x)), (\exists x)S(x)}{(\exists x)(S(x) \& \neg P(x))}$.
- 26.9. $\frac{(\forall x)(M(x) \rightarrow \neg P(x)), (\forall x)(M(x) \rightarrow S(x)), (\exists x)M(x)}{(\exists x)(S(x) \& \neg P(x))}$.
- 26.10. $\frac{(\forall x)(M(x) \rightarrow P(x)), (\forall x)(M(x) \rightarrow S(x)), (\exists x)M(x)}{(\exists x)(P(x) \& S(x))}$.
- 26.11. $\frac{(\forall x)(M(x) \rightarrow \neg P(x)), (\exists x)(M(x) \& S(x))}{(\exists x)(S(x) \& \neg P(x))}$.
- 26.12. $\frac{(\forall x)(M(x) \rightarrow P(x)), (\exists x)(M(x) \& S(x))}{(\exists x)(S(x) \rightarrow P(x))}$.
- 26.13. $\frac{(\exists x)(M(x) \& P(x)), (\forall x)(M(x) \rightarrow S(x))}{(\exists x)(S(x) \& P(x))}$.
- 26.14. $\frac{(\exists x)(M(x) \& \neg P(x)), (\forall x)(M(x) \rightarrow S(x))}{(\exists x)(S(x) \rightarrow \neg P(x))}$.
- 26.15. $\frac{(\forall x)(P(x) \rightarrow \neg M(x)), (\forall x)(S(x) \& M(x)), (\exists x)S(x)}{(\exists x)(S(x) \& \neg P(x))}$.
- 26.16. $\frac{(\forall x)(P(x) \rightarrow M(x)), (\forall x)(S(x) \rightarrow \neg M(x)), (\exists x)S(x)}{(\exists x)(S(x) \& \neg P(x))}$.
- 26.17. $\frac{(\forall x)(P(x) \rightarrow \neg M(x)), (\exists x)(S(x) \rightarrow M(x))}{(\exists x)(S(x) \& \neg P(x))}$.
- 26.18. $\frac{(\forall x)(P(x) \rightarrow M(x)), (\forall x)(S(x) \rightarrow \neg M(x))}{(\forall x)(S(x) \rightarrow \neg P(x))}$.
- 26.19. $\frac{(\forall x)(P(x) \rightarrow \neg M(x)), (\forall x)(S(x) \rightarrow M(x))}{(\forall x)(S(x) \rightarrow \neg P(x))}$.
- 26.20. $\frac{(\forall x)(P(x) \rightarrow M(x)), (\exists x)(S(x) \& \neg M(x))}{(\exists x)(S(x) \& \neg P(x))}$.



$$26.21. \frac{(\forall x)(M(x) \rightarrow \neg P(x)), (\forall x)(S(x) \rightarrow M(x)), (\exists x)S(x)}{(\exists x)(S(x) \& \neg P(x))}.$$

$$26.22. \frac{(\forall x)(M(x) \rightarrow P(x)), (\forall x)(S(x) \rightarrow M(x)), (\exists x)S(x)}{(\exists x)(S(x) \& P(x))}.$$

$$26.23. \frac{(\forall x)(M(x) \rightarrow P(x)), (\exists x)(S(x) \& M(x))}{(\exists x)(S(x) \& P(x))}.$$

$$26.24. \frac{(\forall x)(M(x) \rightarrow \neg P(x)), (\forall x)(S(x) \rightarrow M(x))}{(\forall x)(S(x) \rightarrow \neg P(x))}.$$

$$26.25. \frac{(\forall x)(M(x) \rightarrow P(x)), (\forall x)(S(x) \rightarrow M(x))}{(\forall x)(S(x) \rightarrow P(x))}.$$

$$26.26. \frac{(\forall x)(\neg M(x) \rightarrow \neg P(x)), (\forall x)(M(x) \rightarrow \neg S(x))}{(\forall x)(S(x) \rightarrow \neg P(x))}.$$

$$26.27. \frac{(\forall x)(\neg P(x) \rightarrow \neg M(x)), (\forall x)(S(x) \rightarrow M(x)), (\exists x)S(x)}{(\exists x)(S(x) \& P(x))}.$$

$$26.28. \frac{(\forall x)(\neg P(x) \rightarrow \neg M(x)), (\forall x)(S(x) \rightarrow M(x)), (\exists x)S(x)}{(\exists x)(S(x) \& P(x))}.$$

$$26.29. \frac{(\forall x)(M(x) \rightarrow \neg P(x)), (\forall x)(\neg M(x) \rightarrow \neg S(x))}{(\forall x)(S(x) \rightarrow \neg P(x))}.$$

$$26.30. \frac{(\forall x)(\neg M(x) \rightarrow \neg P(x)), (\exists x)(S(x) \& \neg M(x))}{(\exists x)(S(x) \& \neg P(x))}.$$

Задача 27. Доказать справедливость правил вывода, используя естественный вывод Генцена. Задание взять из задачи 26.

Задача 28. Доказать справедливость правил вывода путем построения обрезанного семантического дерева (указав сначала префиксную и сколемову формы соответствующей формулы, эрбранов универсум и эрбранов базис). Задание взять из задачи 26.

Задача 29. Доказать справедливость правил вывода путем нахождения опровергающего множества основных примеров (указав сначала префиксную и сколемову формы соответствующей формулы, эрбранов универсум и эрбранов базис). Задание взять из задачи 26.

Задача 30. Задание взять из задачи 26. Доказать справедливость правил вывода методом резолюций, для чего выполнить следующее.

а. Построить формулу A , для которой правило вывода верно $\leftrightarrow$ формула A общезначима $\leftrightarrow$ формула $\neg A$ невыполнима.

б. Найти префиксную нормальную форму для формулы A .

в. Найти префиксную нормальную форму для формулы $\neg A$.

г. Найти стандартную форму Сколема для формулы $\neg A$.

д. Указать множество дизъюнктов S для формулы $\neg A$.

- е. Написать эрбранов универсум H для S .
 ж. Написать эрбранов базис B для S .
 з. Указать множество основных примеров дизъюнктов из S .
 и. Построить обрезанное семантическое дерево для S и сделать вывод о верности данного правила вывода.
 к. Найти (конечное) множество основных примеров, опровергающих каждую H -интерпретацию, а потому и все интерпретации множества дизъюнктов S . Сделать вывод о верности данного правила вывода.

Решение. Правило вывода

$$\text{а. ПВ} = \frac{(\forall x)(\overline{P(x)} \rightarrow \overline{M(x)}), (\forall x)(\overline{M(x)} \rightarrow \overline{S(x)}), (\exists x)S(x)}{(\exists x)(S(x) \& P(x))} \text{ верно } \leftrightarrow \text{ формула}$$

$$A = A(M, P, S) =$$

$$(\forall x)(\overline{P(x)} \rightarrow \overline{M(x)}) \& (\forall x)(\overline{M(x)} \rightarrow \overline{S(x)}) \& (\exists x)S(x) \rightarrow (\exists x)(S(x) \& P(x))$$

общезначима $\leftrightarrow$ формула $\neg A = \neg A(M, P, S)$ невыполнима. Формула

$$A = A(M, P, S) =$$

$$(\forall x)(\overline{P(x)} \rightarrow \overline{M(x)}) \& (\forall x)(\overline{M(x)} \rightarrow \overline{S(x)}) \& (\exists x)S(x) \rightarrow (\exists x)(S(x) \& P(x)) \equiv 1.$$

б. Приведем формулу A к префиксной нормальной форме. Формула

$$\begin{aligned} A &= (\forall x)(\overline{P(x)} \vee \overline{M(x)}) \& (\forall x)(\overline{M(x)} \vee \overline{S(x)}) \& (\exists x)S(x) \vee (\exists x)(S(x) \& P(x)) = \\ &= (\exists x)(\overline{P(x)} \& \overline{M(x)}) \vee (\exists x)(\overline{M(x)} \& \overline{S(x)}) \vee (\forall x)\overline{S(x)} \vee (\exists x)(S(x) \& P(x)) = \\ &= (\exists x)(\forall y)(\overline{P(x)} \& \overline{M(x)} \vee \overline{M(x)} \& \overline{S(x)} \vee \overline{S(y)} \vee S(x) \& P(x)). \end{aligned}$$

в. Префиксная нормальная форма формулы

$$\begin{aligned} \neg A &= (\forall x)(\exists y)(\overline{P(x)} \& \overline{M(x)} \vee \overline{M(x)} \& \overline{S(x)} \vee \overline{S(y)} \vee S(x) \& P(x)) = \\ &= (\forall x)(\exists y)((\overline{P(x)} \vee \overline{M(x)}) \& (\overline{M(x)} \vee \overline{S(x)}) \& \overline{S(y)} \& (S(x) \vee \overline{P(x)})) = \\ &= (\forall x)(\exists y)((P(x) \vee \overline{M(x)}) \& (M(x) \vee \overline{S(x)}) \& S(y) \& (\overline{S(x)} \vee \overline{P(x)})). \end{aligned}$$

г. Стандартная форма Сколема формулы $\neg A =$

$$(\forall x)((P(x) \vee \overline{M(x)}) \& (M(x) \vee \overline{S(x)}) \& S(f(x)) \& (\overline{S(x)} \vee \overline{P(x)}))$$

д. Множество дизъюнктов формулы $\neg A$ есть множество

$$D = \{C_1 = S(f(x)), C_2 = \overline{M(x)} \vee P(x), C_3 = M(x) \vee \overline{S(x)}, C_4 = \overline{P(x)} \vee \overline{S(x)}\}.$$

е. Множество $\{a, f\}$ составит материал (множество символов) для построения эрбранова универсума. Эрбранов универсум множества дизъюнктов D есть множество

$$H = \{a, fa = f(a), ffa = f(f(a)), fffa = f(f(f(a))), \dots\}.$$

ж. Множество $\{a, f, M, P, S\}$ составит материал (множество символов) для построения эрбранова базиса.

Эрбранов базис множества дизъюнктов S есть множество

$$B = \{M(t), P(t), S(t) : t \text{ пробегает элементы эрбранова универсума } H\} = \\ \{A_1 = S(fa), A_2 = M(a), A_3 = P(a), A_4 = S(a), A_5 = S(ffa), A_6 = M(fa), \\ A_7 = P(fa), A_8 = M(ffa), A_9 = P(ffa), \dots\}.$$

з. Множество основных примеров дизъюнктов из D состоит из следующих множеств.

$\{S(f(t)) : t \text{ пробегает элементы эрбранова универсума } H\}.$

$\{\overline{M(t)} \vee P(t) : t \text{ пробегает элементы эрбранова универсума } H\}.$

$\{M(t) \vee \overline{S(t)} : t \text{ пробегает элементы эрбранова универсума } H\}.$

$\{\overline{P(t)} \vee \overline{S(t)} : t \text{ пробегает элементы эрбранова универсума } H\}.$

и. Строим замкнутое семантическое дерево для D (рис. П1.13).

Множество дизъюнктов формулы $\neg A$ есть множество

$$D = S(f(x)) \ \& \ (\overline{M(x)} \vee P(x)) \ \& \ (M(x) \vee \overline{S(x)}) \ \& \ (\overline{P(x)} \vee \overline{S(x)}).$$

Эрбранов базис множества дизъюнктов D есть множество

$$B = \{M(t), P(t), S(t) : t \text{ пробегает элементы эрбранова базиса } H\} = \\ \{A_1 = S(fa), A_2 = M(a), A_3 = P(a), A_4 = S(a), A_5 = S(ffa), A_6 = M(fa), \\ A_7 = P(fa), A_8 = M(ffa), A_9 = P(ffa), \dots\}.$$

Узел 0. $A_1 = S(fa) = 0$. Остальные A_i не определены (прочерки). Эрбранова интерпретация $I = (0, -, -, \dots)$.

$$D(a) = S(f(a)) \ \& \ (\overline{M(a)} \vee P(a)) \ \& \ (M(a) \vee \overline{S(a)}) \ \& \ (\overline{P(a)} \vee \overline{S(a)}) = \\ 0 \ \& \ (-\vee-) \ \& \ (-\vee-) \ \& \ - \ \& \ (-\vee-) = 0 \ \& \ - \ \& \ - \ \& \ - = 0.$$

Узел 0 является опровергающим. В узле 0 опровергается дизъюнкт $C_1 = S(f(x))$, ибо его основной пример $C'_1 = A_1 = S(fa) = 0$.

Узел 1. $A_1 = S(fa) = 1$. Остальные A_i не определены.

Эрбранова интерпретация $I = (1, -, -, \dots)$.

$$D(a) = S(f(a)) \ \& \ (\overline{M(a)} \vee P(a)) \ \& \ (M(a) \vee \overline{S(a)}) \ \& \ (\overline{P(a)} \vee \overline{S(a)}) = \\ 1 \ \& \ (-\vee-) \ \& \ (-\vee-) \ \& \ (-\vee-) = 1 \ \& \ - \ \& \ - \ \& \ - = -.$$

Узел 1 опровергающим не является.

Узел 10. $A_1 = S(fa) = 1, A_2 = M(a) = 0$. H -интерпретация

$I = (1, 0, -, -, \dots)$.

$$D(a) = S(f(a)) \ \& \ (\overline{M(a)} \vee P(a)) \ \& \ (M(a) \vee \overline{S(a)}) \ \& \ (\overline{P(a)} \vee \overline{S(a)}) = \\ 1 \ \& \ (1 \vee -) \ \& \ (0 \vee -) \ \& \ (-\vee-) = 1 \ \& \ 1 \ \& \ - \ \& \ - = -.$$

Узел 10 опровергающим не является.



Рис. П1.13

Узел 11. $A_1 = S(fa) = 1, A_2 = M(a) = 1$. H -интерпретация

$I = (1, 1, -, -, \dots)$.

$$D(a) = S(f(a)) \ \& \ (\overline{M(a)} \vee P(a)) \ \& \ (M(a) \vee \overline{S(a)}) \ \& \ (\overline{P(a)} \vee \overline{S(a)}) = \\ 1 \& (0 \vee -) \& (1 \vee -) \& (- \vee -) = 1 \& - \& 1 \& - = -.$$

Узел 11 опровергающим не является.

Узел 100. $A_1 = S(fa) = 1, A_2 = M(a) = 0, A_3 = P(a) = 0$.

H -интерпретация $I = (1, 0, 0, -, \dots)$.

$$D(a) = S(f(a)) \ \& \ (\overline{M(a)} \vee P(a)) \ \& \ (M(a) \vee \overline{S(a)}) \ \& \ (\overline{P(a)} \vee \overline{S(a)}) = \\ 1 \& (1 \vee 0) \& (0 \vee -) \& (1 \vee -) = 1 \& 1 \& - \& 1 = -.$$

Узел 100 опровергающим не является.

Узел 101. $A_1 = S(fa) = 1, A_2 = M(a) = 0, A_3 = P(a) = 1$.

H -интерпретация $I = (1, 0, 1, -, \dots)$.

$$D(a) = S(f(a)) \ \& \ (\overline{M(a)} \vee P(a)) \ \& \ (M(a) \vee \overline{S(a)}) \ \& \ (\overline{P(a)} \vee \overline{S(a)}) = \\ 1 \& (1 \vee 1) \& (0 \vee -) \& (0 \vee -) = 1 \& 1 \& - \& - = -.$$

Узел 101 опровергающим не является.

Узел 110. $A_1 = S(fa) = 1, A_2 = M(a) = 1, A_3 = P(a) = 0$.

H -интерпретация $I = (1, 1, 0, -, \dots)$.

$$D(a) = S(f(a)) \ \& \ (\overline{M(a)} \vee P(a)) \ \& \ (M(a) \vee \overline{S(a)}) \ \& \ (\overline{P(a)} \vee \overline{S(a)}) = \\ 1 \& (0 \vee 0) \& (1 \vee -) \& (1 \vee -) = 1 \& 0 \& 1 \& 1 = 0.$$

Узел 110 является опровергающим. В узле 110 опровергается дизъюнкт

$C_2 = \overline{M(x)} \vee P(x)$, ибо его основной пример $C'_2 = \overline{M(a)} \vee P(a) = 0$.

Узел 111. $A_1 = S(fa) = 1, A_2 = M(a) = 1, A_3 = P(a) = 1$. H -интерпретация

$I = (1, 1, 1, -, \dots)$.

$$D(a) = S(f(a)) \ \& \ (\overline{M(a)} \vee P(a)) \ \& \ (M(a) \vee \overline{S(a)}) \ \& \ (\overline{P(a)} \vee \overline{S(a)}) = \\ 1 \& (0 \vee 1) \& (1 \vee -) \& (0 \vee -) = 1 \& 1 \& 1 \& - = -.$$

Узел 111 опровергающим не является.

Узел 1000. $A_1 = S(fa) = 1, A_2 = M(a) = 0, A_3 = P(a) = 0, A_4 = S(a) = 0$.

H -интерпретация $I = (1, 0, 0, 0, -, \dots)$.

$$D(a) = S(f(a)) \ \& \ (\overline{M(a)} \vee P(a)) \ \& \ (M(a) \vee \overline{S(a)}) \ \& \ (\overline{P(a)} \vee \overline{S(a)}) = \\ 1 \& (1 \vee 0) \& (0 \vee 1) \& (1 \vee 1) = 1 \& 1 \& 1 \& 1 = 1.$$

Узел 1000 опровергающим не является.

Узел 1001. $A_1 = S(fa) = 1, A_2 = M(a) = 0, A_3 = P(a) = 0, A_4 = S(a) = 1$.

H -интерпретация $I = (1, 0, 0, 1, -, \dots)$.

$$D(a) = S(f(a)) \ \& \ (\overline{M(a)} \vee \overline{P(a)}) \ \& \ (M(a) \vee \overline{S(a)}) \ \& \ (\overline{P(a)} \vee \overline{S(a)}) = \\ 1 \& (1 \vee 0) \& (0 \vee 0) \& (1 \vee 0) = 1 \& 1 \& 0 \& 1 = 0.$$

Узел 1001 является опровергающим. В узле 1001 опровергается дизъюнкт

$$C_3 = M(x) \vee \overline{S(x)}, \text{ ибо его основной пример } C'_3 = M(a) \vee \overline{S(a)} = 0.$$

Узел 1010. $A_1 = S(fa) = 1, A_2 = M(a) = 0, A_3 = P(a) = 1, A_4 = S(a) = 0.$

H -интерпретация $I = (1, 0, 1, 0, -, \dots).$

$$D(a) = S(f(a)) \ \& \ (\overline{M(a)} \vee \overline{P(a)}) \ \& \ (M(a) \vee \overline{S(a)}) \ \& \ (\overline{P(a)} \vee \overline{S(a)}) = \\ 1 \& (1 \vee 1) \& (0 \vee 1) \& (0 \vee 1) = 1 \& 1 \& 1 \& 1 = 1.$$

Узел 1010 опровергающим не является.

Узел 1011. $A_1 = S(fa) = 1, A_2 = M(a) = 0, A_3 = P(a) = 1, A_4 = S(a) = 1.$

H -интерпретация $I = (1, 0, 1, 1, -, \dots).$

$$D(a) = S(f(a)) \ \& \ (\overline{M(a)} \vee \overline{P(a)}) \ \& \ (M(a) \vee \overline{S(a)}) \ \& \ (\overline{P(a)} \vee \overline{S(a)}) = \\ 1 \& (1 \vee 1) \& (0 \vee 0) \& (0 \vee 0) = 1 \& 1 \& 0 \& 0 = 0.$$

Узел 1011 является опровергающим. В узле 1011 опровергается дизъюнкт

$$C_4 = \overline{P(x)} \vee \overline{S(x)}, \text{ ибо его основной пример } C'_4 = \overline{P(a)} \vee \overline{S(a)} = 0.$$

Узел 1110. $A_1 = S(fa) = 1, A_2 = M(a) = 1, A_3 = P(a) = 1, A_4 = S(a) = 0.$

H -интерпретация $I = (1, 1, 1, 0, -, \dots).$

$$D(a) = S(f(a)) \ \& \ (\overline{M(a)} \vee \overline{P(a)}) \ \& \ (M(a) \vee \overline{S(a)}) \ \& \ (\overline{P(a)} \vee \overline{S(a)}) = \\ 1 \& (0 \vee 1) \& (1 \vee 1) \& (0 \vee 1) = 1 \& 1 \& 1 \& 1 = 1.$$

Узел 1110 опровергающим не является.

Узел 1111. $A_1 = S(fa) = 1, A_2 = M(a) = 1, A_3 = P(a) = 1, A_4 = S(a) = 1.$

H -интерпретация $I = (1, 1, 1, 1, -, \dots).$

$$D(a) = S(f(a)) \ \& \ (\overline{M(a)} \vee \overline{P(a)}) \ \& \ (M(a) \vee \overline{S(a)}) \ \& \ (\overline{P(a)} \vee \overline{S(a)}) = \\ 1 \& (0 \vee 1) \& (1 \vee 0) \& (0 \vee 0) = 1 \& 1 \& 1 \& 0 = 0.$$

Узел 1111 является опровергающим. В узле 1111 опровергается дизъюнкт

$$C_4 = \overline{P(x)} \vee \overline{S(x)}, \text{ ибо его основной пример } C'_4 = \overline{P(a)} \vee \overline{S(a)} = 0.$$

Узел 10000. $A_1 = S(fa) = 1, A_2 = M(a) = 0, A_3 = P(a) = 0, A_4 = S(a) = 0,$

$A_5 = S(ffa) = 0.$ H -интерпретация $I = (1, 0, 0, 0, 0, -, \dots).$

$$D(fa) = S(ffa) \ \& \ (\overline{M(fa)} \vee \overline{P(fa)}) \ \& \ (M(fa) \vee \overline{S(fa)}) \ \& \ (\overline{P(fa)} \vee \overline{S(fa)}) = \\ 0 \& (- \vee -) \& (- \vee 0) \& (- \vee 0) = 0 \& - \& - \& - = 0.$$

Узел 10000 является опровергающим. В узле 10000 опровергается дизъюнкт

$$C_1 = S(f(x)), \text{ ибо его основной пример } C'_1 = A_5 = S(ffa) = 0.$$

Узел 10001. $A_1 = S(fa) = 1, A_2 = M(a) = 0, A_3 = P(a) = 0, A_4 = S(a) = 0, A_5 = S(ffa) = 1$. H -интерпретация $I = (1, 0, 0, 0, 1, -, \dots)$.

$$D(fa) = S(ffa) \& (\overline{M(fa)} \vee P(fa)) \& (M(fa) \vee \overline{S(fa)}) \& (\overline{P(fa)} \vee \overline{S(fa)}) = 1 \& (-\vee -) \& (-\vee 0) \& (-\vee 0) = 1 \& - \& - \& - = -.$$

Узел 10001 опровергающим не является.

Узел 10100. $A_1 = S(fa) = 1, A_2 = M(a) = 0, A_3 = P(a) = 1, A_4 = S(a) = 0, A_5 = S(ffa) = 0$. H -интерпретация $I = (1, 0, 1, 0, 0, -, \dots)$.

$$D(fa) = S(fa) \& (\overline{M(a)} \vee P(a)) \& (M(a) \vee \overline{S(a)}) \& (\overline{P(a)} \vee \overline{S(a)}) = 0 \& (-\vee -) \& (-\vee 0) \& (-\vee 0) = 0 \& - \& - \& - = 0.$$

Узел 10100 является опровергающим. В узле 10100 опровергается дизъюнкт $C_1 = S(f(x))$, ибо его основной пример $C'_1 = A_5 = S(ffa) = 0$.

Узел 10101. $A_1 = S(fa) = 1, A_2 = M(a) = 0, A_3 = P(a) = 1, A_4 = S(a) = 0, A_5 = S(ffa) = 1$. H -интерпретация $I = (1, 0, 1, 0, 1, -, \dots)$.

$$D(fa) = S(ffa) \& (\overline{M(fa)} \vee P(fa)) \& (M(fa) \vee \overline{S(fa)}) \& (\overline{P(fa)} \vee \overline{S(fa)}) = 1 \& (-\vee -) \& (-\vee 0) \& (-\vee 0) = 1 \& - \& - \& - = -.$$

Узел 10101 опровергающим не является.

Узел 11100. $A_1 = S(fa) = 1, A_2 = M(a) = 1, A_3 = P(a) = 1, A_4 = S(a) = 0, A_5 = S(ffa) = 0$. H -интерпретация $I = (1, 1, 1, 0, 0, -, \dots)$.

$$D(fa) = S(ffa) \& (\overline{M(fa)} \vee P(fa)) \& (M(fa) \vee \overline{S(fa)}) \& (\overline{P(fa)} \vee \overline{S(fa)}) = 0 \& (-\vee -) \& (-\vee 0) \& (-\vee 0) = 0 \& - \& - \& - = 0.$$

Узел 11100 является опровергающим. В узле 11100 опровергается дизъюнкт $C_1 = S(f(x))$, ибо его основной пример $C'_1 = A_5 = S(ffa) = 0$.

Узел 11101. $A_1 = S(fa) = 1, A_2 = M(a) = 1, A_3 = P(a) = 1, A_4 = S(a) = 0, A_5 = S(ffa) = 1$. H -интерпретация $I = (1, 1, 1, 0, 1, -, \dots)$.

$$D(fa) = S(ffa) \& (\overline{M(fa)} \vee P(fa)) \& (M(fa) \vee \overline{S(fa)}) \& (\overline{P(fa)} \vee \overline{S(fa)}) = 1 \& (-\vee -) \& (-\vee 0) \& (-\vee 0) = 1 \& - \& - \& - = -.$$

Узел 11101 опровергающим не является.

Узел 100010. $A_1 = S(fa) = 1, A_2 = M(a) = 0, A_3 = P(a) = 0, A_4 = S(a) = 0, A_5 = S(ffa) = 1, A_6 = M(fa) = 0$. H -интерпретация $I = (1, 0, 0, 0, 1, 0, -, \dots)$.

$$D(fa) = S(ffa) \& (\overline{M(fa)} \vee P(fa)) \& (M(fa) \vee \overline{S(fa)}) \& (\overline{P(fa)} \vee \overline{S(fa)}) = 1 \& (1 \vee -) \& (0 \vee 0) \& (-\vee 0) = 1 \& 1 \& 0 \& - = 0.$$

Узел 100010 является опровергающим. В узле 100010 опровергается дизъюнкт $C_3 = M(x) \vee \overline{S(x)}$, ибо его основной пример $C'_3 = M(fa) \vee \overline{S(fa)} = 0$.

Узел 100011. $A_1 = S(fa) = 1, A_2 = M(a) = 0, A_3 = P(a) = 0, A_4 = S(a) = 0,$
 $A_5 = S(ffa) = 1, A_6 = M(fa) = 1.$ H -интерпретация $I = (1, 0, 0, 0, 1, 1, -, \dots).$

$$D(fa) = S(ffa) \ \& \ (\overline{M(fa)} \vee P(fa)) \ \& \ (M(fa) \vee \overline{S(fa)}) \ \& \ (\overline{P(fa)} \vee \overline{S(fa)}) =$$

$$1 \ \& \ (0 \vee -) \ \& \ (1 \vee 0) \ \& \ (- \vee 0) = 1 \ \& \ - \ \& \ 1 \ \& \ - = -.$$

Узел 100011 опровергающим не является.

Узел 101010. $A_1 = S(fa) = 1, A_2 = M(a) = 0, A_3 = P(a) = 1, A_4 = S(a) = 0,$
 $A_5 = S(ffa) = 1, A_6 = M(fa) = 0.$ H -интерпретация $I = (1, 0, 1, 0, 1, 0, -, \dots).$

$$D(fa) = S(ffa) \ \& \ (\overline{M(fa)} \vee P(fa)) \ \& \ (M(fa) \vee \overline{S(fa)}) \ \& \ (\overline{P(fa)} \vee \overline{S(fa)}) =$$

$$1 \ \& \ (1 \vee -) \ \& \ (0 \vee 0) \ \& \ (- \vee 0) = 1 \ \& \ 1 \ \& \ 0 \ \& \ - = 0.$$

Узел 101010 является опровергающим. В узле 101010 опровергается дизъюнкт $C_3 = M(x) \vee \overline{S(x)}$, ибо его основной пример $C'_3 = M(fa) \vee \overline{S(fa)} = 0.$

Узел 101011. $A_1 = S(fa) = 1, A_2 = M(a) = 0, A_3 = P(a) = 1, A_4 = S(a) = 0,$
 $A_5 = S(ffa) = 1, A_6 = M(fa) = 1.$ H -интерпретация $I = (1, 0, 1, 0, 1, 1, -, \dots).$

$$D(fa) = S(ffa) \ \& \ (\overline{M(fa)} \vee P(fa)) \ \& \ (M(fa) \vee \overline{S(fa)}) \ \& \ (\overline{P(fa)} \vee \overline{S(fa)}) =$$

$$1 \ \& \ (0 \vee -) \ \& \ (1 \vee 0) \ \& \ (- \vee 0) = 1 \ \& \ - \ \& \ 1 \ \& \ - = -.$$

Узел 101011 опровергающим не является.

Узел 111010. $A_1 = S(fa) = 1, A_2 = M(a) = 1, A_3 = P(a) = 1, A_4 = S(a) = 0,$
 $A_5 = S(ffa) = 1, A_6 = M(fa) = 0.$ H -интерпретация $I = (1, 1, 1, 0, 1, 0, -, \dots).$

$$D(fa) = S(ffa) \ \& \ (\overline{M(fa)} \vee P(fa)) \ \& \ (M(fa) \vee \overline{S(fa)}) \ \& \ (\overline{P(fa)} \vee \overline{S(fa)}) =$$

$$1 \ \& \ (1 \vee -) \ \& \ (0 \vee 0) \ \& \ (- \vee 0) = 1 \ \& \ 1 \ \& \ 0 \ \& \ - = 0.$$

Узел 111010 является опровергающим. В узле 111010 опровергается дизъюнкт $C_3 = M(x) \vee \overline{S(x)}$, ибо его основной пример $C'_3 = M(fa) \vee \overline{S(fa)} = 0.$

Узел 111011. $A_1 = S(fa) = 1, A_2 = M(a) = 1, A_3 = P(a) = 1, A_4 = S(a) = 0,$
 $A_5 = S(ffa) = 1, A_6 = M(fa) = 1.$ H -интерпретация $I = (1, 1, 1, 0, 1, 1, -, \dots).$

$$D(fa) = S(ffa) \ \& \ (\overline{M(fa)} \vee P(fa)) \ \& \ (M(fa) \vee \overline{S(fa)}) \ \& \ (\overline{P(fa)} \vee \overline{S(fa)}) =$$

$$1 \ \& \ (0 \vee -) \ \& \ (1 \vee 0) \ \& \ (- \vee 0) = 1 \ \& \ - \ \& \ 1 \ \& \ - = -.$$

Узел 111011 опровергающим не является.

Узел 1000110. $A_1 = S(fa) = 1, A_2 = M(a) = 0, A_3 = P(a) = 0, A_4 = S(a) = 0,$
 $A_5 = S(ffa) = 1, A_6 = M(fa) = 1, A_7 = P(fa) = 0.$

H -интерпретация $I = (1, 0, 0, 0, 1, 1, 0, -, \dots).$

$$D(fa) = S(ffa) \ \& \ (\overline{M(fa)} \vee P(fa)) \ \& \ (M(fa) \vee \overline{S(fa)}) \ \& \ (\overline{P(fa)} \vee \overline{S(fa)}) =$$

$$1 \ \& \ (0 \vee 0) \ \& \ (1 \vee 0) \ \& \ (1 \vee 0) = 1 \ \& \ 0 \ \& \ 1 \ \& \ 1 = 0.$$

Узел 1000110 является опровергающим. В узле 1000110 опровергается дизъюнкт $C_2 = \overline{M(x)} \vee P(x)$, ибо его основной пример $C'_2 = \overline{M(fa)} \vee P(fa) = 0.$

Узел 1000111. $A_1 = S(fa) = 1, A_2 = M(a) = 0, A_3 = P(a) = 0, A_4 = S(a) = 0,$
 $A_5 = S(ffa) = 1, A_6 = M(fa) = 1, A_7 = P(fa) = 1.$

H -интерпретация $I = (1, 0, 0, 0, 1, 1, -, \dots).$

$$D(fa) = S(ffa) \ \& \ (\overline{M(fa)} \vee P(fa)) \ \& \ (M(fa) \vee \overline{S(fa)}) \ \& \ (\overline{P(fa)} \vee \overline{S(fa)}) = 1 \& (0 \vee 1) \& (1 \vee 0) \& (0 \vee 0) = 1 \& 1 \& 1 \& 0 = 0.$$

В узле 1000111 опровергается дизъюнкт $C_4 = \overline{P(x)} \vee \overline{S(x)}$, ибо его основной пример $C'_4 = \overline{P(fa)} \vee \overline{S(fa)} = 0.$

Узел 1010110. $A_1 = S(fa) = 1, A_2 = M(a) = 0, A_3 = P(a) = 1, A_4 = S(a) = 0,$
 $A_5 = S(ffa) = 1, A_6 = M(fa) = 1, A_7 = P(fa) = 0.$

H -интерпретация $I = (1, 0, 1, 0, 1, 1, 0, -, \dots).$

$$D(fa) = S(ffa) \ \& \ (\overline{M(fa)} \vee P(fa)) \ \& \ (M(fa) \vee \overline{S(fa)}) \ \& \ (\overline{P(fa)} \vee \overline{S(fa)}) = 1 \& (0 \vee 0) \& (1 \vee 0) \& (1 \vee 0) = 1 \& 0 \& 1 \& 1 = 0.$$

В узле 1010110 опровергается дизъюнкт $C_2 = \overline{M(x)} \vee P(x)$, ибо его основной пример $C'_2 = \overline{M(fa)} \vee P(fa) = 0.$

Узел 1010111. $A_1 = S(fa) = 1, A_2 = M(a) = 0, A_3 = P(a) = 1, A_4 = S(a) = 0,$
 $A_5 = S(ffa) = 1, A_6 = M(fa) = 1, A_7 = P(fa) = 1.$

H -интерпретация $I = (1, 0, 1, 0, 1, 1, -, \dots).$

$$D(fa) = S(ffa) \ \& \ (\overline{M(fa)} \vee P(fa)) \ \& \ (M(fa) \vee \overline{S(fa)}) \ \& \ (\overline{P(fa)} \vee \overline{S(fa)}) = 1 \& (0 \vee 1) \& (1 \vee 0) \& (0 \vee 0) = 1 \& 1 \& 1 \& 0 = 0.$$

В узле 1010111 опровергается дизъюнкт $C_4 = \overline{P(x)} \vee \overline{S(x)}$, ибо его основной пример $C'_4 = \overline{P(fa)} \vee \overline{S(fa)} = 0.$

Узел 1110110. $A_1 = S(fa) = 1, A_2 = M(a) = 1, A_3 = P(a) = 1, A_4 = S(a) = 0,$
 $A_5 = S(ffa) = 1, A_6 = M(fa) = 1, A_7 = P(fa) = 0.$

H -интерпретация $I = (1, 1, 1, 0, 1, 1, -, \dots).$

$$D(fa) = S(ffa) \ \& \ (\overline{M(fa)} \vee P(fa)) \ \& \ (M(fa) \vee \overline{S(fa)}) \ \& \ (\overline{P(fa)} \vee \overline{S(fa)}) = 1 \& (0 \vee 0) \& (1 \vee 0) \& (1 \vee 0) = 1 \& 0 \& 1 \& 1 = 0.$$

В узле 1110110 опровергается дизъюнкт $C_2 = \overline{M(x)} \vee P(x)$, ибо его основной пример $C'_2 = \overline{M(fa)} \vee P(fa) = 0.$

Узел 1110111. $A_1 = S(fa) = 1, A_2 = M(a) = 1, A_3 = P(a) = 1, A_4 = S(a) = 0,$
 $A_5 = S(ffa) = 1, A_6 = M(fa) = 1, A_7 = P(fa) = 1.$

H -интерпретация $I = (1, 1, 1, 0, 1, 1, -, \dots).$

$$D(fa) = S(ffa) \ \& \ (\overline{M(fa)} \vee P(fa)) \ \& \ (M(fa) \vee \overline{S(fa)}) \ \& \ (\overline{P(fa)} \vee \overline{S(fa)}) = 1 \& (0 \vee 1) \& (1 \vee 1) \& (0 \vee 0) = 1 \& 1 \& 1 \& 0 = 0.$$

В узле 1010111 опровергается дизъюнкт $C_4 = \overline{P(x)} \vee \overline{S(x)}$, ибо его основной пример $C'_4 = \overline{P(fa)} \vee \overline{S(fa)} = 0$.

Все концевые узлы построенного замкнутого семантического дерева опровергающие. Множество дизъюнктов S опровержимо на каждой H -интерпретации.

Замечание. Следующие утверждения эквивалентны.

1. Множество дизъюнктов D невыполнимо (на всех H -интерпретациях).
2. Множество дизъюнктов D невыполнимо на всех интерпретациях.
3. Конъюнкция дизъюнктов D невыполнима.
4. Стандартная форма Скулема формулы $\neg A$ невыполнима.
5. Формула $\neg A$ невыполнима.
6. Формула A общезначима.
7. Правило вывода, описываемое формулой A , верно.

Отсюда сразу следует верность исходного правила вывода.

к. (Конечное) множество основных примеров, опровергающих каждую H -интерпретацию множества дизъюнктов D , выписывается из сведений в опровергающих узлах обрезанного семантического дерева для множества дизъюнктов D .

Множество опровергающих узлов обрезанного семантического дерева для множества дизъюнктов D есть множество

$$U = \{0, 110, 1001, 1011, 1111, \\ 10000, 10100, 11100, \\ 100010, 101010, 111010, \\ 1000110, 1000111, 1010110, 1010111, 1110110, 1110111\}.$$

Множеству узлов U соответствует выписанное (без повторов) по узлам соответствующее конечное множество основных примеров BE , опровергающих каждую H -интерпретацию множества дизъюнктов D .

В узле 0 опровергается дизъюнкт $C_1 = S(f(x))$, ибо его основной пример $C_1 = A_1 = S(fa) = 0$.

В узле 110 опровергается дизъюнкт $C_2 = \overline{M(x)} \vee P(x)$, ибо его основной пример $C'_2 = \overline{M(a)} \vee P(a) = 0$.

В узле 1001 опровергается дизъюнкт $C_3 = M(x) \vee \overline{S(x)}$, ибо его основной пример $C'_3 = M(a) \vee \overline{S(a)} = 0$.

В узле 1011 опровергается дизъюнкт $C_4 = \overline{P(x)} \vee \overline{S(x)}$, ибо его основной пример $C'_4 = \overline{P(a)} \vee \overline{S(a)} = 0$.

В узле 1111 опровергается дизъюнкт $C_4 = \overline{P(x)} \vee \overline{S(x)}$, ибо его основной пример $C'_4 = \overline{P(a)} \vee \overline{S(a)} = 0$.

В узле 10000 опровергается дизъюнкт $C_1 = S(f(x))$, ибо его основной пример $C'_1 = A_5 = S(fa) = 0$.

В узле 10100 опровергается дизъюнкт $C_1 = S(f(x))$, ибо его основной пример $C'_1 = A_5 = S(ffa) = 0$.

В узле 11100 опровергается дизъюнкт $C_1 = S(f(x))$, ибо его основной пример $C'_1 = A_5 = S(ffa) = 0$.

В узле 100010 опровергается дизъюнкт $C_3 = M(x) \vee \overline{S(x)}$, ибо его основной пример $C'_3 = M(fa) \vee \overline{S(fa)} = 0$.

В узле 101010 опровергается дизъюнкт $C_3 = M(x) \vee \overline{S(x)}$, ибо его основной пример $C'_3 = M(fa) \vee \overline{S(fa)} = 0$.

В узле 111010 опровергается дизъюнкт $C_2 = \overline{M(x)} \vee P(x)$, ибо его основной пример $C'_2 = \overline{M(fa)} \vee P(fa) = 0$.

В узле 1000110 опровергается дизъюнкт $C_2 = \overline{M(x)} \vee P(x)$, ибо его основной пример $C'_2 = \overline{M(fa)} \vee P(fa) = 0$.

В узле 1000111 опровергается дизъюнкт $C_4 = \overline{P(x)} \vee \overline{S(x)}$, ибо его основной пример $C'_4 = \overline{P(fa)} \vee \overline{S(fa)} = 0$.

В узле 1010110 опровергается дизъюнкт $C_2 = \overline{M(x)} \vee P(x)$, ибо его основной пример $C'_2 = \overline{M(fa)} \vee P(fa) = 0$.

В узле 1010111 опровергается дизъюнкт $C_4 = \overline{P(x)} \vee \overline{S(x)}$, ибо его основной пример $C'_4 = \overline{P(fa)} \vee \overline{S(fa)} = 0$.

В узле 1110110 опровергается дизъюнкт $C_2 = \overline{M(x)} \vee P(x)$, ибо его основной пример $C'_2 = \overline{M(fa)} \vee P(fa) = 0$.

В узле 1010111 опровергается дизъюнкт $C_4 = \overline{P(x)} \vee \overline{S(x)}$, ибо его основной пример $C'_4 = \overline{P(fa)} \vee \overline{S(fa)} = 0$.

Опроверяющее множество основных примеров $BE =$

$\{S(fa), \overline{M(a)} \vee P(a), M(a) \vee \overline{S(a)}, \overline{P(a)} \vee \overline{S(a)}, S(ffa), M(fa) \vee \overline{S(fa)}, \overline{M(fa)} \vee P(fa), \overline{P(fa)} \vee \overline{S(fa)}\}$.

Замечание. Следующие утверждения эквивалентны.

1. Множество дизъюнктов D невыполнимо.
2. Множество дизъюнктов D опровержимо на каждой H -интерпретации.
3. Замкнутое семантическое дерево T для D конечно (имеет конечное число узлов). Все концевые узлы в T опровергающие.
4. Множество BE основных примеров, опровергающих дизъюнкты из D в концевых узлах дерева T , невыполнимо на всякой H -интерпретации.
5. Множество BE основных примеров, опровергающих дизъюнкты из D в концевых узлах дерева T , невыполнимо на всякой интерпретации.

Отсюда сразу следует верность правила вывода.

к. Покажем, что $D \vdash \square$.

(1) $S(f(x))$, условие,

(2) $\overline{M(x)} \vee P(x)$, условие,

(3) $M(x) \vee \overline{S(x)}$, условие,

(4) $\overline{P(x)} \vee \overline{S(x)}$, условие,

(5) $P(x) \vee \overline{S(x)}$, ПР(2,3),

(6) $\overline{S(f(x))}S(f(x))$, ПР(4,5), подстановка $\theta = \{f(x) \mid x\}$,

(7) $\square$, ПР(1,6).

Так как $D \vdash \square$, то $D \equiv 0$ и потому исходное правило вывода верно.

Задача 31. Написать протокол работы Пролог-программ для предикатов

`member(X,Y)`, `first(X,Y)`, `last(X,Y)`, `append(X,Y,Z)`, `reverse(X,Y)`, `add(X,Y)`, `delete(X,Y,Z)`,
`delall(X,Y,Z)`, `substitute(X,Y,Z)`, `sublist([X|L],[X|M])`, `subset(X,Y)`, `unionset(X,Y,Z)`,
`intersect(X,Y,Z)`, `difset(X,Y,Z)`, `go(S,G,T)`,

заданных следующими программами. (Исходные множества 1, 2, 3, ... получить как три первые буквы: 1) вашей фамилии, имени, отчества; 2) фамилии, имени, отчества ваших родственников; 3) и так далее.

```
member(X, [X|Y]).
member(X, [_|Z]) :- member(X,Z).

first(X, [X|Y]).

last(X, [X]).
last(X, [_|Y]) :- last(X,Y).

append([], L, L).
append([X|L1], L2, [X|L3]) :- append(L1, L2, L3).

reverse([], []).
reverse([H|T], L) :- reverse(T, Z), append(Z, [H], L).
reverse1(L1, L2) :- rev(L1, [], L2).

rev([], L, L).
rev([X|L], L2, L3) :- rev(L, [X|L2], L3).

add(X, L, [X|L]).

delete(A, [A|B], B) :- !.
delete(A, [_|B], [B|M]) :- delete(A, L, M).

delall(_, [], []).
delall(X, [X|L], M) :- !, delall(X, L, M).
delall(X, [_|L1], [Y|L2]) :- delall(X, L1, L2).

substitute(_, [], [], []).
substitute(X, [X|L], A, [A|M]) :- !, substitute(X, L, A, M).
substitute(X, [_|L], A, [Y|M]) :- substitute(X, L, A, M).
```

```

sublist([X|L],[X|M]) :- coincide(L,M),!.
sublist(L,[_|M]) :- sublist(L,M).

coincide([],_).
coincide([X|L],[X|M]) :- coincide(L,M).

subset([],Y).
subset([A|X],Y) :- member(A,Y),subset(X,Y).

unionset([X|R],Y,Z) :- member(X,Y),!,unionset(R,Y,Z).
unionset([X|R],Y,[X|Z]) :- unionset(R,Y,Z).
unionset([],X,X).

intersect([],X,[]).
intersect([X|R],Y,[X|Z]) :- member(X,Y),!,intersect(R,Y,Z).
intersect([X|R],Y,Z) :- intersect(R,Y,Z).

difset(X,Y,T) :- dif1(X,Y,X,T).
dif1([R|X],Y,[R|Z],T) :- not(member(R,Y)),
    append(Z,[R],Z1),dif1(X,Y,Z1,T).
dif1([R|X],Y,[R|Z],T) :- member(R,Y),dif1(X,Y,Z,T).
dif1([],Y,Z,Z).

go(S,G,T) :- go1(S,G,[],T).
a(n,k). a(k,p). a(d,n). a(p,d). a(w,k). a(w,p).
go1(S,S,Tr,T) :- T=[S|Tr].
go1(S,N,Tr,T) :-
nextnode(N,Tr,N1),go1(S,N1,[N|Tr],T).
nextnode(N,Tr,N1) :-
    (a(N,N1) ; a(N1,N)),not(member(N1,Tr).

member(X,[X|Y]).
member(X,[Y|Z]) :- member(X,Z).

```



Приложение 2

Конечные автоматы

Задача 1. Построить по автомату Мили $A = (X, Y, Q, q_1, T, B)$ (рис. П2.1) эквивалентный ему автомат Мура. Начальное состояние q_1 . Множества входных и выходных символов $X = \{0, 1, 2\}$, $Y = \{a, b, c\}$. Вариант автомата получить, взяв указанный в варианте переход из одного состояния в другое при поступлении на вход автомата указанных в варианте входного и выходного символов. Например, в варианте 30 указан переход $(q_2, 0a, q_2)$. Это значит, что из граф-схемы автомата A надо удалить стрелку $(q_2, 0a, q_4)$ и добавить стрелку $(q_2, 0a, q_2)$.

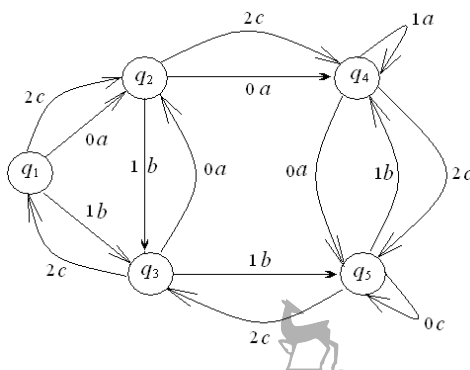


Рис. П2.1

1.1. $(q_1, 2c, q_1)$.	1.2. $(q_1, 2c, q_3)$.	1.3. $(q_1, 2c, q_4)$.	1.4. $(q_1, 2c, q_5)$.
1.5. $(q_2, 2c, q_1)$.	1.6. $(q_2, 2c, q_2)$.	1.7. $(q_2, 2c, q_3)$.	1.8. $(q_2, 2c, q_5)$.
1.9. $(q_3, 2c, q_2)$.	1.10. $(q_3, 2c, q_3)$.	1.11. $(q_3, 2c, q_4)$.	1.12. $(q_3, 2c, q_5)$.
1.13. $(q_4, 2c, q_1)$.	1.14. $(q_4, 2c, q_2)$.	1.15. $(q_4, 2c, q_3)$.	1.16. $(q_4, 2c, q_4)$.
1.17. $(q_5, 2c, q_1)$.	1.18. $(q_5, 2c, q_2)$.	1.19. $(q_5, 2c, q_4)$.	1.20. $(q_5, 2c, q_5)$.
1.21. $(q_1, 0a, q_1)$.	1.22. $(q_1, 0a, q_3)$.	1.23. $(q_1, 0a, q_4)$.	1.24. $(q_1, 0a, q_5)$.
1.25. $(q_1, 1b, q_1)$.	1.26. $(q_1, 1b, q_2)$.	1.27. $(q_1, 1b, q_4)$.	1.28. $(q_1, 1b, q_5)$.
1.29. $(q_2, 0a, q_1)$.	1.30. $(q_2, 0a, q_2)$.	1.31. $(q_2, 0a, q_3)$.	1.32. $(q_2, 0a, q_5)$.

Задача 2. Построить автоматы $A = (X, Q, q_1, T, F)$ для объединения и для пересечения двух языков, представимых детерминированными автоматами

$A' = (X, Q', q'_1, T', F')$, $A'' = (X, Q'', q''_1, T'', F'')$ (рис. П2.2) со множеством входных символов $X = \{0, 1, 2\}$, с начальными состояниями q'_1 и q''_1 и с выделенными состояниями $F' = \{q'_3, q'_5\}$, $F'' = \{q''_3\}$. Каждый вариант автомата получить, взяв указанный в варианте переход из одного состояния в другое при поступлении на вход автомата указанного в варианте входного символа. Например, в варианте 30 указан переход $(q''_3, 1, q''_2)$. Это значит, что из граф-схемы автомата A'' надо убрать стрелку $(q''_3, 1, q''_3)$ и добавить стрелку $(q''_3, 1, q''_2)$.

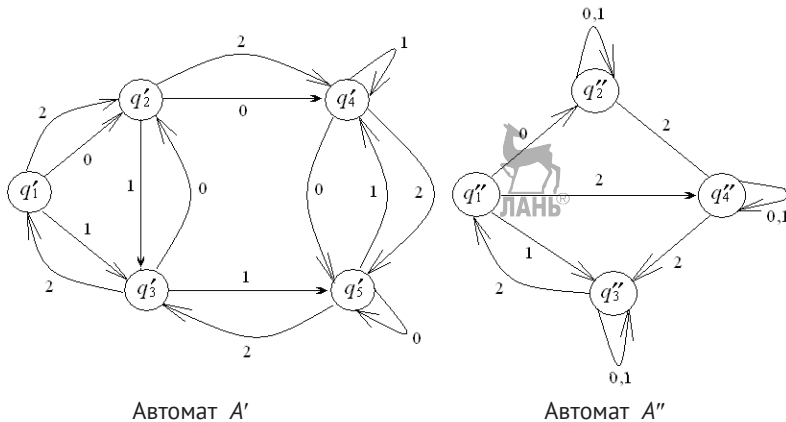


Рис. П2.2

2.1. $(q'_1, 2, q'_2)$.	2.2. $(q'_1, 2, q'_3)$.	2.3. $(q'_1, 2, q'_4)$.	2.4. $(q'_1, 2, q'_5)$.
2.5. $(q'_2, 2, q'_3)$.	2.6. $(q'_2, 2, q'_4)$.	2.7. $(q'_2, 2, q'_5)$.	2.8. $(q'_3, 2, q'_4)$.
2.9. $(q'_3, 2, q'_5)$.	2.10. $(q'_4, 2, q'_5)$.	2.11. $(q''_1, 2, q''_2)$.	2.12. $(q''_1, 2, q''_3)$.
2.13. $(q''_1, 2, q''_4)$.	2.14. $(q''_2, 2, q''_3)$.	2.15. $(q''_2, 2, q''_4)$.	2.16. $(q''_3, 2, q''_4)$.
2.17. $(q'_2, 2, q'_2)$.	2.18. $(q'_3, 2, q'_1)$.	2.19. $(q'_4, 2, q'_1)$.	2.20. $(q'_5, 2, q'_1)$.
2.21. $(q'_3, 2, q'_2)$.	2.22. $(q'_4, 2, q'_2)$.	2.23. $(q'_5, 2, q'_2)$.	2.24. $(q'_4, 2, q'_3)$.
2.25. $(q'_5, 2, q'_3)$.	2.26. $(q'_5, 2, q'_4)$.	2.27. $(q''_2, 2, q''_1)$.	2.28. $(q''_3, 2, q''_1)$.
2.29. $(q''_4, 2, q''_1)$.	2.30. $(q''_3, 1, q''_2)$.		

Задача 3. Построить автомат для дополнения языка, представленного детерминированным автоматом. Вариант автомата взять из задачи 2.

Задача 4. Детерминизировать источник, граф-схема которого изображена на рис. П2.3. Множество входных символов $X = \{0, 1, 2\}$. Множество начальных состояний $Q_0 = \{q_1, q_2\}$. Множество выделенных состояний $F = \{q_3, q_5\}$. Вариант источника получить, добавив к граф-схеме на рис. П2.3 стрелку варианта. Например, для варианта 30 к граф-схеме источника надо добавить стрелку $(q_3, 2, q_2)$.

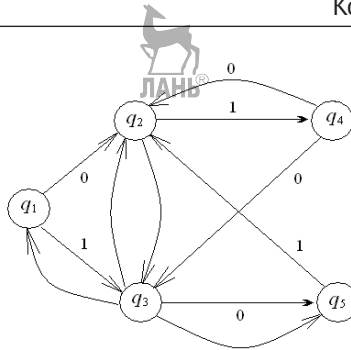


Рис. П2.3

4.1. $(q_1, 2, q_2)$.	4.2. $(q_1, 2, q_3)$.	4.3. $(q_1, 2, q_4)$.	4.4. $(q_1, 2, q_5)$.
4.5. $(q_2, 2, q_3)$.	4.6. $(q_2, 2, q_4)$.	4.7. $(q_2, 2, q_5)$.	4.8. $(q_3, 2, q_4)$.
4.9. $(q_3, 2, q_5)$.	4.10. $(q_4, 2, q_5)$.	4.11. $(q_1, 1, q_2)$.	4.12. $(q_1, 0, q_3)$.
4.13. $(q_1, 1, q_4)$.	4.14. $(q_2, 2, q_3)$.	4.15. $(q_2, 2, q_4)$.	4.16. $(q_3, 2, q_4)$.
4.17. $(q_2, 1, q_1)$.	4.18. $(q_3, 2, q_1)$.	4.19. $(q_4, 2, q_1)$.	4.20. $(q_5, 2, q_1)$.
4.21. $(q_3, 2, q_2)$.	4.22. $(q_4, 2, q_2)$.	4.23. $(q_5, 2, q_2)$.	4.24. $(q_4, 2, q_3)$.
4.25. $(q_5, 2, q_3)$.	4.26. $(q_5, 2, q_4)$.	4.27. $(q_2, 2, q_1)$.	4.28. $(q_3, 0, q_1)$.
4.29. $(q_4, 2, q_1)$.	4.30. $(q_3, 2, q_2)$.		

Задача 5. Найти источник $S = (X, Q, Q_0, D, F)$ для объединения языков, представимых источниками $S' = (X, Q', Q'_0, D', F')$, $S'' = (X, Q'', Q''_0, D'', F'')$ (рис. П2.4). Множество входных символов $X = \{0, 1, 2\}$. Множество начальных состояний $Q'_0 = \{q'_1\}$, $Q''_0 = \{q''_1, q''_2\}$. Множество выделенных состояний $F' = \{q'_3\}$, $F'' = \{q''_3, q''_5\}$. Детерминизировать полученный источник. Вариант источника получить, добавив к граф-схеме источника S'' на рис. П2.4 стрелку варианта. Например, для варианта 30 к граф-схеме источника S'' надо добавить стрелку $(q_3, 2, q_2)$.

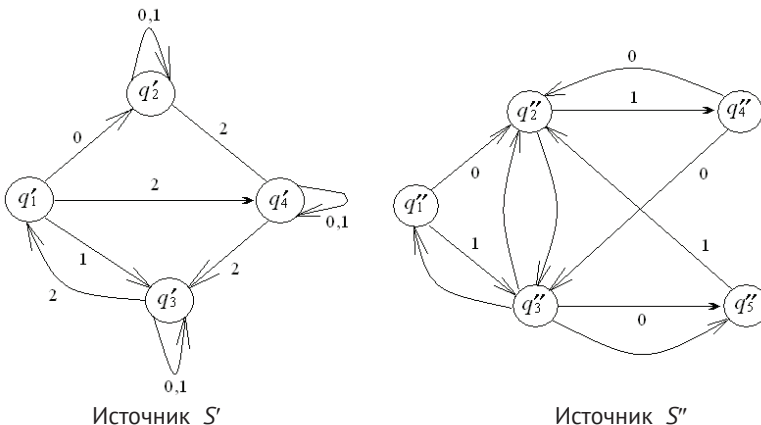


Рис. П2.4

- 5.1. $(q_1'', 2, q_2'')$. 5.2. $(q_1'', 2, q_3'')$. 5.3. $(q_1'', 2, q_4'')$. 5.4. $(q_1'', 2, q_5'')$.
 5.5. $(q_2'', 2, q_3'')$. 5.6. $(q_2'', 2, q_4'')$. 5.7. $(q_2'', 2, q_5'')$. 5.8. $(q_3'', 2, q_4'')$.
 5.9. $(q_3'', 2, q_5'')$. 5.10. $(q_4'', 2, q_5'')$. 5.11. $(q_1'', 1, q_2'')$. 5.12. $(q_1'', 0, q_3'')$.
 5.13. $(q_1'', 1, q_4'')$. 5.14. $(q_2'', 2, q_3'')$. 5.15. $(q_2'', 2, q_4'')$. 5.16. $(q_3'', 2, q_4'')$.
 5.17. $(q_2'', 1, q_1'')$. 5.18. $(q_3'', 2, q_1'')$. 5.19. $(q_4'', 2, q_1'')$. 5.20. $(q_5'', 2, q_1'')$.
 5.21. $(q_3'', 2, q_2'')$. 5.22. $(q_4'', 2, q_2'')$. 5.23. $(q_5'', 2, q_2'')$. 5.24. $(q_4'', 2, q_3'')$.
 5.25. $(q_5'', 2, q_3'')$. 5.26. $(q_5'', 2, q_4'')$. 5.27. $(q_2'', 2, q_1'')$. 5.28. $(q_3'', 0, q_1'')$.
 5.29. $(q_4'', 2, q_1'')$. 5.30. $(q_3'', 2, q_2'')$.

Задача 6. Найти источник для конкатенации языков, представимых источниками из задачи 5. Детерминизировать полученный источник.

Задача 7. Найти источник для итерации языка, представимого источником из задачи 4. Детерминизировать полученный источник.

Задача 8. По заданному источнику S , представляющему язык L , построить источник, представляющий язык L^{-1} . Детерминизировать полученный источник. Вариант источника S взять из задачи 4.

Задача 9. По заданному источнику S (рис. П2.5), представляющему язык L во входном алфавите $X = \{0, 1, 2, 3, 4, 5\}$, построить источник, представляющий проекцию языка L при отображении $f: X \rightarrow Y$ с алфавита X на алфавит Y . Положить алфавит $Y = \{a, b, c\}$. Функция f определяется вариантом задания.

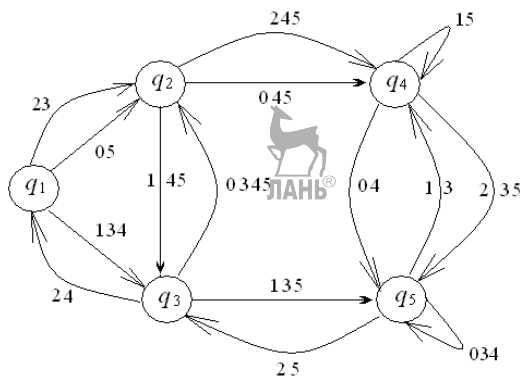


Рис. П2.5

	x	0	1	2	3	4	5
9.1.	$f(x)$	a	a	b	b	c	c
9.2.	$f(x)$	a	b	b	a	c	c
9.3.	$f(x)$	a	b	b	c	a	c
9.4.	$f(x)$	a	b	a	c	c	a
9.5.	$f(x)$	a	b	a	b	c	c

	x	0	1	2	3	4	5
9.17.	$f(x)$	b	a	c	b	a	c
9.18.	$f(x)$	b	a	c	b	c	a
9.19.	$f(x)$	a	b	c	a	b	c
9.20.	$f(x)$	b	a	a	a	b	c
9.21.	$f(x)$	b	c	a	a	b	c

9.6.	$f(x)$	b	a	b	b	c	c
9.7.	$f(x)$	b	a	b	a	c	c
9.8.	$f(x)$	b	a	b	c	a	c
9.9.	$f(x)$	b	a	a	c	c	a
9.10.	$f(x)$	b	b	a	a	c	c
9.11.	$f(x)$	b	b	a	c	a	c
9.12.	$f(x)$	b	b	c	c	c	a
9.13.	$f(x)$	b	b	c	a	a	c
9.14.	$f(x)$	b	b	c	a	c	a
9.15.	$f(x)$	b	b	c	c	a	a
9.16.	$f(x)$	b	a	c	a	b	c

9.22.	$f(x)$	b	c	a	b	a	c
9.23.	$f(x)$	b	c	a	b	c	a
9.24.	$f(x)$	c	a	b	b	b	c
9.25.	$f(x)$	c	a	a	a	b	c
9.26.	$f(x)$	c	a	a	b	a	c
9.26.	$f(x)$	c	a	a	b	c	a
9.28.	$f(x)$	c	b	b	a	b	c
9.29.	$f(x)$	c	b	a	b	a	c
9.30.	$f(x)$	c	b	b	b	c	a
9.31.	$f(x)$	c	b	b	a	c	a
9.32.	$f(x)$	c	c	b	b	a	b

Задача 10. По заданному источнику S со множеством входных символов $X = \{a, b, c\}$, допускающему язык L , построить источник, допускающий язык $Trunc(L, a)$. Множество входных символов $X = \{a, b, c\}$. Множество начальных состояний $Q_0 = \{q_1, q_2\}$. Множество выделенных состояний $F = \{q_3, q_5\}$. Вариант источника получить, добавив к граф-схеме на рис. П2.6 стрелку варианта. Например, для варианта 30 к граф-схеме источника надо добавить стрелку (q_3, c, q_2) .

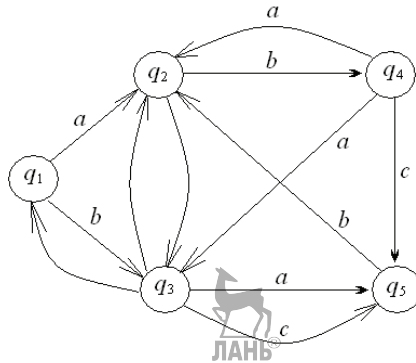


Рис. П2.6

Указание. Пусть $x = x(0)x(1)...x(k)aa...a$ при $x(k) \neq a$ есть слово в алфавите X , содержащем букву a . Тогда операция усечения слова x по букве a (обозначение: $Trunc(x, a)$) определяется как $Trunc(x, a) = x(0)x(1)...x(k)$. Если $M \subseteq X$, то множество $Trunc(M, a) = \{Trunc(x, a) : x \in M\}$.

Множество всех слов в алфавите $X = \{a, b, c\}$, не заканчивающихся на символ a , допустимо (детерминированным) автоматом A , приведенным на рис. П2.7. Начальное состояние есть q_0 . Множество выделенных состояний $F = \{q_b, q_c\}$.

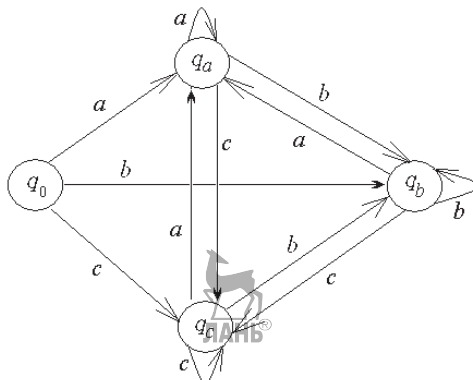


Рис. П2.7

Замечание. Класс языков, представимых источниками, замкнут относительно операции усечения.

В самом деле. Пусть язык M представим источником $S = (X, Q, Q_0, D, F)$. Пусть $a \in X$ и $a^k = aaa...a$, k раз. Построим источник $S' = (X, Q, Q_0, D, G)$, где $G = Q_a \cup F$, где $Q_a = \{q \in Q : \exists k \exists q' \in F ((q, a, q') \in D)\}$, т. е. G есть F , объединенное со множеством Q_a всех тех состояний $q \in Q$, для которых существует слово a^k при некотором натуральном k , переводящее источник S из состояния q в состояние q' , при этом $q' \in F$. Например, для источника S (рис. П2.6) множество Q_a строится следующим образом (рис. П2.8). Состояние q_4 добавляется к F , ибо a^k при $k = 1$ переводит q_4 в $q_3 \in F$. То же самое относительно q_3 . Поэтому $G = \{q_3, q_4\} \cup \{q_3, q_5\} = \{q_3, q_4, q_5\}$. Источник S' отличается от источника S лишь множеством выделенных состояний G . Источник S' с поведением $M' = Beh(S')$ допускает все те слова, которые допускают продолжение буквами a до слова, допустимого источником S , а также все слова, допустимые источником S .

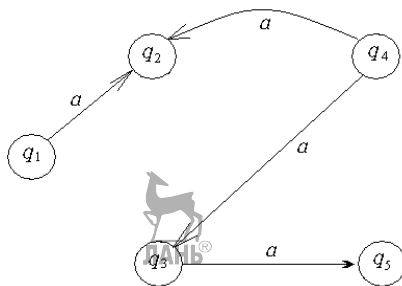


Рис. П2.8

Пусть S'' есть источник, допускающий множество M'' всех слов в алфавите X , не заканчивающихся на букву a . Источник, допускающий множество $M' \cap M''$, искомый.



Замечание. Множество всех слов в алфавите $X = \{a, b, c\}$, не заканчивающихся на символ a , допустимо (детерминированным) автоматом A , приведенным на рис. П2.7. Начальное состояние есть q_0 . Множество выделенных состояний $F = \{q_b, q_c\}$.

10.1. (q_1, c, q_2) .	10.2. (q_1, c, q_3) .	10.3. (q_1, c, q_4) .	10.4. (q_1, c, q_5) .
10.5. (q_2, c, q_3) .	10.6. (q_2, c, q_4) .	10.7. (q_2, c, q_5) .	10.8. (q_3, c, q_4) .
10.9. (q_3, c, q_5) .	10.10. (q_4, c, q_5) .	10.11. (q_1, b, q_2) .	10.12. (q_1, a, q_3) .
10.13. (q_1, b, q_4) .	10.14. (q_2, c, q_3) .	10.15. (q_2, c, q_4) .	10.16. (q_3, c, q_4) .
10.17. (q_2, b, q_1) .	10.18. (q_3, c, q_1) .	10.19. (q_4, c, q_1) .	10.20. (q_5, c, q_1) .
10.21. (q_3, c, q_2) .	10.22. (q_4, c, q_2) .	10.23. (q_5, c, q_2) .	10.24. (q_4, c, q_3) .
10.25. (q_5, c, q_3) .	10.26. (q_5, c, q_4) .	10.27. (q_2, c, q_1) .	10.28. (q_3, a, q_1) .
10.29. (q_4, c, q_1) .	10.30. (q_3, c, q_2) .		

Задача 11. Пусть $x = x(0)x(1)...x(k)$ есть некоторое слово в алфавите X , причем слово x содержит букву a . Аннулирование буквы a в слове x есть удаление буквы a в слове x всюду, где она в x встречается. Обозначим эту операцию через $An(x, a)$. Пусть $M \subseteq X^*$. Тогда $An(M, a) = \{An(x, a) : x \in M\}$. По заданному источнику S из задачи 10 со множеством входных символов $X = \{a, b, c\}$, допускающему язык L , построить источник, допускающий язык L . Детерминизировать полученный источник.



Приложение 3

Анализ конечных автоматов

Задача анализа конечных автоматов состоит в построении регулярного выражения R по данному конечному автомату A , такому, что представляемые ими множества слов одинаковы.

Алгоритм МакНотона–Ямады анализа автомата (McNaughton–Yamada)

Пусть автомат $A = (X, Q, q_0, \delta, F)$ со множеством состояний $Q = \{0, 1, \dots, n\}$ с одним финальным состоянием $F = \{r\}$.

Далее вычисление регулярных выражений выполняется индукцией по числу n состояний $0, 1, \dots, n$ автомата.

Базис. Шаг $k = 0$. Для i от 0 до n , для j от 0 до n выполнить следующее.

$R_{ij}^0 = \{w \in X^*: \delta(q_i, w) = q_j\}$; $i, j = 0, 1, \dots, n$, есть множество всех входных слов w , для которых $\delta(q_i, w) = q_j$; $i, j = 0, 1, \dots, n$; причем промежуточные состояния не могут быть больше k .

Предположение индукции. Шаг $k - 1$. Предположим, что на шаге $k - 1$ для i от 0 до n , для j от 0 до n регулярные выражения R_{ij}^{k-1} построены.

Шаг индукции. Шаг k . На шаге k для i от 0 до n , для j от 0 до n регулярные выражения

$$R_{ij}^k = R_{ij}^{k-1} + R_{ik}^{k-1} \cdot (R_{kk}^{k-1})^* \cdot R_{kj}^{k-1} \text{ (знак } + \text{ означает объединение множеств.)}$$

При этом вычисления упрощаются, если:

○ для $i = k, j = 0, 1, 2, 3$, регулярное выражение $R_{kj}^k = (R_{kk}^{k-1})^* \cdot R_{kj}^{k-1}$;

○ для $j = k, i = 0, 1, 2, 3$, регулярное выражение $R_{ik}^k = R_{ik}^{k-1} \cdot (R_{kk}^{k-1})^*$.

Регулярное выражение $R = R_{0r}^n$.

При множестве финальных состояний $F = \{i_1, i_2, \dots, i_r\}$ из Q регулярное выражение $R = E_{0,i_1}^n + E_{0,i_2}^n + \dots + E_{0,i_r}^n$.

Замечание. Алгоритм применим и к источникам.

Пример 1. Алгоритм МакНотона–Ямады (McNaughton–Yamada) для автомата, заданного диаграммой. На рис. ПЗ.1 представлена диаграмма с состояниями $0, 1, \dots, n$.

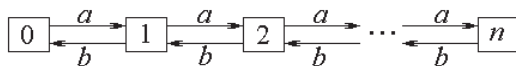


Рис. ПЗ.1

В примере рассматривается случай задания автомата диаграммой при $n = 3$ (рис. ПЗ.2). Начальное состояние 0, финальное состояние $n = 3$.

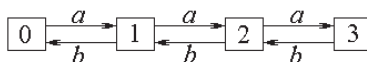


Рис. ПЗ.2

Шаг $k = 0$. Промежуточные состояния не больше $k = 0$.

$$R_{ij}^0 = \{w \in X^*: \delta(q_i, w) = q_j, i, j = 0, 1, 2, 3\}.$$

$$R_{00}^0 = \emptyset, R_{01}^0 = a, R_{02}^0 = \emptyset, R_{03}^0 = \emptyset;$$

$$R_{10}^0 = b, R_{11}^0 = \emptyset, R_{12}^0 = a, R_{13}^0 = \emptyset;$$

$$R_{20}^0 = \emptyset, R_{21}^0 = b, R_{22}^0 = \emptyset, R_{23}^0 = a;$$

$$R_{30}^0 = \emptyset, R_{31}^0 = \emptyset, R_{32}^0 = b, R_{33}^0 = \emptyset.$$



R_{ij}^0	0	1	2	3
0	$\emptyset$	a	$\emptyset$	$\emptyset$
1	b	$\emptyset$	a	$\emptyset$
2	$\emptyset$	b	$\emptyset$	a
3	$\emptyset$	$\emptyset$	b	$\emptyset$

Шаг $k = 1$.

$$R_{ij}^k = R_{ij}^{k-1} + R_{ik}^{k-1} \cdot (R_{kk}^{k-1})^* \cdot R_{kj}^{k-1}; i, j = 0, 1, 2, 3; i \neq k, j \neq k.$$

$$R_{kj}^k = (R_{kk}^{k-1})^* \cdot R_{kj}^{k-1}; i = k, j = 0, 1, 2, 3.$$

$$R_{ik}^k = R_{ik}^{k-1} \cdot (R_{kk}^{k-1})^*; j = k, i = 0, 1, 2, 3.$$

$$R_{ij}^1 = R_{ij}^0 + R_{i1}^0 \cdot (R_{11}^0)^* \cdot R_{1j}^0;$$

$$i = 0, j = 0; R_{00}^1 = R_{00}^0 + R_{01}^0 \cdot (R_{11}^0)^* \cdot R_{10}^0 = \emptyset + b \cdot (\emptyset)^* \cdot \emptyset = \emptyset.$$

$$i = 0, j = 1; R_{01}^1 = (R_{11}^0)^* \cdot R_{01}^0 = (\emptyset)^* \cdot \emptyset = \emptyset.$$

$$i = 0, j = 2; R_{02}^1 = R_{02}^0 + R_{01}^0 \cdot (R_{11}^0)^* \cdot R_{12}^0 = \emptyset + a \cdot (\emptyset)^* \cdot a = aa.$$

$$i = 0, j = 3; R_{03}^1 = R_{03}^0 + R_{01}^0 \cdot (R_{11}^0)^* \cdot R_{13}^0 = \emptyset + a \cdot (\emptyset)^* \cdot \emptyset = \emptyset.$$

$$i = 1, j = 0; R_{10}^1 = (R_{11}^0)^* \cdot R_{10}^0 = (\emptyset)^* \cdot b = b.$$

$$i = 1, j = 1; R_{11}^1 = (R_{11}^0)^* \cdot R_{11}^0 = (\emptyset)^* \cdot \emptyset = \emptyset.$$

$$i = 1, j = 2; R_{12}^1 = (R_{11}^0)^* \cdot R_{12}^0 = (\emptyset)^* \cdot a = a.$$

$$i = 1, j = 3; R_{13}^1 = (R_{11}^0)^* \cdot R_{13}^0 = (\emptyset)^* \cdot \emptyset = \emptyset.$$

$$i = 2, j = 0; R_{20}^1 = R_{20}^0 + R_{21}^0 \cdot (R_{11}^0)^* \cdot R_{10}^0 = \emptyset + b \cdot (\emptyset)^* \cdot b = bb.$$

$$i = 2, j = 1; R_{21}^1 = R_{21}^0 \cdot (R_{11}^0)^* = b \cdot (\emptyset)^* = b.$$

$$i = 2, j = 2; R_{22}^1 = R_{22}^0 + R_{21}^0 \cdot (R_{11}^0)^* \cdot R_{12}^0 = \emptyset + b \cdot (\emptyset)^* \cdot a = ba.$$

$$i = 2, j = 3; R_{23}^1 = R_{23}^0 + R_{21}^0 \cdot (R_{11}^0)^* \cdot R_{13}^0 = a + b \cdot (\emptyset)^* \cdot \emptyset = a.$$



$$i = 3, j = 0; R_{30}^1 = R_{30}^0 + R_{31}^0 \cdot (R_{11}^0)^* \cdot R_{10}^0 = \emptyset + \emptyset \cdot (\emptyset)^* \cdot b = \emptyset.$$

$$i = 3, j = 1; R_{31}^1 = R_{31}^0 \cdot (R_{11}^0)^* = \emptyset \cdot (\emptyset)^* = \emptyset.$$

$$i = 3, j = 2; R_{32}^1 = R_{32}^0 + R_{31}^0 \cdot (R_{11}^0)^* \cdot R_{12}^0 = b + \emptyset \cdot (\emptyset)^* \cdot a = b.$$

$$i = 3, j = 3; R_{33}^1 = R_{33}^0 + R_{31}^0 \cdot (R_{11}^0)^* \cdot R_{13}^0 = \emptyset + \emptyset \cdot (\emptyset)^* \cdot \emptyset = \emptyset.$$

R_{ij}^1	0	1	2	3
0	$\emptyset$	$\emptyset$	aa	$\emptyset$
1	b	$\emptyset$	a	$\emptyset$
2	b	b	ba	a
3	$\emptyset$	$\emptyset$	b	$\emptyset$

$$R_{00}^1 = \emptyset, R_{01}^1 = \emptyset, R_{02}^1 = aa, R_{03}^1 = \emptyset;$$

$$R_{10}^1 = b, R_{11}^1 = \emptyset, R_{12}^1 = a, R_{13}^1 = \emptyset;$$

$$R_{20}^1 = bb, R_{21}^1 = b, R_{22}^1 = bb, R_{23}^1 = a;$$

$$R_{30}^1 = \emptyset, R_{31}^1 = b, R_{32}^1 = bb, R_{33}^1 = a;$$

Шаг $k = 2$.

$$R_{ij}^k = R_{ij}^{k-1} + R_{ik}^{k-1} \cdot (R_{kk}^{k-1})^* \cdot R_{kj}^{k-1}; i, j = 0, 1, 2, 3; i \neq k, j \neq k.$$

$$R_{kj}^k = (R_{kk}^{k-1})^* \cdot R_{kj}^{k-1}; i = k, j = 0, 1, 2, 3.$$

$$R_{ik}^k = R_{ik}^{k-1} \cdot (R_{kk}^{k-1})^*; j = k, i = 0, 1, 2, 3.$$

$$R_{ij}^2 = R_{ij}^1 + R_{i1}^1 \cdot (R_{11}^1)^* \cdot R_{1j}^1; i, j = 0, 1, 2, 3.$$

$$i = 0, j = 0; R_{00}^2 = R_{00}^1 + R_{01}^1 \cdot (R_{11}^1)^* \cdot R_{10}^1 = \emptyset + aa \cdot (ba)^* \cdot bb = aa \cdot (ba)^* \cdot bb.$$

$$i = 0, j = 1; R_{01}^2 = R_{01}^1 + R_{01}^1 \cdot (R_{11}^1)^* \cdot R_{11}^1 = \emptyset + aa \cdot (ba)^* \cdot b = aa \cdot (ba)^* \cdot b.$$

$$i = 0, j = 2; R_{02}^2 = R_{02}^1 \cdot (R_{11}^1)^* = aa \cdot (ba)^*.$$

$$i = 0, j = 3; R_{03}^2 = R_{03}^1 + R_{01}^1 \cdot (R_{11}^1)^* \cdot R_{13}^1 = \emptyset + aa \cdot (ba)^* \cdot a = aa \cdot (ba)^* \cdot a.$$

$$i = 1, j = 0; R_{10}^2 = R_{10}^1 + R_{11}^1 \cdot (R_{11}^1)^* \cdot R_{10}^1 = b + a \cdot (ba)^* \cdot bb.$$

$$i = 1, j = 1; R_{11}^2 = R_{11}^1 + R_{11}^1 \cdot (R_{11}^1)^* \cdot R_{11}^1 = \emptyset + a \cdot (ba)^* \cdot b = a \cdot (ba)^* \cdot b.$$

$$i = 1, j = 2; R_{12}^2 = R_{12}^1 \cdot (R_{11}^1)^* = a \cdot (ba)^*.$$

$$i = 1, j = 3; R_{13}^2 = R_{13}^1 + R_{11}^1 \cdot (R_{11}^1)^* \cdot R_{13}^1 = \emptyset + aa \cdot (ba)^* \cdot a = aa \cdot (ba)^* \cdot a.$$

$$i = 2, j = 0; R_{20}^2 = (R_{22}^1)^* \cdot R_{20}^1 = (ba)^* \cdot bb.$$

$$i = 2, j = 1; R_{21}^2 = (R_{22}^1)^* \cdot R_{21}^1 = (ba)^* \cdot b.$$

$$i = 2, j = 2; R_{22}^2 = (R_{22}^1)^* \cdot R_{22}^1 = (ba)^* \cdot ba.$$

$$i = 2, j = 3; R_{23}^2 = (R_{22}^1)^* \cdot R_{23}^1 = (ba)^* \cdot a.$$

$$i = 3, j = 0; R_{30}^2 = R_{30}^1 + R_{32}^1 \cdot (R_{22}^1)^* \cdot R_{20}^1 = \emptyset + b \cdot (ba)^* \cdot bb = b \cdot (ba)^* \cdot bb.$$

$$i = 3, j = 1; R_{31}^2 = R_{31}^1 + R_{32}^1 \cdot (R_{22}^1)^* \cdot R_{21}^1 = \emptyset + b \cdot (ba)^* \cdot b = b \cdot (ba)^* \cdot b.$$

$$i = 3, j = 2; R_{32}^2 = (R_{32}^1)^* \cdot R_{32}^1 = (ba)^* \cdot ba.$$

$$i = 3, j = 3; R_{33}^2 = R_{33}^1 + R_{32}^1 \cdot (R_{22}^1)^* \cdot R_{23}^1 = \emptyset + b \cdot (ba)^* \cdot a = b \cdot (ba)^* \cdot a.$$

E_{ij}^2	0	1	2	3
0	$aa \cdot (ba)^* \cdot bb$	$aa \cdot (ba)^* \cdot b$	$aa \cdot (ba)^*$	$aa \cdot (ba)^* \cdot a$
1	$b + a \cdot (ba)^* \cdot bb$	$a \cdot (ba)^* \cdot b$	$a \cdot (ba)^*$	$aa \cdot (ba)^* \cdot a$
2	$(ba)^* \cdot bb$	$(ba)^* \cdot b$	$(ba)^* \cdot ba$	$(ba)^* \cdot a$
3	$b \cdot (ba)^* \cdot bb$	$b \cdot (ba)^* \cdot b$	$(ba)^* \cdot ba$	$b \cdot (ba)^* \cdot a$

$$\begin{aligned}
R_{00}^2 &= aa \cdot (ba)^* \cdot bb, R_{01}^2 = aa \cdot (ba)^* \cdot b, R_{02}^2 = aa \cdot (ba)^*, R_{03}^2 = aa \cdot (ba)^* \cdot a; \\
R_{10}^2 &= b + a \cdot (ba)^* \cdot bb, R_{11}^2 = a \cdot (ba)^* \cdot b, R_{12}^2 = a \cdot (ba)^*, R_{13}^2 = aa \cdot (ba)^* \cdot a; \\
R_{20}^1 &= (ba)^* \cdot bb, R_{21}^1 = (ba)^* \cdot b, R_{22}^1 = (ba)^* \cdot ba, R_{23}^1 = (ba)^* \cdot a; \\
R_{30}^2 &= b \cdot (ba)^* \cdot bb, R_{31}^2 = b \cdot (ba)^* \cdot b, R_{32}^2 = (ba)^* \cdot ba, R_{33}^2 = b \cdot (ba)^* \cdot a.
\end{aligned}$$

Шаг $k = 3$.

$$R_{ij}^k = R_{ij}^{k-1} + R_{ik}^{k-1} \cdot (R_{kk}^{k-1})^* \cdot R_{kj}^{k-1}; i, j = 0, 1, 2, 3; i \neq k, j \neq k.$$

$$R_{kj}^k = (R_{kk}^{k-1})^* \cdot R_{kj}^{k-1}; i = k, j = 0, 1, 2, 3.$$

$$R_{ik}^k = R_{ik}^{k-1} \cdot (R_{kk}^{k-1})^*; j = k, i = 0, 1, 2, 3.$$

$$R_{ij}^3 = R_{ij}^2 + R_{i1}^2 \cdot (R_{22}^2)^* \cdot R_{1j}^2; i, j = 0, 1, 2, 3.$$

$$\begin{aligned}
i = 0, j = 0; R_{00}^3 &= R_{00}^2 + R_{03}^2 \cdot (R_{33}^2)^* \cdot R_{30}^2 = \\
&= (aa(ba)^*bb) + (aa(ba)^*a) \cdot (b(ba)^*a) \cdot (b(ba)^*bb).
\end{aligned}$$

$$\begin{aligned}
i = 0, j = 1; R_{01}^3 &= R_{01}^2 + R_{03}^2 \cdot (R_{33}^2)^* \cdot R_{31}^2 = \\
&= (aa(ba)^*b) + (aa(ba)^*a) \cdot (b(ba)^*a)^* \cdot (b(ba)^*b).
\end{aligned}$$

$$\begin{aligned}
i = 0, j = 2; R_{02}^3 &= R_{02}^2 + R_{03}^2 \cdot (R_{33}^2)^* \cdot R_{32}^2 = \\
&= (aa(ba)^*) + (aa(ba)^*a) \cdot (b(ba)^*a)^* \cdot ((ba)^*ba).
\end{aligned}$$

$$i = 0, j = 3; R_{03}^3 = R_{03}^2 \cdot (R_{33}^2)^* = (aa(ba)^*a) \cdot (b(ba)^*a)^*.$$

$$\begin{aligned}
i = 1, j = 0; R_{10}^3 &= E_{10}^2 + R_{13}^2 \cdot (R_{33}^2)^* \cdot R_{30}^2 = \\
&= (b + a(ba)^*bb) + (ba(ba)^*a) \cdot (b \cdot (ba)^* \cdot a) \cdot (b(ba)^*bb).
\end{aligned}$$

$$\begin{aligned}
i = 1, j = 1; R_{11}^3 &= R_{11}^2 + R_{13}^2 \cdot (R_{33}^2)^* \cdot R_{31}^2 = \\
&= (a(ba)^*b) + (aa(ba)^*a) \cdot (b \cdot (ba)^* \cdot a)^* \cdot (b(ba)^*b).
\end{aligned}$$

$$\begin{aligned}
i = 1, j = 2; R_{12}^3 &= E_{12}^2 + R_{13}^2 \cdot (R_{33}^2)^* \cdot R_{32}^2 = \\
&= (a(ba)^*) + (aa(ba)^*a) \cdot (b \cdot (ba)^* \cdot a)^* \cdot (ba(ba)^*).
\end{aligned}$$

$$i = 1, j = 3; R_{13}^3 = R_{13}^2 \cdot (R_{33}^2)^* = (aa(ba)^*a) \cdot (b \cdot (ba)^* \cdot a)^*.$$

$$\begin{aligned}
i = 2, j = 0; R_{20}^3 &= R_{20}^2 + R_{23}^2 \cdot (R_{33}^2)^* \cdot R_{30}^2 = \\
&= ((ba)^*bb) + ((ba)^*a) \cdot (b \cdot (ba)^* \cdot a)^* \cdot (b(ba)^*bb).
\end{aligned}$$

$$\begin{aligned}
i = 2, j = 1; R_{21}^3 &= R_{21}^2 + R_{23}^2 \cdot (R_{33}^2)^* \cdot R_{31}^2 = \\
&= ((ba)^*bb) + ((ba)^*a) \cdot (b \cdot (ba)^* \cdot a)^* \cdot (b(ba)^*b).
\end{aligned}$$

$$\begin{aligned}
i = 2, j = 2; R_{22}^3 &= R_{22}^2 + R_{23}^2 \cdot (R_{33}^2)^* \cdot R_{32}^2 = \\
&= ((ba)^*ba) + ((ba)^*a) \cdot (b \cdot (ba)^* \cdot a)^* \cdot ((ba)^*ba).
\end{aligned}$$

$$i = 2, j = 3; R_{23}^3 = R_{23}^2 \cdot (R_{33}^2)^* = ((ba)^*a) \cdot (b \cdot (ba)^* \cdot a)^*.$$

$$i = 3, j = 0; R_{30}^3 = (R_{33}^2)^* \cdot R_{30}^2 = (b \cdot (ba)^* \cdot a)^* \cdot (b(ba)^*bb).$$

$$i = 3, j = 1; R_{31}^3 = (R_{33}^2)^* \cdot R_{31}^2 = (b(ba)^* \cdot a)^* \cdot (b(ba)^*b).$$

$$i = 3, j = 2; R_{32}^3 = (R_{33}^2)^* \cdot E_{32}^2 = (b \cdot (ba)^* \cdot a)^* \cdot ((ba)^*ba).$$

$$i = 3, j = 3; R_{33}^3 = (R_{33}^2)^* \cdot E_{33}^2 = (b \cdot (ba)^* \cdot a)^* \cdot (b(ba)^*a).$$

$$R_{00}^3 = (aa(ba)^*bb) + (aa(ba)^*a) \cdot (b(ba)^*a) \cdot (b(ba)^*bb).$$

$$R_{01}^3 = (aa(ba)^*b) + (aa(ba)^*a) \cdot (b(ba)^*a)^* \cdot (b(ba)^*b).$$

$$R_{02}^3 = (aa(ba)^*) + (aa(ba)^*a) \cdot (b(ba)^*a)^* \cdot ((ba)^*ba).$$

$$R_{03}^3 = (aa(ba)^*a) \cdot (b(ba)^*a)^*.$$

$$R_{10}^3 = (b + a(ba)^*bb) \cdot (ba(ba)^*a) \cdot (b \cdot (ba)^* \cdot a) \cdot (b(ba)^*bb).$$

$$R_{11}^3 = (a(ba)^*b) + (aa(ba)^*a) \cdot (b \cdot (ba)^* \cdot a)^* \cdot (b(ba)^*b).$$

$$R_{12}^3 = (a(ba)^*) + (aa(ba)^*a) \cdot (b \cdot (ba)^* \cdot a)^* \cdot (ba(ba)^*).$$

$$R_{13}^3 = (aa(ba)^*a) \cdot (b \cdot (ba)^* \cdot a)^*.$$

$$R_{20}^3 = ((ba)^*bb) + ((ba)^*a) \cdot (b \cdot (ba)^* \cdot a)^* \cdot (b(ba)^*bb).$$

$$R_{21}^3 = ((ba)^*bb) + ((ba)^*a) \cdot (b \cdot (ba)^* \cdot a)^* \cdot (b(ba)^*b).$$

$$R_{22}^3 = ((ba)^*ba) + ((ba)^*a) \cdot (b \cdot (ba)^* \cdot a)^* \cdot ((ba)^*ba).$$

$$R_{23}^3 = ((ba)^*a) \cdot (b \cdot (ba)^* \cdot a)^*.$$

$$R_{30}^3 = (b \cdot (ba)^* \cdot a)^* \cdot (b(ba)^*bb).$$

$$R_{31}^3 = (b(ba)^* \cdot a)^* \cdot (b(ba)^*b).$$

$$R_{32}^3 = (b \cdot (ba)^* \cdot a)^* \cdot ((ba)^*ba).$$

$$R_{33}^3 = (b \cdot (ba)^* \cdot a)^* \cdot (b(ba)^*a).$$



Для начального состояния 0 и $F = \{3\}$ регулярное выражение

$$R = E_{03}^3 = (aa(ba)^*a) \cdot (b(ba)^*a)^*.$$

Для начального состояния 0 и $F = \{0\}$ регулярное выражение

$$R = (aa(ba)^*bb) + (aa(ba)^*a) \cdot (b(ba)^*a) \cdot (b(ba)^*bb).$$

Для начального состояния 3 и $F = \{1, 2\}$ регулярное выражение

$$R = R_{31}^3 + R_{32}^3 = (b(ba)^* \cdot a)^* \cdot (b(ba)^*b) + (b \cdot (ba)^* \cdot a)^* \cdot ((ba)^*ba).$$

Пример 2. Пусть конечный автомат $A = (X, Q, q_0, T, F)$, где входной алфавит $X = \{a, b\}$, множество состояний $Q = \{0, 1, 2\}$, начальное состояние 0, множество финальных состояний $F = \{0, 1\}$. Функция переходов δ задана на рис. П3.3.

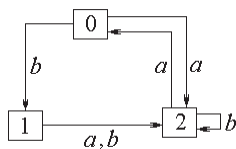


Рис. П3.3

Шаг $k = 0$. Промежуточные состояния не больше $k = 0$.

$$R_{ij}^0 = \{a \in X: \delta(q_i, a) = q_j, i, j = 0, 1, 2\}.$$

$$R_{00}^0 = \emptyset, R_{01}^0 = b, R_{02}^0 = a;$$

$$R_{10}^0 = \emptyset, R_{11}^0 = \emptyset, R_{12}^0 = a+b;$$

$$R_{20}^0 = a, R_{21}^0 = ab, R_{22}^0 = aa.$$

E_{ij}^0	0	1	2
0	$\emptyset$	b	a
1	$\emptyset$	$\emptyset$	$a+b$
2	a	ab	aa



Шаг $k = 1$.

$$R_{ij}^k = R_{ij}^{k-1} + R_{ik}^{k-1} \cdot (R_{kk}^{k-1})^* \cdot R_{kj}^{k-1}; i, j = 0, 1, 2; i \neq k, j \neq k.$$

$$R_{kj}^k = (R_{kk}^{k-1})^* \cdot R_{kj}^{k-1}; i = k, j = 0, 1, 2.$$

$$R_{ik}^k = R_{ik}^{k-1} \cdot (R_{kk}^{k-1})^*; j = k, i = 0, 1, 2.$$

$$R_{ij}^1 = R_{ij}^0 + R_{i1}^0 \cdot (R_{11}^0)^* \cdot R_{1j}^0; i, j = 0, 1, 2.$$

$$i = 0, j = 0; R_{00}^1 = R_{00}^0 + R_{01}^0 \cdot (R_{11}^0)^* \cdot E_{10}^0 = \emptyset + b \cdot (\emptyset)^* \cdot \emptyset = \emptyset.$$

$$i = 0, j = 1; R_{01}^1 = R_{01}^0 \cdot (R_{11}^0)^* = b \cdot (\emptyset)^* = b.$$

$$i = 0, j = 2; R_{02}^1 = R_{02}^0 + R_{01}^0 \cdot (R_{11}^0)^* \cdot R_{12}^0 = a + b \cdot (\emptyset)^* \cdot (a+b) = a + b(a+b).$$

$$i = 1, j = 0; R_{10}^1 = (R_{11}^0)^* \cdot R_{10}^0 = (\emptyset)^* \cdot \emptyset = \emptyset.$$

$$i = 1, j = 1; R_{11}^1 = (R_{11}^0)^* \cdot E_{11}^0 = (\emptyset)^* \cdot \emptyset = \emptyset.$$

$$i = 1, j = 2; R_{12}^1 = (R_{11}^0)^* \cdot R_{12}^0 = (\emptyset)^* \cdot (a+b) = a+b.$$

$$i = 2, j = 0; R_{20}^1 = R_{20}^0 + E_{21}^0 \cdot (R_{11}^0)^* \cdot R_{10}^0 = a + ab \cdot (\emptyset)^* \cdot \emptyset = a.$$

$$i = 2, j = 1; R_{21}^1 = R_{21}^0 \cdot (R_{11}^0)^* = ab \cdot (\emptyset)^* = ab.$$

$$i = 2, j = 2; R_{22}^1 = R_{22}^0 + R_{21}^0 \cdot (R_{11}^0)^* \cdot R_{12}^0 = b + ab \cdot (\emptyset)^* \cdot (a+b) = b + ab(a+b).$$

R_{ij}^1	0	1	2
0	$\emptyset$	b	$a + b(a+b)$
1	$\emptyset$	$\emptyset$	$a+b$
2	a	ab	$b + ab(a+b)$

$$R_{00}^1 = \emptyset, R_{01}^1 = b, R_{02}^1 = a + b(a+b);$$

$$R_{10}^1 = \emptyset, R_{11}^1 = \emptyset, R_{12}^1 = a + b;$$

$$R_{20}^1 = a, R_{21}^1 = ab, R_{22}^1 = b + ab(a+b).$$

Шаг $k = 2$.

$$R_{ij}^k = R_{ij}^{k-1} + R_{ik}^{k-1} \cdot (R_{kk}^{k-1})^* \cdot R_{kj}^{k-1}; i, j = 0, 1, 2; i \neq k, j \neq k.$$

$$R_{kj}^k = (R_{kk}^{k-1})^* \cdot R_{kj}^{k-1}; i = k, j = 0, 1, 2.$$

$$R_{ik}^k = R_{ik}^{k-1} \cdot (R_{kk}^{k-1})^*; j = k, i = 0, 1, 2.$$

$$R_{ij}^2 = R_{ij}^1 + R_{i1}^1 \cdot (R_{22}^1)^* \cdot R_{1j}^1; i, j = 0, 1, 2.$$

$$i = 0, j = 0; R_{00}^2 = R_{00}^1 + R_{01}^1 \cdot (R_{22}^1)^* \cdot R_{10}^1 = \emptyset + (a + b(a+b)) \cdot (b + ab(a+b))^* \cdot a = (a + b(a+b)) \cdot (b + ab(a+b))^* \cdot a.$$

$$i = 0, j = 1; R_{01}^2 = R_{01}^1 + R_{01}^1 \cdot (R_{22}^1)^* \cdot R_{11}^1 = b + (a + b(a+b)) \cdot (b + ab(a+b))^* \cdot ab.$$

$$i = 0, j = 2; R_{02}^2 = R_{02}^1 \cdot (R_{22}^1)^* = (a + b(a+b)) \cdot (b + ab(a+b))^*.$$

$$i = 1, j = 0; R_{10}^2 = R_{10}^1 + R_{12}^1 \cdot (R_{22}^1)^* \cdot R_{20}^1 = (a+b) \cdot (b + ab(a+b))^* \cdot a.$$

$$i = 1, j = 1; R_{11}^2 = R_{11}^1 + R_{12}^1 \cdot (R_{22}^1)^* \cdot R_{21}^1 = \emptyset + (a+b) \cdot (b + ab(a+b))^* \cdot ab.$$

$$i = 1, j = 2; R_{12}^2 = R_{12}^1 \cdot (R_{22}^1)^* = (a+b) \cdot (b + ab(a+b))^*.$$

$$i = 2, j = 0; R_{20}^2 = (R_{22}^1)^* \cdot R_{20}^1 = (b + ab(a+b))^* \cdot a.$$

$$i = 2, j = 1; R_{21}^2 = (R_{22}^1)^* \cdot R_{21}^1 = (b + ab(a+b))^* \cdot ab.$$

$$i = 2, j = 2; R_{22}^2 = (R_{22}^1)^* \cdot R_{22}^1 = (b + ab(a+b))^* \cdot (b + ab(a+b)).$$

$$R_{00}^2 = (a + b(a+b)) \cdot (b + ab(a+b))^* \cdot a.$$

$$R_{01}^2 = b + (a + b(a+b)) \cdot (b + ab(a+b))^* \cdot ab.$$

$$R_{02}^2 = (a + b(a + b)) \cdot (b + ab(a + b))^*.$$

$$R_{10}^2 = (a + b) \cdot (b + ab(a + b))^* \cdot a.$$

$$R_{11}^2 = (a + b) \cdot (b + ab(a + b))^* \cdot ab.$$

$$R_{12}^2 = (a + b) \cdot (b + ab(a + b))^*.$$

$$R_{20}^2 = (b + ab(a + b))^* \cdot a.$$

$$R_{21}^2 = (b + ab(a + b))^* \cdot ab.$$

$$R_{22}^2 = (b + ab(a + b))^* \cdot (b + ab(a + b)).$$

Для начального состояния 0 и $F = \{0, 1\}$ регулярное выражение

$$R = R_{00}^2 + R_{01}^2 = (a + b(a + b)) \cdot (b + ab(a + b))^* \cdot a + b + (a + b(a + b)) \cdot (b + ab(a + b))^* \cdot ab.$$

Замечание. В двух предыдущих примерах вычислены регулярные выражения для любых начальных и финальных состояний данного конечного автомата. Для конкретных начального и конечного состояний число промежуточных вычислений можно сократить, выбрав только необходимые промежуточные регулярные выражения.

Алгоритм МакКласки анализа автомата

методом удаления состояний (Brzowski J. A., McCluskey E. J.)

Пусть $A = (X, Q, q_0, \delta, F)$ есть источник со множеством состояний $Q = \{0, 1, \dots, n\}$, с единственным начальным состоянием 0, с единственным финальным состоянием $f = n$. Источник задан диаграммой D . Далее индукция по удаляемым состояниям.

Базис. Шаг 0. Каждую дугу диаграммы D_0 , ведущую из состояния i в состояние j (дуга может быть петлей) и помеченную входными символами $a_1, \dots, a_u$ (среди них может быть пустой символ ϵ), пометить регулярным выражением $R = a_1 + \dots + a_u$. Если некоторое состояние имеет петлю, помеченную регулярным выражением R без внешней звезды, то заменить R на R^* . Построена диаграмма D_0 .

Предположение индукции. Допустим, что на шаге $k - 1$ построена диаграмма D_{k-1} , в которой число состояний уменьшилось на $k - 1$ состояний и в которой каждая дуга помечена некоторым регулярным выражением.

Шаг индукции. Шаг k . Выберем в D_{k-1} любое состояние s вне $\{0, f\}$ для удаления из D_{k-1} .

Пусть в D_{k-1} в состояние s заходят дуги из состояний $p_1, \dots, p_u$, помеченные регулярными выражениями $P_1, \dots, P_u$ соответственно.

Пусть в D_{k-1} из состояния s выходят дуги в состояния $q_1, \dots, q_v$, помеченные регулярными выражениями $Q_1, \dots, Q_v$ соответственно (табл. ПЗ.1).

Таблица ПЗ.1

Состояния	p_1	...	p_u	q_1	...	q_v
Регулярные выражения	P_1	...	P_u	Q_1	...	Q_v

Пусть в D_{k-1} всякая дуга из p_i ($i = 1, \dots, u$) в q_j ($j = 1, \dots, v$) помечена регулярным выражением R_{ij} . Если дуги из p_i в q_j нет, то положим $R_{ij} = \emptyset$. Построим табл. ПЗ.2 значений регулярных выражений R_{ij} .

Таблица ПЗ.2

p_i / q_j	q_1	...	q_v
p_1	R_{11}	...	R_{1v}
...	...	...	...
p_u	R_{u1}	...	R_{uv}



Далее для простоты будем иногда состояния p_i , q_j обозначать индексами i , j соответственно. Состояние s из D_{k-1} удаляем следующим образом.

Для всех $i = 1, \dots, u$; $j = 1, \dots, v$ строим графы входящих из p_i в s ребер и исходящих из s в q_j ребер и их пометки P_i , Q_j соответственно, а также дуги от p_i до q_j (если они существуют) с пометками R_{ij} из табл. ПЗ.1 и табл. ПЗ.2 (рис. ПЗ.4).

Удаляем состояние s и все инцидентные (принадлежащие) s дуги (рис. ПЗ.5).

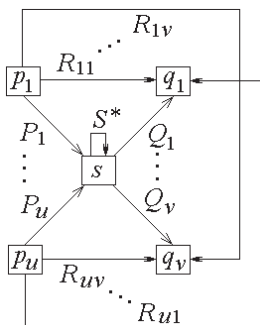


Рис. ПЗ.4

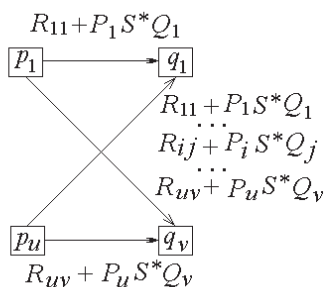


Рис. ПЗ.5

Если удаляются две дуги из p_i в s и из s в q_j , то дуга из p_i в q_j помечается:

- регулярным выражением $R_{ij} + P_i \cdot S^* \cdot Q_j$, если петля на s помечена регулярным выражением S^* (рис. ПЗ.6);
- регулярным выражением $R_{ij} + P_i \cdot \emptyset^* \cdot Q_j$, если s петли не имеет (рис. ПЗ.7).

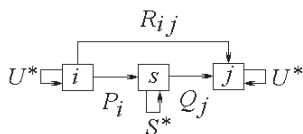


Рис. ПЗ.6

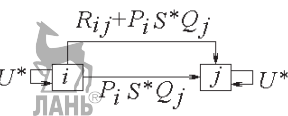


Рис. ПЗ.7

При удалении состояния s при $q_i = p_i$ появляется петля (рис. ПЗ.8, ПЗ.9).

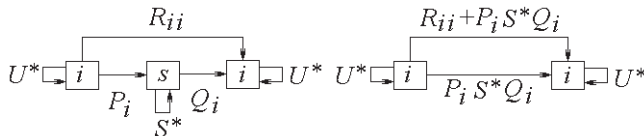


Рис. П3.8

Рис. П3.9

Петля на рис. П3.9 изображена на рис. П3.10 и после упрощений на рис. П3.11.



Рис. П3.10

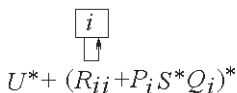


Рис. П3.11

В D_{k-1} удаляем состояние s и все его дуги. Достаиваем получившуюся диаграмму графами из рис. П3.7, П3.11 и получаем диаграмму D_k .

Индукция заканчивается следующими двумя случаями.

Случай 1. Начальное состояние является допускающим. Поэтому индукция заканчивается диаграммой D_1 с одним лишь начальным состоянием 0. Тогда состояние 0 имеет петлю, помеченную некоторым регулярным выражением $R = T$ (рис. П3.12). Оно представляет язык, допустимый данным источником A .

Случай 2. Начальное состояние не является допускающим. Тогда индукция заканчивается диаграммой D_2 с лишь начальным и финальным состояниями 0 и f , дуги которых помечены, например, следующим образом (рис. П3.12).

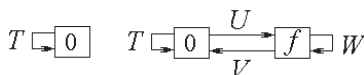


Рис. П3.12

Тогда регулярное выражение $R = (T + UW^*V)^*UW^*$ представляет язык, допустимый источником A . Не следует дублировать звезду у W , если W звезду уже имеет.

Если отсутствуют петли у состояния 0 или у состояния f , то в R соответствующие T и W заменяются на $\emptyset$.

Если отсутствует дуга из f в 0, то $R = (T + UW^*)UW^*$.

Замечание. Вид (форма) регулярного выражения зависит от порядка удаления состояний конечного автомата.

Пример 1. Диаграмма D автомата A приведена на рис. П3.13. Начальное состояние 0, финальное состояние $f = 1$. Найти регулярное выражение R , представляющее конечноавтоматный язык L .

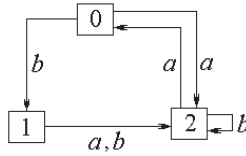


Рис. ПЗ.13

Начальное состояние 0, финальное состояние 1

Шаг $k = 0$. Каждую дугу диаграммы D_0 , ведущую из состояния i в состояние j (дуга может быть петлей) и помеченную входными символами $a_1, \dots, a_u$ (среди них может быть пустой символ ϵ), пометить регулярным выражением $R = a_1 + \dots + a_u$. Состояние 2 в D_0 имеет петлю, помеченную регулярным выражением $R = b$. Заменяем R на b^* (рис. ПЗ.14).

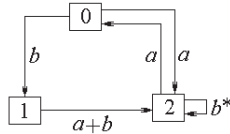


Рис. ПЗ.14

Шаг $k = 1$. Состояние $s = 2$ вне $\{0, f\}$ есть единственное состояние для удаления.

В D_{k-1} в состояние $s = 2$ заходит дуга:

- из состояния 1 с регулярным выражением $a + b$,
- из состояния 0 с регулярным выражением a .

В D_{k-1} из состояния $s = 2$ выходит дуга в состояние 0 с регулярным выражением a (рис. ПЗ.15).

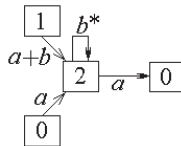


Рис. ПЗ.15

Удаление состояния 2 на рис. ПЗ.15 дает рис. ПЗ.16 и ПЗ.17. Цикл на рис. ПЗ.17 помещаем на рис. ПЗ.16 и получаем рис. ПЗ.18.

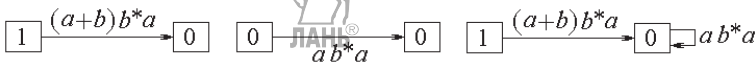


Рис. ПЗ.16

Рис. ПЗ.17

Рис. ПЗ.18

Из диаграммы на рис. ПЗ.12 удаляем состояние 2 и все инцидентные с 2 дуги. Получаем рис. ПЗ.19. Достаиваем диаграмму на рис. ПЗ.19 диаграммой на рис. ПЗ.18 и получаем диаграмму D_1 на рис. ПЗ.20.

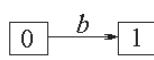


Рис. ПЗ.19

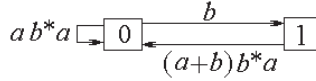


Рис. ПЗ.20

В диаграмме $D_{k=1}$ (рис. ПЗ.20) регулярное выражение $T = ab^*a$, $U = b$, $V = (a+b)b^*a$, $W = \emptyset$. Тогда

$$R = (T + UW^*V)^*UW^* = (ab^*a + b\emptyset^*(a+b)b^*a)^*b\emptyset^* = (ab^*a + b(a+b)b^*a)^*b.$$

Регулярное выражение $R = (ab^*a + b(a+b)b^*a)^*b$ представляет язык L , допустимый источником A .



Начальное состояние 0, финальное состояние 2

Удаляем состояние 1. Исходя из диаграммы на рис. ПЗ.12, строим диаграмму входящих и выходящих для состояния 1 стрелок (рис. ПЗ.21). Удаляем состояние 1 и получаем диаграмму на рис. ПЗ.22.

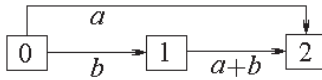


Рис. ПЗ.21

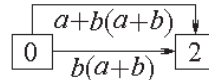


Рис. ПЗ.22

Удаляем из диаграммы на рис. ПЗ.12 состояние 1 и инцидентные с ним дуги (рис. ПЗ.23), достаиваем диаграмму на рис. ПЗ.23 диаграммой на рис. ПЗ.22 и получаем диаграмму на рис. ПЗ.24, эквивалентную диаграмме на рис. ПЗ.25.

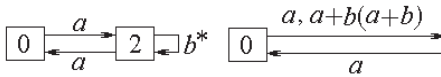


Рис. ПЗ.23

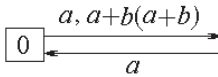


Рис. ПЗ.24

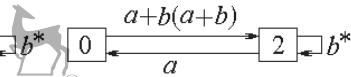


Рис. ПЗ.25

В диаграмме $D_{k=1}$ (рис. ПЗ.25) регулярное выражение $T = \emptyset$, $U = a+b(a+b)$, $V = a$, $W = b^*$. Тогда

$$R = (T + UW^*V)^*UW^* = (\emptyset + (a+b(a+b)) \cdot b^* \cdot a)^* \cdot (a+b(a+b)) \cdot b^* = ((a+b(a+b)) \cdot b^* \cdot a)^* \cdot (a+b(a+b)) \cdot b^*.$$

Регулярное выражение $R = ((a+b(a+b)) \cdot b^* \cdot a)^* \cdot (a+b(a+b)) \cdot b^*$ представляет язык, допустимый источником A .

Начальное состояние 0, финальное состояние 0

После удаления состояния 2 из диаграммы на рис. П3.12 получаем следующую диаграмму (рис. П3.26).

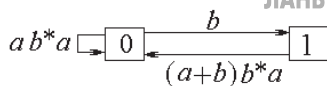


Рис. П3.26

Строим диаграмму входящих и исходящих для состояния 1 стрелок (рис. П3.27).

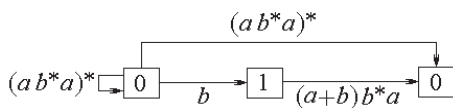


Рис. П3.27

Удаляем на рис. П3.12 состояние 1 и получаем рис. П3.28 и П3.29.

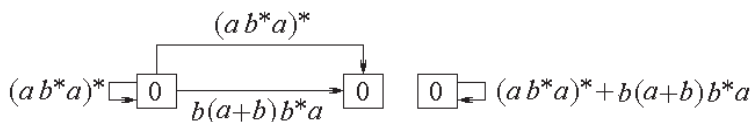


Рис. П3.28

Рис. П3.29

В диаграмме $D_{k=1}$ (рис. П3.29) регулярное выражение $T = (ab^*a)^* + b(a+b)b^*a$. Тогда $R = T = (ab^*a)^* + b(a+b)b^*a$.

Регулярное выражение $R = (ab^*a)^* + b(a+b)b^*a$ представляет язык, допустимый источником A .

Пример 2. Диаграмма D автомата A приведена на рис. П12.30. Начальное состояние 0, финальное состояние $f = 3$. Найти регулярное выражение R , представляющее конечноавтоматный язык L .

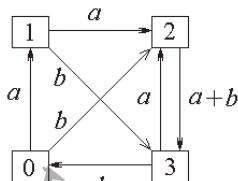


Рис. П3.30

Удаление состояния 2. Все входящие в состояние 2 и выходящие из состояния 2 дуги приведены на рис. ПЗ.31. Цикл из состояния 3 в состояние 3 подсоединяем к состоянию 3 и снабжаем пометку дуги звездой. Получаем рис. ПЗ.32.

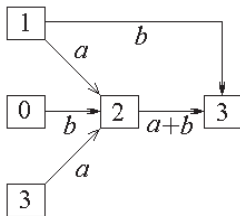


Рис. ПЗ.31

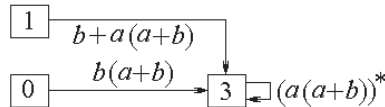


Рис. ПЗ.32

Из диаграммы на рис. ПЗ.30 удаляем состояние 2 и все его дуги. Результат достраиваем графом на рис. ПЗ.32. Получаем новую диаграмму (рис. ПЗ.33).

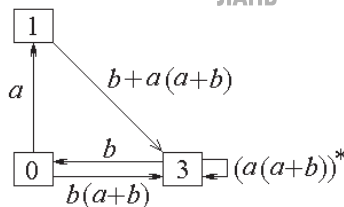


Рис. ПЗ.33

Удаление состояния 1 в диаграмме на рис. ПЗ.33. Для этого строим диаграмму входящих и исходящих ребер для состояния 1 (рис. ПЗ.34). Удаляем в рис. ПЗ.34 состояние 1 и получаем рис. ПЗ.35.

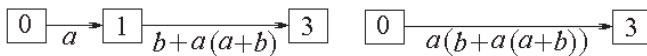


Рис. ПЗ.34

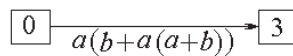


Рис. ПЗ.35

Удаляем из рис. ПЗ.33 состояние 1 с его дугами, получаем рис. ПЗ.36.

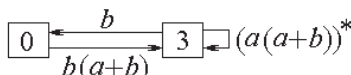


Рис. ПЗ.36

Достраиваем рис. ПЗ.36 с помощью рис. ПЗ.35 и получаем граф на рис. ПЗ.37.

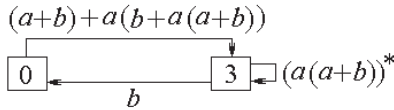


Рис. ПЗ.37

В диаграмме на рис. ПЗ.37 регулярное выражение $T = \emptyset, U = (a+b) + a(b+a(a+b)), V = b, W = (a(a+b))^*$. Тогда

$$R = (T + UW^*V)^*UW^* = (\emptyset + (a+b) + a(b+a(a+b))) \cdot (a(a+b))^* \cdot b \cdot ((a+b) + a(b+a(a+b))) \cdot (a(a+b))^*.$$

Регулярное выражение

$$R = ((a+b) + a(b+a(a+b))) \cdot (a(a+b))^* \cdot b \cdot ((a+b) + a(b+a(a+b))) \cdot (a(a+b))^*.$$

представляет язык, допустимый источником А.

Пример 3. Источник А задан диаграммой на рис. ПЗ.38. Начальное состояние 0, финальное состояние $f = 3$. Найти регулярное выражение, представляющее язык, определяемый источником А.

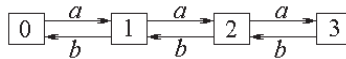


Рис. ПЗ.38

1. Удаление состояния 2 в диаграмме на рис. ПЗ.38. Строим все входящие в состояние 2 дуги, все дуги, выходящие из состояния 2 дуги, а также все дуги из левых состояний в правые состояния там, где они существуют (рис. ПЗ.39). Строим все пути от левых состояний в правые состояния (рис. ПЗ.40).

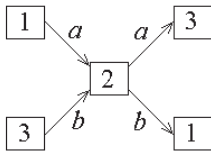


Рис. ПЗ.39

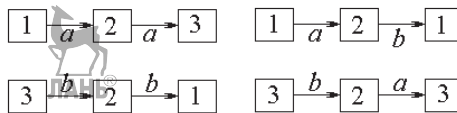


Рис. ПЗ.40

В диаграмме на рис. ПЗ.40 удаляем состояние 2, сохраняя пометки на принадлежащих состоянию 2 дугах (рис. ПЗ.41). Снабжаем звездами пометки на появившихся петлях (рис. ПЗ.42).

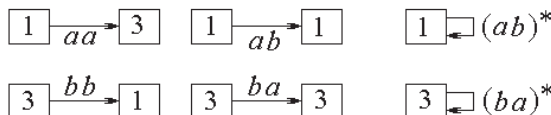


Рис. ПЗ.41

Рис. ПЗ.42

В диаграмме на рис. ПЗ.38 удаляем состояние 2 и все его дуги вместе пометками (рис. ПЗ.43). Достаиваем диаграмму на рис. ПЗ.43 дугами с их пометками из диаграмм на рис. ПЗ.41 и ПЗ.42 (рис. ПЗ.44).

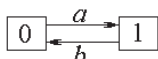


Рис. ПЗ.43

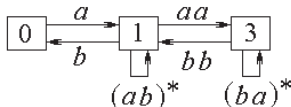


Рис. ПЗ.44

2. Удаление состояния 1 в диаграмме на рис. ПЗ.44. Строим все входящие в состояние 1 дуги, все выходящие из состояния 1 дуги, а также все дуги из левых состояний в правые состояния там, где они существуют (рис. ПЗ.45). Строим все пути от левых состояний в правые состояния (рис. ПЗ.46).

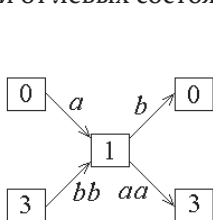


Рис. ПЗ.45

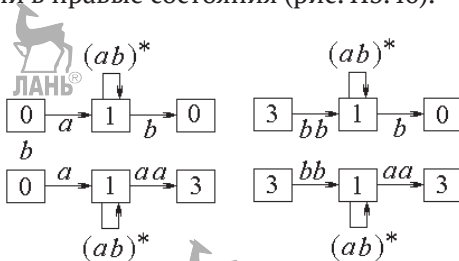


Рис. ПЗ.46

В двух диаграммах рис. ПЗ.46 удаляем состояние 1 и все его дуги, сохраняя пометки на принадлежащих состоянию 1 дугах. Снабжаем звездами пометки на появившихся петлях (рис. ПЗ.47, ПЗ.48).

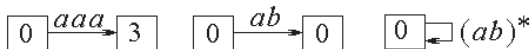


Рис. ПЗ.47

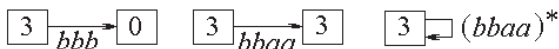


Рис. ПЗ.48

Задача 1. Диаграмма источника задана на рис. ПЗ.49. Методом МакНотона–Ямады найти регулярное выражение R , представляющее язык, допустимый данным источником. Начальные и финальные состояния источника указаны как пары (i, j) ; $i, j = 0, 1, \dots, 5$. Варианты после задачи 2.

Задача 2. Диаграмма источника задана на рис. ПЗ.49. Методом МакКласки найти регулярное выражение R , представляющее язык, допустимый данным источником. Начальные и финальные состояния источника указаны как пары (i, j) ; $i, j = 0, 1, \dots, 5$.

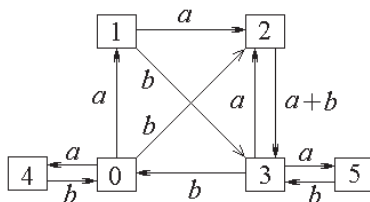


Рис. ПЗ.49

1.1. (0, 0).	1.2. (0, 1).	1.3. (0, 2).	1.4. (0, 3).	1.5. (0, 4).	1.6. (0, 5).
1.7. (1, 0).	1.8. (1, 1).	1.9. (1, 2).	1.10. (1, 3).	1.11. (1, 4).	1.12. (1, 5).
1.13. (2, 0).	1.14. (2, 1).	1.15. (2, 2).	1.16. (2, 3).	1.17. (2, 4).	1.18. (2, 5).
1.19. (3, 0).	1.20. (3, 1).	1.21. (3, 2).	1.22. (3, 3).	1.23. (3, 24).	1.24. (3, 5).
1.25. (4, 0).	1.26. (4, 1).	1.27. (4, 2).	1.28. (4, 3).	1.29. (4, 4).	1.30. (4, 5).
1.31. (5, 0).	1.32. (5, 1).	1.33. (5, 2).	1.34. (5, 3).	1.35. (5, 4).	1.36. (5, 5).



Приложение 4

Синтез



конечных автоматов

Синтез конечного автомата по регулярному выражению R состоит в построении такого конечного автомата A , что языки, представимые автоматом A и регулярным выражением R , совпадают.

Задача 1. Найти источник для языка в алфавите $\{0,1,2\}$, представимого регулярным выражением. Детерминизировать полученный источник.

Пример. Найти источник для языка в алфавите $\{a,b,c\}$, представимого регулярным выражением $R = a \cdot b^* \cdot (a \vee b \cdot c)^* \cdot c$. Детерминизировать полученный источник.

1. Индукцией по построению формулы R выполнить парсинг регулярного выражения R , то есть синтаксический разбор регулярного выражения R . Дерево разбора от корня к листьям показано на рис. П4.1.

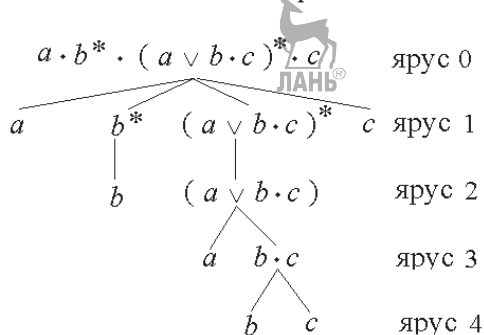


Рис. П4.1

Построение источника для регулярного выражения R выполнять от листьев к корню дерева разбора. В следующих диаграммах строящихся источников знак i (initial) означает начальное состояние, а знак f (final) – финальное состояние.

Для яруса 4 из рис. П4.1 построить диаграммы источников для языков регулярных выражений b и c (рис. П4.2).

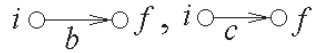


Рис. П4.2

Для яруса 3 из рис. П4.1 построить диаграммы источников для языков регулярных выражений a и $b \cdot c$ (рис. П4.3).

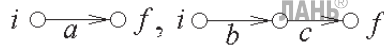


Рис. П4.3

Для яруса 2 из рис. П4.1 построить диаграммы источников для языков регулярных выражений b и $a \vee b \cdot c$ (рис. П4.4).

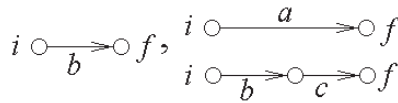


Рис. П4.4

Для яруса 1 из рис. П4.1 построить диаграммы источников для языков регулярных выражений a , $(a \vee b \cdot c)^*$, c (рис. П4.5).

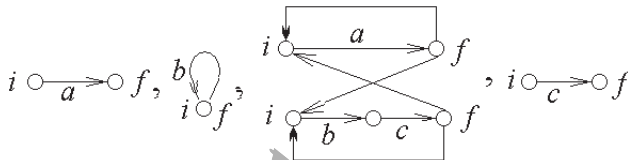


Рис. П4.5

Для яруса 0 из рис. 1 построить диаграмму источника для языка регулярного выражения $R = a \cdot (a \vee b \cdot c)^* \cdot c$ (рис. П4.6).

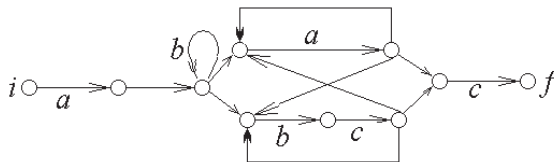


Рис. П4.6

Источник A , допускающий язык для регулярного выражения R , построен. Обозначим его состояния (рис. П4.7).

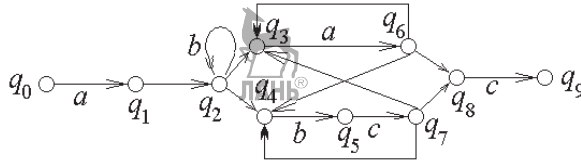


Рис. П4.7

Источник $A = (X, Q, Q_0, T, F)$, где

$X = \{a, b, c\}$,

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9\}$,

$Q_0 = \{q_0\}$,

$T = \{(q_0, a, q_1), (q_1, *, q_2), (q_2, b, q_2), (q_2, *, q_3), (q_2, *, q_4), (q_3, a, q_6), (q_4, b, q_5), (q_5, c, q_7), (q_6, *, q_3), (q_6, *, q_4), (q_6, *, q_8), (q_7, *, q_4), (q_7, *, q_3), (q_7, *, q_8), (q_8, c, q_9)\}$,

$F = \{q_9\}$.

Таблица переходов состояний источника A

	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9
a	q_1			q_6						
b			q_2		q_5					
c						q_7			q_9	
$*$		q_2	q_3, q_4				q_3, q_4, q_8	q_5, q_4, q_8		

Далее следует детерминизировать этот источник и получить требуемый конечный автомат.

Задача. Найти источник для языка в алфавите $\{0, 1, 2\}$, представимого регулярным выражением. Детерминизировать полученный источник.

1.1. $0 \cdot 1^* (0 \cdot 2 \cdot 1)^* \vee 2$.

1.2. $0 \cdot 1^* \cdot (0 \cdot 2 \vee 1)^* 2$.

1.3. $0 \cdot 1^* (0 \cdot 2 \vee 1)^* \vee 2$.

1.4. $0 \cdot 1^* \cdot (0 \vee 2 \cdot 1)^* 2$.

1.5. $0 \cdot 1^* (0 \vee 2 \cdot 1)^* \vee 2$.

1.6. $0 \cdot 1^* \cdot (0 \vee 2 \vee 1)^* 2$.

1.7. $0 \cdot 1^* (0 \vee 2 \vee 1)^* \vee 2$.

1.8. $0 \cdot 1^* \vee (0 \cdot 2 \cdot 1)^* 2$.

1.9. $0 \cdot 1^* \vee (0 \cdot 2 \cdot 1)^* \vee 2$.

1.10. $0 \cdot 1^* \vee (0 \cdot 2 \vee 1)^* 2$.

1.11. $0 \cdot 1^* \vee (0 \cdot 2 \vee 1)^* \vee 2$.

1.12. $0 \cdot 1^* \vee (0 \vee 2 \cdot 1)^* 2$.

1.13. $0 \cdot 1^* \vee (0 \vee 2 \cdot 1)^* \vee 2$.

1.14. $0 \cdot 1^* \vee (0 \vee 2 \vee 1)^* 2$.

1.15. $0 \cdot 1^* \vee (0 \vee 2 \vee 1)^* \vee 2$.

1.16. $0 \vee 1^* \cdot (0 \cdot 2 \cdot 1)^* 2$.

1.17. $0 \vee 1^* \cdot (0 \cdot 2 \cdot 1)^* \vee 2$.

1.18. $0 \vee 1^* \cdot (0 \cdot 2 \vee 1)^* 2$.

1.19. $0 \vee 1^* \cdot (0 \cdot 2 \vee 1)^* \vee 2$.

1.20. $0 \vee 1^* \cdot (0 \vee 2 \cdot 1)^* 2$.

1.21. $0 \vee 1^* \cdot (0 \vee 2 \cdot 1)^* \vee 2$.

1.22. $0 \vee 1^* \cdot (0 \vee 2 \vee 1)^* 2$.

1.23. $0 \vee 1^* \cdot (0 \vee 2 \vee 1)^* \vee 2$.

1.24. $0 \vee 1^* \cdot (0 \cdot 2 \cdot 1)^* 2$.

1.25. $0 \vee 1^* \cdot (0 \cdot 2 \cdot 1)^* \vee 2$.

1.26. $0 \vee 1^* \cdot (0 \cdot 2 \vee 1)^* 2$.

1.27. $0 \vee 1^* \cdot (0 \cdot 2 \vee 1)^* \vee 2$.

1.28. $0 \vee 1^* \cdot (0 \vee 2 \cdot 1)^* 2$.

1.29. $0 \vee 1^* \cdot (0 \vee 2 \cdot 1)^* \vee 2$.

1.30. $0 \vee 1^* \cdot (0 \vee 2 \vee 1)^* 2$.

1.31. $0 \vee 1^* \cdot (0 \vee 2 \vee 1)^* \vee 2$.

1.32. $0 \cdot 1^* \cdot (0 \cdot 2 \cdot 1)^* 2$.

Литература

1. *Алексеев В. Б.* Введение в теорию сложности алгоритмов. М.: Макс ПРЕСС, 2002. 82 с.
2. *Алексеев В. Б.* Лекции по дискретной математике. М.: Изд. отд. фак. ВМиК МГУ им. М. В. Ломоносова, 2004. 74 с.
3. *Андерсон Дж.* Дискретная математика и комбинаторика. М.: Вильямс, 2004. 960 с.
4. *Баранов С. И., Скляров В. А.* Цифровые устройства на программируемых БИС с матричной структурой. М.: Радио и связь, 1986. 272с.
5. *Белоусов А. И., Ткачев С. Б.* Дискретная математика: учеб. для вузов / под ред. В. С. Зарубина и А. П. Крищенко. 4-е изд. М.: Изд-во МГТУ им. Н. Э. Баумана, 2006. 743 с.
6. *Биркгоф Г.* Теория решеток. М.: Наука, 1984. 568 с.
7. *Боднарчук В. Г., Калужнин Л. А., Котов В. Н., Ромов Б. А.* Теория Галуа для алгебр Поста // Кибернетика. 1969. № 3. С. 1–10 (Киев); № 5. С. 1–9.
8. *Большакова Е. И., Груздева Н. В.* Программирование на языке Пролог. М.: МАКС Пресс, 2013. 112 с.
9. *Бордя Т. Д., Дружинин В. А., Васюткина И. А.* Дискретная математика. Тирасполь, 2001. 163 с.
10. *Бохман Д., Постхоф Х.* Двоичные динамические системы. М.: Энергоатомиздат, 1986. 400 с.
11. *Братко И.* Программирование на языке Пролог для искусственного интеллекта. М.: Мир, 1990. 560 с.
12. *Бухштаб А. А.* Теория чисел. М.: Просвещение, 1966. 384 с.
13. *Бюхи Дж. Р.* Слабая арифметика второго порядка и конечные автоматы // Кибернетический сборник. Вып. 8. 1964. С. 43–77.
14. *Бюхи Дж. Р.* О разрешающем методе для ограниченной арифметики второго порядка // Кибернетический сборник. Вып. 8. 1964. С. 78–90.
15. *Вагин В. Н., Головина Е. Ю., Загорянская А. А., Фомина М. В.* Достоверный и правдоподобный вывод в интеллектуальных системах. М.: Физматлит, 2004. 704с.
16. *Виноградов И. М.* Основы теории чисел. М.: Высшая школа, 1965. 172 с.
17. *Гаврилов Г. П., Сапоженко А. А.* Сборник задач по дискретной математике. М.: Наука, 1977. 368 с.
18. *Гильберт Д., Бернайс П.* Основания математики. Т. 1: Логические исчисления и формализация арифметики. М.: Наука, 1982. 557 с.; Т. 2: Теория доказательств. М.: Наука, 1982. 653 с.
19. *Горбатов В. А., Горбатов А. В., Горбатова М. В.* Дискретная математика: учеб. для студентов втузов. М.: АСТ; Астрель, 2003. 447 с.

20. Дискретная математика и математические вопросы кибернетики. Т. 1 / под ред. С. В. Яблонского и О. Б. Лупанова. М.: Наука, 1974. 312 с.
21. Ершов Ю. Л., Палютин Е. А. Математическая логика. М.: Наука, 1979. 320 с.
22. Журавлев Ю. И., Флеров Ю. А., Вялый М. Н. Дискретный анализ. Основы высшей алгебры. М.: МЗ Пресс, 2007. 224 с.
23. Клини С. К. Введение в метаматематику. М.: ИЛ, 1957. 526 с.
24. Клини С. К. Математическая логика. М.: МИР, 1973. 480 с.
25. Кудрявцев В. Б., Алешин С. В., Подколзин А. С. Введение в теорию автоматов. М.: Наука, 1985. 319 с.
26. Кузнецов О. П., Адельсон-Вельский Г. М. Дискретная математика для инженера. М.: Энергоатомиздат, 1988. 480 с.
27. Лупанов О. Б. О сравнении двух типов конечных источников // Проблемы кибернетики. Вып. 9. М.: Физматгиз, 1963. С. 321–326.
28. Лупанов О. Б. Асимптотические оценки сложности управляющих систем. М.: Изд-во МГУ, 1984. 137 с.
29. Лупанов О. Б. Введение в математическую логику. М.: МГУ Мехмат, 2007. 192 с.
30. Мендельсон Э. Введение в математическую логику. М.: Наука, 1984. 319 с.
31. Марченков С. С. Замкнутые классы булевых функций. М.: Физматгиз, 2000. 128 с.
32. Набебин А. А. Логика и Пролог в дискретной математике. М.: Изд-во Моск. энерг. ин-та, 1996. 452 с.
33. Набебин А. А., Кораблин Ю. П. Математическая логика и теория алгоритмов. М.: Научный мир, 2008. 343 с.
34. Набебин А. А. Сборник заданий по дискретной математике. М.: Научный мир, 2009. 280 с.
35. Непейвода Н. Н. Прикладная логика. Ижевск: Изд-во Удмуртского ун-та, 1997. 385 с.
36. Новиков П. С. Элементы математической логики. М.: Наука, 1973. 399 с.
37. Плиско В. Е., Хаханян В. Х. Интуиционистская логика. М.: Изд-во при мех.-мат. ф-те МГУ, 2009. 159 с.
38. Сикорский Р. Булевы алгебры. М.: МИР, 1969. 376 с.
39. Трахтенброт Б. А., Барздынь Я. М. Конечные автоматы. М.: Наука, 1970. 400 с.
40. Чень Ч., Ли Р. Математическая логика и автоматическое доказательство теорем. М.: Мир, 1983. 360 с.
41. Черч А. Введение в математическую логику. М.: ИЛ, 1960. 485 с.
42. Яблонский С. В. Введение в дискретную математику. М.: Наука, 1986. 384 с.
43. Яблонский С. В., Гаврилов Г. П., Набебин А. А. Предполные классы в многозначных логиках. М.: Издат-во МЭИ, 1997. 144 с.
44. Abstract State Machines, Alloy, B, TLA, VDM, and Z 5th International Conference, ABZ 2016. Linz. Austria. May 23–27. 2016. Proceedings.
45. Alagar V. S., Periyasamy K. Specification of software systems. Springer Verlag London Limited. 2011. 644 p.

46. Gödel K. Ueber formal unentscheidbare Sätze der Principia Mathematicae und verwandter Systeme I // Monatshefte fuer Mathematik und Physik. 1931. V. 38. P. 173–198.
47. Goguen Joseph A. Introducing OBJ // SRI CSL-88-9. SRI International. USA. 1988. www.kindsoftware.com/products/opensource/.
48. Kozen D. A completeness theorem for Kleene algebra and the algebra of regular events // Infor. and comput. 110:2 (May 1994), 366–390.
49. Kröger F. Temporal Logic of Programs. Springer Verlag, 1987. 148 p.
50. Lichtenstein O., Pnueli A. Propositional temporal logics: Decidability and completeness. L. J. of the IGPL. (2000). V. 8. № 1. P. 55–85.
51. McNaughton R. Testing and generating infinite sequences by a finite automaton // Information and Control. 9. № 5. 1966. P. 521–530.
52. Mukund M. Linear-Time Temporal Logic and Buchi Automata // Winter School on Logic & Computer Science ISI Calcutta. Dec. 1996 – Jan. 1997.
53. Peano G. Arithmetices principia, nova methodo exposita. Turin (Bocca), 1889. xvi + 20 p.
54. Software engineering with OBJ: algebraic specification in action / ed. by J. Goguen, G. Malcolm. Springer Science + Business Media, LLC, 2000. 428 p.



Обозначения

$=, \neq$ – знаки равенства и неравенства.

$\leq, <$ – знаки нестрогого и строгого неравенств.

$a = b$ – элемент a равен элементу b .

$a \neq b$ – элемент a не равен элементу b .

$\in, \notin$ – знаки принадлежности и непринадлежности.

$a \in A$ – элемент a принадлежит множеству A .

$a \notin A$ – элемент a не принадлежит множеству A .

$\subseteq, \supseteq$ – знаки нестрогого включения множеств.

$A \subseteq B$ – A включается в B ; A подмножество B ; B надмножество A .

$A \supseteq B$ – A включает B ; B подмножество A ; A надмножество B .

$A = B$ – множество A равно множеству B .

$\subset, \supset$ – знаки строгого включения множеств.

$A \subset B$ – A строго включается в B .

$A \supset B$ – A строго включает B .

$\emptyset$ – пустое множество.

U – универсальное множество (универсум).

$P(A), 2^A$ – множество всех подмножеств множества A .

$|A|, \text{card}(A), c(A)$ – мощность множества A .

$A \cup B$ – объединение множеств A и B .

$A \cap B$ – пересечение множеств A и B .

$A - B$ – разность множеств A и B .

$A \div B$ – симметрическая разность множеств A и B .

$\overline{A} = U - A$ – дополнение множества A до универсума U .

$A \times B$ – декартово произведение множеств A и B .

$A_1 \times A_2 \times \dots \times A_n$ – декартово произведение n множеств.

A^n – натуральная степень множества A .

$\mathbb{N}$ – множество натуральных чисел.

$\mathbb{Z}$ – множество целых чисел.

$\mathbb{Q}$ – множество рациональных чисел.

$\mathbb{R}$ – множество вещественных чисел.

$\mathbb{C}$ – множество комплексных чисел.

$f: A \rightarrow B$ – f есть функция из множества A во множество B .

$f(x)$ – f есть функция переменной x .

$D(f)$ – область определения функции f .

$R(f), \text{Im}(f), f(A)$ – область значений функции f .

$f^{-1}(b)$ – полный прообраз элемента b .

$f^{-1}(C)$ – полный прообраз множества C .

$g \circ f, gf, g(f(x))$ – композиция функций f и g .



$\rho^n \subseteq A_1 \times A_2 \times \dots \times A_n$ – n -арное отношение.

$\rho(x, \dots, x_n), (x_1, \dots, x_n) \in \rho$ – элементы $x_1, \dots, x_n$ находятся в отношении ρ .

$\sigma \subseteq A \times A$ – бинарное отношение.

$a \sigma b, \sigma(a, b), (a, b) \in \sigma$, – a находится в отношении σ к b .

$\sim$ – знак отношения эквивалентности.

$[a]_\sigma$ – множество всех элементов, эквивалентных элементу a по отношению σ .

A/σ – фактор-множество A по эквивалентности σ .

$f: A^n \rightarrow A$ – n -арная (n -местная) операция (функция) на A .

$n(f)$ – арность (местность) операции f .

f^n – n -местная функция.

C_n^k – число сочетаний из n по k .

$E_2 = \{0, 1\}$.

E_2^n – множество всех наборов длины n из 0 и 1.

P_2 – множество всех функций алгебры логики.

$E_k = \{0, 1, \dots, k-1\}$ – множество всех натуральных чисел, меньших k .

E_k^n – множество всех наборов длины n с элементами из E_k .

P_k – множество всех функций k -значной логики.

$[A]$ – замыкание множества A функций относительно суперпозиции.

$\&, \vee$ – знаки конъюнкции и дизъюнкции.

$\neg, \neg$ – знаки отрицания.

$\rightarrow$ – знак импликации.

$\equiv$ – знак логической эквивалентности.

$|$ – штрих Шеффера.

$\uparrow$ – стрелка Пирса.

$\leftrightarrow$ – знак «тогда и только тогда», «если и только если».

$\Leftrightarrow$ – знак равносильности.

$\forall, \exists$ – кванторы общности и существования.

$\exists!$ – квантор существования единственного элемента.

$\frac{\partial f}{\partial x_i}, f'_{x_i}$ – производная первого порядка от функции f по переменной x_i .

$\frac{\partial^k f}{\partial x_{i_1} \partial x_{i_2} \dots \partial x_{i_k}}, f_{x_{i_1} x_{i_2} \dots x_{i_k}}^k$ – смешанная производная порядка k от функции $f(x_1, x_2, \dots, x_n)$ по переменным $x_{i_1}, x_{i_2}, \dots, x_{i_k}$.

$\frac{\partial f^{(k)}}{\partial (x_{i_1}, x_{i_2}, \dots, x_{i_k})}, f_{(x_{i_1}, x_{i_2}, \dots, x_{i_k})}^{(k)}$ – производная (Бохмана) от функции $f(x_1, x_2, \dots, x_n)$ по направлению $x_{i_1}, x_{i_2}, \dots, x_{i_k}$.

$p \left(\frac{\partial^{(k)} f}{\partial (x_{i_1}, x_{i_2}, \dots, x_{i_k})} \right)$ – вес производной.

$\sum_{i=1}^n, \sum_{i=1}^n$ – сумма по i от 1 до n .



$\prod_{i=1}^n, \prod_{i=1}^n$ – произведение по i от 1 до n .

$\vdash$ – знак выводимости и доказуемости в ИВ и ИП.

$\vdash_L$ – знак выводимости и доказуемости в исчислении L .

$\Rightarrow$ – знак секвенциальной выводимости.

$[P]$ – знак для множества истинности предиката P .

$\models$ – знак выполнимости.

$\square$ – пустой (ложный) дизъюнкт; знак окончания теоремы, леммы и т. д.

A2 – аксиома A2.

AB – аксиома Бернаиса.

AP – аксиома равенства.

БД – база данных.

ДНФ – дизъюнктивная нормальная форма.

И – истина.

ИВ – исчисление высказываний.

ИИВ – интуиционистское исчисление высказываний.

ИП – исчисление предикатов.

КНФ – конъюнктивная нормальная форма.

Л – ложь.

ЛП – логика предикатов.

ЛОП – логика одноместных предикатов.

МДНФ – минимальная дизъюнктивная нормальная форма.

МКНФ – минимальная конъюнктивная нормальная форма.

МНК – монадическая логика натуральных чисел.

НОУ – наиболее общий унификатор.

НОУ(W) – наиболее общий унификатор для множества литер W .

$P_r^B(A)$ – подстановка формулы B на место переменной r в формуле A .

$\Pi_{p_1 p_2 \dots p_k}^{B_1 B_2 \dots B_k}(A)$ – результат одновременного замещения переменных $p_1, p_2, \dots, p_k$ в формуле A на формулы $B_1, B_2, \dots, B_k$ соответственно.

ПВ – правило вывода.

ПВ4(1,2) – правило вывода 4 применяется к формулам 1 и 2.

ПЗ – правило заключения.

ПИ – простой импликант.

ПН – (модальное) правило необходимости.

ПР – правило резолюции.

ПР(6,9) – правило резолюции применяется к формулам 6 и 9.

ПТЛ – пропозициональная темпоральная логика.

ПЭ – правило эквивалентности.

СДНФ – совершенная дизъюнктивная нормальная форма.

СИВ – секвенциальное исчисление высказываний.

СИП – секвенциальное исчисление предикатов.

СКНФ – совершенная конъюнктивная нормальная форма.

ТЗ – теорема 3.

ТД – теорема дедукции.

ТДНФ – тупиковая дизъюнктивная нормальная форма.

ИП – исчисление предикатов.

$U(W)$ – унификатор для множества литер W .

$U(3,7)$ – унификатор формул 3 и 7.

$\forall$ -ПР (или $\forall$ -правило) – правило навешивания квантора общности.

$\forall$ -ПР(4) – $\forall$ -правило применяется к формуле (4).

$\exists$ -ПР (или $\exists$ -правило) – правило навешивания квантора существования.

$\exists$ -ПР(5) – $\exists$ -правило применяется к формуле 5.

$\bigcirc$ – темпоральный символ следующий (Next).

$\diamond$ – темпоральный символ существует (Eventually).

$\square$ – темпоральный символ всегда (Always).

$\mathcal{U}$ – темпоральный символ до тех пор, пока не (Until).

$\mathcal{W}$ – темпоральный символ пока (While, Unless).

$\mathcal{S}$ – темпоральный символ с тех пор, как (Since).

$\mathcal{R}$ – темпоральный символ отпускает, освобождает (Release).



Книги издательства «ДМК Пресс» можно заказать
в торгово-издательском холдинге «Планета Альянс» наложенным платежом,
выслав открытку или письмо по почтовому адресу:
115487, г. Москва, 2-й Нагатинский пр-д, д. 6А.
При оформлении заказа следует указать адрес (полностью),
по которому должны быть высланы книги;
фамилию, имя и отчество получателя.
Желательно также указать свой телефон и электронный адрес.
Эти книги вы можете заказать и в интернет-магазине: www.aliants-kniga.ru.
Оптовые закупки: тел. **(499) 782-38-89**.
Электронный адрес: books@aliants-kniga.ru.



Авдошин Сергей Михайлович,
Набебин Алексей Александрович



Дискретная математика.
Формально-логические системы и языки

Главный редактор *Мовчан Д. А.*
dmkpress@gmail.com
Корректор *Синяева Г. И.*
Верстка *Чаннова А. А.*
Дизайн обложки *Мовчан А. Г.*

Формат 70×100 1/16.
Гарнитура «PT Serif». Печать офсетная.
Усл. печ. л. 33. Тираж 200 экз.

Веб-сайт издательства: www.dmk.ru