МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Государственное образовательное учреждение
высшего профессионального образования
«Южно-Российский государственный университет экономики и сервиса»
(ГОУ ВПО «ЮРГУЭС»)

ЭЛЕМЕНТЫ МАТЕМАТИЧЕСКОЙ ЛОГИКИ И ТЕОРИИ АЛГОРИТМОВ

Учебно-методическое пособие
для студентов специальностей:
230201.65 «Информационные системы и технологии»,
080801.65 «Прикладная информатика (в сфере сервиса)»,
направления подготовки бакалавров 230220
«Информационные системы»

ШАХТЫ
ГОУ ВПО «ЮРГУЭС»
2011

УДК 510(07)
ББК 22.12я73
Э456

Рекомендовано к внутривузовскому изданию
редакционно-издательским советом ЮРГУЭС

Составители:
д.ф.-м.н., профессор кафедры «Математика» ГОУ ВПО «ЮРГУЭС»
В.Г. Фетисов
к.ф.-м.н., доцент кафедры «Математика» ГОУ ВПО «ЮРГУЭС»
В.И. Филиппенко

Рецензенты:
д.т.н., директор НПФ «Электронные информационные системы»
ГОУ ВПО «ЮРГУЭС»
А.А. Сапронов
к.ф.-м.н., директор ФГУ ИМЦА
И.М. Мальцев

Э456 Элементы математической логики и теории алгоритмов : учеб.-метод. пособие / составители В.Г. Фетисов, В.И. Филиппенко. – Шахты : ГОУ ВПО «ЮРГУЭС», 2011. – 66 с.

Учебно-методическое пособие охватывает традиционные разделы математической логики и теории алгоритмов. Значительное место в пособии занимает описание методов, наиболее часто применяемых на практике при решении задач математической логики и теории алгоритмов. В пособии рассмотрено достаточное количество практических задач, иллюстрирующих теоретический материал.

Рекомендовано для студентов ЮРГУЭС, обучающихся по специальностям: 230201.65 «Информационные системы и технологии», 080801.65 «Прикладная информатика (в сфере сервиса)», направление подготовки бакалавров 2302200 «Информационные системы».

УДК 510(07)
ББК 22.12я73

Режим доступа к электронному аналогу печатного издания:
http://www.libdb.sssu.ru

© ГОУ ВПО «Южно-Российский государственный
университет экономики и сервиса», 2011

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Логика как наука об особенностях и формах мышления имеет древнюю историю, истоки которой берут начало с творчества таких античных философов, как Аристотель (384–322 до н. э.), Зенон (336–264 до н. э.), Демокрит (460–370 до н. э.). Изучая структуру и формы человеческой мысли, логика отвлекается от конкретного содержания, которое может заключать в себе те или иные формы мысли. Тем самым логика абстрагируется от реальных объектов, предметов и явлений, которые являются элементами мыслительного процесса.

Начиная с античных философов, предметом логики принято считать изучение структуры и взаимосвязей форм, в рамках которых реализуется процесс мышления. Разработка формально-логических средств анализа оснований математики привела к появлению математической логики. Идея построения универсального символического языка для математики и формализации на его основе процесса математического доказательства теорем принадлежит немецкому математику и философу Г. Лейбницу (1646–1716). Большой вклад в начальный этап разработки аппарата математической логики внесли Дж. Буль (1815–1864), Г. Фреге (1848–1925), Дж. Пеано (1858–1932), Н.И. Лобачевский (1792–1856). Именно в этот период были сформулированы принципы аксиоматического метода в математике, а также разработаны основы логики высказываний и логики предикатов.

Важнейшим событием не только в истории логики, но и в науке в целом стала программа обоснования математики на базе математической логики, с которой выступил на II Математическом Конгрессе математиков в 1900 г. немецкий математик Д. Гильберт (1862–1943). Сформулированные им проблемы математики предопределили развитие некоторых направлений фундаментальной математики и математической логики. Именно этот период принято считать началом современного этапа развития математической логики, характерной особенностью которого является применение строгих математических средств изучения формальных аксиоматических теорий. Наибольший вклад в разработку математической логики внесли К. Гедель (1906–1978), А. Черч (1903–1995), А.М. Тьюринг (1912–1954), П.С. Новиков (1901–1975), Ф.И. Мальцев (1909–1967), А.А. Марков (1903–1979).

Отличие современной формальной логики от классической заключается не в предмете исследования, а в применяемых методах, которые основаны на математических средствах аксиоматизации и фор-

мализации точных рассуждений. Именно эти методы способствовали интенсивному развитию математической логики, в рамках которой были решены многие проблемы, такие как установление и доказательство полноты, непротиворечивости и разрешимости формально-логических теорий.

На развитие математической логики в последние десятилетия оказали влияние две основные тенденции. С одной стороны, инженерная разработка вычислительных устройств потребовала развития математических средств моделирования и проектирования соответствующих технических устройств, что послужило стимулом к исследованию различных логико-алгебраических систем. С другой – интенсивное развитие проблематики искусственного интеллекта привело не только к разработке логико-лингвистических моделей представления знаний, но и к созданию неклассических логик, которые специально предназначены для моделирования тех или иных аспектов мышления. В этом контексте нечёткая логика представляет собой вариант неклассической логики.

Исторически тремя базовыми конструкциями формальной логики являются понятие, суждение и умозаключение.

Понятие – это форма мышления, в которой отражаются существенные признаки одноэлементного класса или класса однородных предметов. Содержанием понятия называется совокупность существенных признаков одноэлементного класса или класса однородных предметов, отражённых в этом понятии. Объёмом понятия называют совокупность (класс) предметов, которая мыслится в этом понятии.

Суждение – форма мышления, в которой что-либо утверждается или отрицается о существовании предметов, связях между предметом и его свойствами или об отношениях между предметами.

Умозаключение – форма мышления, в которой из одного или нескольких суждений на основании определённых правил вывода получается новое суждение, с необходимостью или определённой степенью вероятности следующее из них. Умозаключения делятся на такие виды: дедуктивные, индуктивные, по аналогии. Выведение следствий из данных посылок – широко распространённая логическая операция. Формальная логика знакомит с правилами различных видов умозаключений. Математическая логика даёт формальный аппарат, с помощью которого в определённых частях логики можно выводить следствия из данных посылок.

Дедуктивные умозаключения есть умозаключения, у которых между посылками и заключением имеется отношение логического следования. Индукцией называется умозаключение от знания меньшей степени общности к новому знанию большей степени общности (т.е. от отдельных частных случаев мы переходим к общему суждению). При втором подходе, присущем современной математической логике, индукцией называется умозаключение, дающее вероятностное суждение. Аналогия – умозаключение о принадлежности предмету определённого признака (т.е. свойства или отношения) на основе сходства в признаках с другим предметом.

В математической логике главным образом исследуются формы взаимосвязей между логическими высказываниями или предложениями. Существенное значение при этом имеют используемые для этих целей символические обозначения или синтаксис формально-логических теорий, а также логические формулы, которые могут быть построены на основе высказываний, предикатных и функциональных символов, логических связок и кванторов.

Математическая логика (теоретическая логика, символическая логика) – раздел математики, изучающий доказательства и вопросы оснований математики.

Применение в логике математических методов становится возможным тогда, когда суждения формулируются на некотором точном языке. Такие точные языки имеют две стороны: синтаксис и семантику. Синтаксисом называется совокупность правил построения объектов языка (обычно называемых формулами). Семантикой называется совокупность соглашений, описывающих наше понимание формул (или некоторых из них) и позволяющих считать одни формулы верными, а другие – нет.

Важную роль в математической логике играют понятия дедуктивной теории и исчисления. Исчислением называется совокупность правил вывода, позволяющих считать некоторые формулы выводимыми.

Отметим, что на практике множество элементарных логических операций является обязательной частью набора инструкций всех современных микропроцессоров и соответственно входит в языки программирования. Это является одним из важнейших практических приложений методов математической логики.

1. ЛОГИКА ВЫСКАЗЫВАНИЙ

1.1. Понятие высказывания

Рассмотрим логику высказываний, которая лежит в основе всех других разделов математической логики (МЛ) и необходима для их понимания.

Логика высказываний строится так же, как и другие математические теории. В качестве основных понятий берётся некоторый класс объектов, а также некоторые свойства, отношения и операции над этими объектами.

Основным объектом логики высказываний служат простые высказывания.

Высказыванием (элементарным высказыванием) в математической логике называется повествовательное предложение, выражающее законченную мысль, относительно которой его можно считать истинным или ложным, но не то и другое одновременно.

Например, утверждение «Москва – столица России» является истинным высказыванием, а «Пингвины живут в Африке» – ложное высказывание. Вместе с тем, утверждение «В созвездии Кассиопеи существует жизнь», в соответствии с приведённым определением, высказыванием не является, т.к. мы не можем в настоящее время судить о его истинности или ложности.

Высказывания, не допускающие расчленения на более простые, называются элементарными. Для обозначения элементарных высказываний принято использовать символы x, y, z, p, q, r, s, t, которые в логике называют пропозициональными переменными. Составные высказывания строятся из элементарных с использованием логических связок.

Важной особенностью логики высказываний является то, что само элементарное высказывание не подлежит какому-либо синтаксическому или лингвистическому анализу. Другими словами, ответ на вопрос: «Почему то или иное элементарное высказывание является истинным или ложным?» лежит за пределами логики и может служить предметом различных специальных исследований.

Примеры 1.1.1.

1. Число 100 делится на 5.
2. Число 3 больше числа 5.
3. Луна больше Земли.

4. Сегодня светит солнце.

5. Вечером мы пойдём в кино.

Из простых высказываний с помощью некоторого числа логических операций можно построить сложные высказывания.

6. Число 100 делится на 5 и число 100 делится на 10.

7. Неверно, что 3 больше 5.

8. Сегодня мы пойдём в кино или мы пойдём в театр.

При изучении логики высказываний несущественно содержание простых высказываний, а интересуются только их истинностью или ложностью.

Сложные высказывания, получаемые из простых высказываний, будут также истинными или ложными. Их истинность или ложность будет зависеть от истинности образующих их простых высказываний.

1.2. Основные логические операции над высказываниями (логические связки)

Для изучения логических операций введём следующую систему обозначений:

- простые высказывания будем обозначать буквами $a, b, c, ..., x, y, z$;
- значения истинности будем обозначать: 1 – истинно, 0 – ложно.

Действия логических операций будем представлять в виде таблиц истинности.

Отрицание или инверсия **($\bar{x}$, *–x, НЕ*).** Это унарная логическая операция над элементарными высказываниями, результат которой является также высказыванием и принимает значение «истина», если исходное высказывание ложно и значение «ложь», если исходное высказывание истинно.

Примеры 1.2.1.

1. Высказывание x = «7 делится на 5 без остатка».

2. Высказывание $\bar{x}$ = «Неверно, что 7 делится на 5 без остатка».

Конъюнкция **($\wedge$, *&, ·, логическое И*).** Это бинарная операция над двумя высказываниями, результат которой равен значению «истина», если истинны оба высказывания одновременно и значение «ложь» во всех остальных случаях.

Высказывание $a \wedge b$ читается «a и b». Логическую конъюнкцию также называют логическим «и».

Примеры 1.2.2.

Если высказывание: x = «6 делится на 3 без остатка» (1);

y =«10 больше 5» (1);

u = «7 делится на 3 без остатка» (0);

v = «3 больше 7» (0).

Значения истинности этих высказываний следующие: $x = 1,\ y = 1,\ u = 0,\ v = 0$. Тогда $x \& y = 1,\ x \& u = 0,\ u \& v = 0$.

***Дизъюнкция* (∨,+, *логическое ИЛИ*).** Это бинарная операция над двумя высказываниями, результат которой равен значению «ложь», если ложны оба высказывания одновременно и значение «истина» во всех остальных случаях.

Высказывание: $a \vee b$ читается – «a или b». Логическую дизъюнкцию также называют логическим неисключающим «или».

Примеры 1.2.3.

Для высказываний, рассмотренных в предыдущих примерах, имеем $x \vee y = 1,\ x \vee u = 1,\ u \vee v = 0$.

***Импликация* (→, ⊃, *“если a, то b”*).** Бинарная операция над двумя высказываниями, результат которой принимает значение «ложь» в случае истинности высказывания a и ложности высказывания bи значение «истина» во всех остальных случаях. Высказывание: $a \to b$ читается – «a влечёт b», «из a следует b», «если a, то b».

***Эквивалентность* (~, ↔).** Это бинарная операция над двумя высказываниями, которая принимает значение «истина» в случае истинности или ложности обоих высказываний одновременно и значение «ложь» во всех остальных случаях. Высказывание $a \sim b$ читается – «a эквивалентно b».

Эквивалентность соответствует употреблению выражения «тогда и только тогда».

Штрих Шеффера (|, антиконъюнкция). Это бинарная операция, которая принимает значение «ложь» лишь в том случае, когда оба высказывания принимают значение «истина», и значение «истина» в остальных случаях. Иногда эту операцию называют «не и» (так как она равна отрицанию конъюнкции.

Стрелка Пирса (↓, антидизъюнкция). Это бинарная операция, которая принимает значение «истина» лишь в том случае, когда оба высказывания принимают значение «ложь», и значение «ложь» в остальных случаях. Эта функция является отрицанием дизъюнкции и поэтому её называют «не или».

Сумма по модулю два **(**$\oplus$ ***или антиэквиваленция*****).** Это бинарная операция, которая принимает значение «ложь» лишь в том случае, когда оба высказывания принимают одинаковые значение, и значение «истина» в остальных случаях.

Семантика формул определяется их свойством быть истинными или ложными, т.е. принимать значение 1 – истина, или 0 – ложь. Интерпретировать формулу логики высказываний – значит приписать значения истинности составляющим её элементарным высказываниям и вычислить истинностное значение всей формулы. При этом логические связки рассматриваются как функции на множестве истинностных значений (булевы функции). Интерпретация, при которой формула принимает значение 1, называется моделью этой формулы. Формула выполнима (непротиворечива), если она допускает хотя бы одну модель, т.е. её можно интерпретировать со значением 1.

В математической логике принято определять логические операции с помощью таблиц истинности, которые определяют истинность результата рассматриваемой операции для каждого из значений истинности исходного или исходных операций.

Таблица 1.2.1

Таблица истинности основных бинарных логических операций

x	y	$(x\&y)$	$(x\vee y)$	$(x\to y)$	$(x\leftrightarrow y)$	$(x\mid y)$	$(x\downarrow y)$	$(x\oplus y)$
0	0	0	0	1	1	1	1	0
0	1	0	1	1	0	1	0	1
1	0	0	1	0	0	1	0	1
1	1	1	1	1	1	0	0	0

Символы $\neg$, $\&$, $\vee$, $\to$, $\leftrightarrow$, $\oplus$, $\mid$, $\downarrow$ называются пропозициональными связками, a, b, c,... и т.д. – пропозициональными переменными. Выражение, построенное из пропозициональных переменных с помощью пропозициональных связок, называется пропозициональной формой или формулой.

Используя эти логические операции, можно строить сколь угодно сложные высказывания.

Приоритет выполнения операций следующий:

$\neg$, $\&$, $\vee$, $\to$, $\leftrightarrow$, $\oplus$, $\mid$, $\downarrow$.

Примеры 1.2.4

1. Сложное высказывание: «Если вы не пропускаете занятия и успешно занимаетесь, то Вы сдадите экзамен хорошо» можно записать следующим образом. Обозначим:

p – пропускаете занятия;

q – успешно занимаетесь;

r – сдадите экзамен хорошо, тогда всё высказывание запишется так: $\overline{p}\,\&\,q \to r$.

2. Записать следующее утверждение в виде формулы логики высказываний и определить выполнимость, общезначимость и количество моделей этой формулы:

«Студент не допускается к экзамену тогда и только тогда, когда у него не сдана курсовая работа или есть задолженности по лабораторным работам».

Решение. Обозначим все встретившиеся элементарные высказывания пропозициональными переменными:

p – «студент допущен к экзамену»;

q – «курсовая работа сдана»;

r – «есть задолженность по лабораторным работам».

Тогда формула запишется в следующем виде: $\neg p \leftrightarrow (\neg q \vee r)$. Построим таблицу истинности, вычисляя сначала значения подформул.

p	q	r	$\neg p$	$\neg q$	$\neg q \vee r$	$\neg p \leftrightarrow (\neg q \vee r)$
0	0	0	1	1	1	1
0	0	1	1	1	1	1
0	1	0	1	0	0	0
0	1	1	1	0	1	1
1	0	0	0	1	1	0
1	0	1	0	1	1	0
1	1	0	0	0	0	1
1	1	1	0	0	1	0

Таким образом, эта формула выполнима, т.к. имеет модели и не общезначима, т.к. имеются интерпретации, при которых она ложна. Формула имеет четыре модели.

Значение истинности всего выражения будет зависеть от истинности переменных, обозначающих простые высказывания.

1.3. Булевы функции

Сначала определим понятие бинарного отношения между множествами. Действительно, множество – фундаментальное неопределяемое понятие, представляющее собой совокупность объектов, которые, с одной стороны, различны и отличимы друг от друга, а с другой – воспринимаются как единое целое.

Пусть A и B – два множества. $\langle a,b\rangle$ – упорядоченная пара, где первый элемент $a \in A$, а второй элемент $b \in B$.

Декартово произведение $A \times B$ – это множество пар $A \times B = \{< a,b > | \, a \in A, b \in B\}$. Декартово произведение n экземпляров множества A обозначают через A^n, т.е. $A \times A \times ... \times A = A^n$.

Бинарным отношением f из множества A в множество B называется подмножество $A \times B$: $f \subset A \times B$.

Функция – это такое бинарное отношение, для которого выполняется следующее условие: что из $< x, y > \in f$ и $< x, z > \in f$ следует, что $x = z$, т.е. функциональность – это однозначность.

Пример 1.3.1.

Пусть $A = \{1,2,3,4,5\}, B = \{1,4,9,16,25\}$.

Тогда $A \times B = \{\langle 1,1\rangle, \langle 1,4\rangle, \langle 1,9\rangle, \langle 1,16\rangle, \langle 1,25\rangle, \langle 2,1\rangle, \langle 2,4\rangle, \langle 2,9\rangle, \langle 2,16\rangle, \langle 2,25\rangle, ..., (3,9), ..., \langle 5,25\rangle\}$.

Бинарное отношение $f = \{\langle 1,1\rangle, \langle 2,4\rangle, \langle 3,9\rangle, \langle 4,16\rangle, \langle 5,25\rangle\}$ есть функция, где $b = a^2$.

Функция $f : E_2^n \to E_2$, где $E_2 = \{0,1\}$, а $E_2^n = E_2 \times E_2 \times \cdots \times E_2$ называется функцией алгебры логики.

Пример функции алгебры логики можно составить используя, рассмотренные выше логические символы: $((ab \to c) \vee \overline{a}) \sim ac = f(a,b,c)$.

Таким образом, каждое элементарное высказывание может принимать значение либо 0, либо 1. Каждому набору значений *a*, *b*, *c* соответствует одно значение всего сложного высказывания (0 или 1).

Булеву функцию от *n* переменных можно задать таблицей истинности:

x_1	…	x_{n-1}	x_n	$f(x_1,...,x_n)$
0	…	0	0	$f\ (0,...,0)$
0	…	0	1	$f(0,...,1)$
…	…	…	…	…
1		1	1	$f(1,...,1)$

Переменные, которые принимают значения 0 или 1, называются булевыми переменными.

Некоторые функции всегда принимают значение 1 (на любом наборе переменных). Такие функции называются тавтологиями. Некоторые функции всегда принимают значение 0 (на любом наборе переменных). Такие функции называются противоречиями.

1.4. Формулы

Пусть $F = \{f_1, f_2, ... f_n\}$ – множество булевых функций. Формулой над F называется выражение $\Im[F] = f(t_1, ..., t_m)$, где $f \in F$, $t_i, i = 1,2,\cdots,n$ – либо переменная, либо формула над F. F называется базисом формулы, f – главной (внешней) операцией, t_i – подформулами. Всякой формуле $\Im$ однозначно соответствует некоторая функция f. Это соответствие задаётся алгоритмом интерпретации, который позволяет вычислить значение функции при заданных значениях переменных.

Зная таблицы истинности для функций базиса, можно вычислить таблицу той функции, которую реализует данная формула.

Примеры 1.4.1.

1. $F_1 = (x_1 \,\&\, x_2) \vee (x_1 \,\&\, \bar{x}_2) \vee (\bar{x}_1 \,\&\, x_2)$.

x_1	x_2	$x_1 x_2$	$x_1 \overline{x_2}$	$\overline{x_1} x_2$	$\Im_1$
0	0	0	0	0	0
0	1	0	0	1	1
1	0	0	1	0	1
1	1	1	0	0	1

Функция F_1 реализует дизъюнкцию на базисе $\{\neg, \&, \mathrm{V}\}$.

2. $F_2 = (x_1 \sim x_2) \to (\bar{x}_1 \,\&\, x_2)$.

x_1	x_2	$x_1 \sim x_2$	$\overline{x_1} x_2$	$\Im_2$
0	0	1	0	0
0	1	0	1	1
1	0	0	0	1
1	1	1	0	0

Функция F_2 реализует антиэквиваленцию на базисе $\{\neg, \&, \to\}$.

3. $F_3 = ((x \to y) \& ((\bar{y} \to z) \to \bar{x}))$.

Будем строить таблицу истинности последовательно в соответствии с шагами построения формулы F_3:

x	y	z	$(x \to y)$	$(\bar{y} \to z)$	$((\bar{y} \to z) \to \bar{x})$	F_3
0	0	0	1	0	1	1
0	0	1	1	1	1	1
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	0	0	1	0
1	0	1	0	1	0	0
1	1	0	1	1	0	0
1	1	1	1	1	0	0

Функция F_3 реализует отрицание.

Таким образом, каждая формула определяет некоторую логическую функцию, которую можно представить в виде таблицы истинности для этой формулы. Если в формуле имеется n переменных, то возможны 2^n различных истинностных значений для этой формулы. Следовательно, таблица истинности будет иметь 2^n строк.

1.5. Простейшие эквивалентности

Формулы φ и ψ называются эквивалентными $(\varphi \sim \psi)$, если совпадают их таблицы истинности. Иными словами, формулы эквивалентны, если они реализуют равные функции.

Отметим основные эквивалентности между формулами:

1. $((\varphi \& \psi) \& x) \sim (\varphi \& (\psi \& x))$, $((\varphi \vee \psi) \vee x) \sim (\varphi \vee (\psi \vee x))$ (ассоциативность конъюнкции и дизъюнкции);
2. $(\varphi \& \psi) \sim (\psi \& \varphi)$, $(\varphi \vee \psi) \sim (\psi \vee \varphi)$ (коммутативность конъюнкции и дизъюнкции);
3. $(\varphi \& \varphi) \sim \varphi$, $(\varphi \vee \varphi) \sim \varphi$ (идемпотентность конъюнкции и дизъюнкции);
4. $(\varphi \& (\psi \vee x)) \sim ((\varphi \& \psi) \vee (\varphi \& x))$, $(\varphi \vee (\psi \& x)) \sim (\varphi \vee \psi) \& (\varphi \vee x))$ (законы дистрибутивности).
5. $(\varphi \& (\varphi \vee \psi)) \sim \varphi$, $(\varphi \vee (\varphi \& \psi)) \sim \varphi$ (законы поглощения);
6. $\neg(\varphi \& \psi) \sim \neg\varphi \vee \neg\psi$, $\neg(\varphi \vee \psi) \sim \neg\varphi \& \neg\psi$ (законы де Моргана);
7. $\neg\neg\varphi \sim \varphi$ (закон двойного отрицания).

Следующие эквивалентности показывают, что все основные логические операции могут быть выражены через операции конъюнкции, дизъюнкции и отрицания:

1. $(A \to B) = (\neg A \vee B) = \neg(A \& \neg B)$.

2. $A \sim B = (A \to B) \& (B \to A) = (A \& B) \vee (\neg A \& \neg B) = (A \vee \neg B) \& (\neg A \vee B)$.

3. $A \oplus B = (A \vee \neg B) \vee (\neg A \& B)$.

4. $A \downarrow B = \neg(A \vee B) = \neg A \& \neg B$.

5. $A \mid B = \neg(A \& B) \neg A \vee \neg B$.

Используя метод математической индукции, нетрудно получить также следующие равносильности (обобщённые законы дистрибутивности):

6. $(A_1 \vee A_2 \vee \ldots \vee A_n) \& (B_1 \vee B_2 \vee \ldots \vee B_m) =$
$A_1 \& B_1 \vee A_1 \& B_2 \vee \ldots \vee A_1 \& B_m \vee \ldots \vee A_n \& B_1 \vee A_n \& B_2 \vee \ldots \vee A_n \& B_m$.

7. $(A_1 \& A_2 \& \ldots \& A_n) \vee (B_1 \& B_2 \& \ldots \& B_m) =$
$(A_1 \vee B_1) \& (A_1 \vee B_2) \& \ldots \& (A_1 \vee B_m) \& \ldots \& (A_n \vee B_1) \& (A_n \vee B_2) \& \ldots \& (A_n \vee B_m)$.

Используя метод математической индукции, можно получить также следующие равносильности (обобщённые законы де Моргана):

8. $\neg(A_1 \& A_2 \& \ldots \& A_n) = \neg A_1 \vee \neg A_2 \vee \ldots \vee \neg A_n$.

9. $\neg (A_1 \vee A_2 \vee \ldots \vee A_n) = \neg A_1 \& \neg A_2 \& \ldots \& \neg A_n$.

Имеет место следующее правило равносильных преобразований: пусть для формул A и B справедливо утверждение $A = B$. Пусть C_A – формула, содержащая A в качестве своей подформулы. Пусть C_B получается из C_A заменой A на B. Тогда $C_A = C_B$.

Пример 1.5.1.

Пусть $A = x \supset y$, $B = \neg x \vee y$.

Используя таблицы истинности, можно убедиться в том, что $A = B$.

Пусть $C_A = (x \supset y) \& z$, т.е. A есть подформула C_A. Тогда $C_B = (\neg x \vee y) \& z$ и $C_A = C_B$, т.е. $(x \supset y) \& z \sim (\neg x \vee y) \& z$.

Пример 1.5.2.

Упростить формулу: $f = x \vee xy \vee yz \vee \bar{x}z$.

Для решения этой задачи используем основные равносильности алгебры логики.

$f = x \vee xy \vee yz \vee \bar{x}z = x \vee yz \vee \bar{x}z$ (закон поглощения)=

$= (x \vee \bar{x})(x \vee z) \vee yz$ (коммутативность и закон дистрибутивности) =

$= x \vee z \vee yz$ (свойства дополнения и идемпотентность) =

$= x \vee z$ (закон поглощения).

Символы &, V называются *двойственными*. Формула A^* называется *двойственной* формуле A, если она получена из A одновременной заменой всех символов &, V на двойственные.

Например, если $A = x \vee (y \& \neg z)$, то $A^* = x \,\&\, (y \vee \neg z)$.

Теорема 1.5.1. (Принцип двойственности). Если $A \sim B$, то $A^* \sim B^*$.

Принцип двойственности можно использовать для нахождения новых равносильностей. Например, для 1-го закона поглощения имеем: $A\&(A \vee B) \sim A$. Следуя принципу двойственности, получим новую равносильность:

$A \vee (A\&B) \sim A$ (2-й закон поглощения).

Как известно, *алгеброй* называют систему, включающую в себя некоторое непустое множество объектов с заданными на нём функциями (операциями), результатами применения которых к объектам данного множества являются объекты того же множества.

Булевой алгеброй, или *алгеброй логики,* называется двухэлементное множество $B = \{0, 1\}$ вместе с операциями конъюнкции, дизъюнкции и отрицания.

Система булевых функций $\{f_1, f_2, \ldots, f_n\}$ называется полной, если любая булева функция может быть выражена в виде суперпозиции этих функций. Из рассмотренных выше равносильностей следует, что все логические операции могут быть выражены через операции конъюнкции, дизъюнкции и отрицания. Поэтому система функций $\{\neg, \&, \vee\}$ является полной. Также полными являются следующие системы функций: а) $\{\neg, \vee\}$; б) $\{\neg, \&\}$; в) $\{\neg, \rightarrow\}$.

Полнота систем $\{\neg, \vee\}$ и $\{\neg, \&\}$ следует из полноты системы $\{\neg, \&, \vee\}$, а также законов де Моргана и двойного отрицания, следствием которых является возможность выразить конъюнкцию через дизъюнкцию и наоборот: $A\&B \sim \neg(\neg A \vee \neg B)$; $A \vee B \sim \neg(\neg A \& \neg B)$.

Поэтому система $\{\neg, \&, \vee\}$ может быть сокращена на одну функцию.

Полнота системы $\{\neg, \rightarrow\}$ следует из полноты системы $\{\neg, \vee\}$ и равносильности, позволяющей выразить импликацию через отрицание и дизъюнкцию: $(A \rightarrow B) \sim (\neg A \vee B)$.

Широкий набор базисов открывает большие возможности при решении задач минимизации схем устройств дискретного действия, поскольку из базисных схем с помощью суперпозиций можно составить схему, соответствующую любой булевой функции.

1.6. Нормальные формы

Определение 1.6.1. *Элементарной конъюнкцией* называется конъюнкция (возможно одночленная), составленная из переменных или отрицаний переменных.

Например, x, y, $x\&y$, $\neg x_1\&x_2\&(\neg x_3)$ – элементарные конъюнкции, а $x \vee y$, $x_1\&\neg x_2 \vee \neg x_1\&x_2$ – не являются элементарными конъюнкциями.

Определение 1.6.2. *Дизъюнктивной нормальной формой (ДНФ)* называется формула, имеющая вид дизъюнкции элементарных конъюнкций (в вырожденном случае это может быть одна конъюнкция).

Например, x, $x\&y$, $x \vee \neg x\&(\neg y)$, $\neg x_1\&x_2\&(\neg x_3) \vee x_1\&(\neg x_2)\&x_3 \vee x_1\&x_2\&(\neg x_3)$ – ДНФ, а $(x \vee y)\& \neg x$ – не ДНФ.

Определение 1.6.3. *Совершенной дизъюнктивной нормальной формой (СДНФ)* называется такая дизъюнктивная нормальная форма, каждый конъюнктивный член которой содержит *все* переменные либо их отрицания.

Например, $x\&y$, $x\&\neg y \vee \neg x\&y$ – СДНФ функции двух переменных, а $x \vee \neg x\&y$, $x \vee y$ – не СДНФ.

Определение 1.6.4. *Элементарной дизъюнкцией* называется дизъюнкция (возможно одночленная), составленная из переменных или отрицаний переменных.

Например, x, y, $x \vee y$, $\neg x_1 \vee x_2 \vee (\neg x_3)$ – элементарные дизъюнкции, а $x\&y$, $(x_1 \vee \neg x_2)\ \&\ (\neg x_1 \vee x_2)$ – не являются элементарными дизъюнкциями.

Определение 1.6.5. *Конъюнктивной нормальной формой (КНФ)* называется формула, имеющая вид конъюнкции элементарных дизъюнкций (в вырожденном случае это может быть одна дизъюнкция).

Например, x, $x\&y$, $x\ \&\ \neg x\&(\neg y)$, $(\neg x_1 \vee x_2)\&(\neg x_3)\&(x_1 \vee \neg x_2 \vee x_3)$ – КНФ, а $x\&y \vee \neg x$ – не является КНФ.

Определение 1.6.6. *Совершенной конъюнктивной нормальной формой (СКНФ)* называется такая конъюнктивная нормальная форма, каждый дизъюнктивный член которой содержит *все* переменные, либо их отрицания.

Например, $x \vee y$, $(x \vee \neg y)\ \&(\neg x \vee y)$ – СКНФ функции двух переменных, а $x\ \&(\neg x \vee y)$, $x\&y$ – не является СКНФ.

Теорема 1.6.1. Для каждой формулы булевой функции A имеется равносильная ей дизъюнктивная нормальная форма (ДНФ) и конъюнктивная нормальная форма (КНФ).

Доказательство теоремы состоит просто в указании алгоритмов нахождения для любой формулы A равносильных ей ДНФ и КНФ. Процесс нахождения этих форм называется *приведением* формулы A соответственно к ДНФ и КНФ. Для каждой формулы A имеется бесконечное множество ДНФ и КНФ, но для решения задач, в которых эти формы нужны, требуется, как правило, найти одну из них.

Алгоритм приведения формул булевых функций к ДНФ (КНФ)

Шаг 1. Все подформулы A вида $B \supset C$ (т.е. содержащие импликацию) заменяем на $\neg B \vee C$ или на $\neg (B \& \neg C)$.

Шаг 2. Все подформулы A вида $B \sim C$ (т.е. содержащие эквивалентность) заменяем на $(A \& B) \vee (\neg A \& \neg B)$ или на $(A \vee \neg B) \& (\neg A \vee B)$.

Шаг 3. Все отрицания, стоящие над сложными подформулами, опускаем по законам де Моргана.

Шаг 4. Устраняем все двойные отрицания над формулами в соответствии с законом двойного отрицания.

Шаг 5. Осуществляем раскрытие всех скобок по закону дистрибутивности конъюнкции относительно дизъюнкции для ДНФ или по закону дистрибутивности дизъюнкции относительно конъюнкции для КНФ.

Шаг 6. Для получения более простой формулы целесообразно удалить эквивалентные элементарные дизъюнкции или эквивалентные элементарные конъюнкции.

Очевидно, что в результате всех указанных операций формула примет вид ДНФ или КНФ. Указанные операции, вообще говоря, могут осуществляться в любом порядке, однако целесообразно придерживаться изложенного выше, за исключением снятия двойных отрицаний (шаг 4), от которых следует избавляться по мере их появления.

Пример 1.6.1.

Приведём к ДНФ и КНФ рассмотренную выше формулу $f(x_1, x_2, x_3) = \neg (x_2 \to \neg x_3) \sim (\neg x_1 \vee x_2)$.

1. Устранив импликацию, получим:

$\neg (\neg x_2 \vee \neg x_3) \sim (\neg x_1 \vee x_2)$.

2. Применив закон де Моргана к первой скобке и сняв двойные отрицания, получим: $(x_2 \& x_3) \sim (\neg x_1 \vee x_2)$.

3. Устранив эквивалентность, получим:

$(x_2 \& x_3) \& (\neg x_1 \vee x_2) \vee \neg (x_2 \& x_3) \& \neg (\neg x_1 \vee x_2)$.

4. Применив закон де Моргана ко второму члену дизъюнкции, получим:

$(x_2 \& x_3) \& (\neg x_1 \vee x_2) \vee (\neg x_2 \vee \neg x_3) \& (x_1 \& \neg x_2)$.

5. Применив закон дистрибутивности, получим:

$(x_2 \& x_3 \& \neg x_1 \vee x_2 \& x_3 \& x_2) \vee (\neg x_2 \& x_1 \& \neg x_2 \vee \neg x_3 \& x_1 \& \neg x_2)$.

6. Применив законы идемпотентности и располагая переменные по возрастанию индексов, получим:

$\neg x_1 \& x_2 \& x_3 \vee x_2 \& x_3 \vee x_1 \& x_2 \vee x_1 \& \neg x_2 \& \neg x_3$.

7. Применив закон поглощения, вместо $\neg x_1 \& x_2 \& x_3 \vee x_2 \& x_3$ запишем $x_2 \& x_3$, а вместо $x_1 \& \neg x_2 \vee x_1 \& \neg x_2 \& \neg x_3$ запишем $x_1 \& \neg x_2$ и в результате получим ДНФ нашей формулы: $f(x_1, x_2, x_3) \equiv x_2 \& x_3 \vee x_1 \& \neg x_2$.

При приведении к КНФ применим второй закон дистрибутивности и получим: $x_2 \& x_3 \vee x_1 \& \neg x_2 \equiv (x_2 \vee x_1) \& (x_2 \vee \neg x_2) \& (x_3 \vee x_1) \& (x_3 \vee \neg x_2)$.

Учитывая, что $x_2 \vee \neg x_2 \equiv$ (закон противоречия), получим окончательно КНФ для $f(x_1, x_2, x_3)$:

$f(x_1, x_2, x_3) \equiv (x_1 \vee x_2) \& (x_1 \vee x_3) \& (\neg x_2 \vee x_3)$.

Приведение некоторой формулы к ДНФ и КНФ не является однозначным. Количество равносильных ДНФ и КНФ, вообще говоря, бесконечно. Однако совершенные дизъюнктивные и конъюнктивные нормальные формы (СДНФ и СКНФ) или не существуют, или единственны.

Теорема 1.6.2. Каждая формула A, не равная тождественно нулю, может быть приведена к СДНФ, которая является единственной с точностью до перестановки дизъюнктивных членов.

Теорема 1.6.3. Каждая формула A, не равная тождественно единице, может быть приведена к СКНФ, которая является единственной с точностью до перестановки конъюнктивных членов.

Доказательство этих теорем состоит в указании алгоритма приведения формулы A к СДНФ и СКНФ.

Алгоритм приведения формулы булевой функции к СДНФ

Шаг 1. Используя алгоритм построения ДНФ, находим формулу B, являющуюся ДНФ формулы A .

Шаг 2. Вычёркиваем в B все элементарные конъюнкции, в которые одновременно входят какая-нибудь переменная и её отрицание. Это обосновывается равносильностями:

$A \& \neg A \equiv 0$, $B \& 0 \equiv 0$, $C \vee 0 \equiv C$.

Шаг 3. Если в элементарной конъюнкции формулы B некоторая переменная или её отрицание встречается несколько раз, то оставляем только одно её вхождение. Это обосновывается законом идемпотентности для конъюнкции: $A \& A \equiv A$.

Шаг 4. Если в элементарную конъюнкцию C формулы B не входит ни переменная x, ни её отрицание $\neg x$, то на основании 1-го закона расщепления заменяем C на $(C \& x) \vee (C \& \neg x)$.

Шаг 5. В каждой элементарной конъюнкции формулы B переставляем конъюнктивные члены так, чтобы для каждого i ($i = 1, ..., n$) на i-м месте была либо переменная x_i, либо её отрицание $\neg x_i$.

Шаг 6. Устраняем возможные повторения конъюнктивных членов согласно закону идемпотентности для дизъюнкции: $C \vee C \equiv C$.

Пример 1.6.2.

Найдём СДНФ для следующей формулы:

$f(x_1, x_2, x_3) = \neg\ (x_2 \to \neg x_3) \sim (\neg x_1 \mathrm{V} x_2)$.

1. Найденная ранее в примере ДНФ формулы $f(x_1, x_2, x_3)$ имеет вид: $x_2 \& x_3\ \mathrm{V}\ x_1 \& \neg x_2$.

2. Шаги 2 и 3 алгоритма не требуются (они уже выполнены), поэтому переходим к шагу 4 и применяем 1-й закон расщепления. В результате вместо каждого из двух конъюнктивных членов получим две элементарные конъюнкции (всего их будет четыре):

$f(x_1, x_2, x_3) \equiv x_2 \& x_3 \& x_1\ \mathrm{V}\ x_2 \& x_3 \& \neg x_1 \mathrm{V}\ x_1 \& \neg x_2 \& x_3\ \mathrm{V}\ x_1 \& \neg x_2 \& \neg x_3)$.

3. После применения шага 5 получим:

$f(x_1, x_2, x_3) \equiv x_1 \& x_2 \& x_3\ \mathrm{V}\ \neg x_1 \& x_2 \& x_3\ \mathrm{V}\ x_1 \& \neg x_2 \& x_3\ \mathrm{V}\ x_1 \& \neg x_2 \& \neg x_3$.

4. Шаг 6 не требуется. Найденное выражение формулы $f(x_1, x_2, x_3)$ является СДНФ этой формулы.

Алгоритм нахождения СКНФ полностью повторяет алгоритм нахождения СДНФ, если произвести двойственную замену & на V и V на &.

Пример 1.6.3.

Найдём СКНФ для формулы:

$f(x_1, x_2, x_3) = \neg\ (x_2 \to \neg x_3) \sim (\neg x_1 \mathrm{V} x_2)$.

1. Найденная выше КНФ формулы $f(x_1, x_2, x_3)$ имеет вид:

$f(x_1, x_2, x_3) \equiv (x_1 \mathrm{V} x_2)\ \&\ (x_1 \mathrm{V} x_3)\ \&\ (\neg x_2 \mathrm{V} x_3)$.

2. Шаги 2 и 3 алгоритма не требуются, поэтому переходим к шагу 4 и применяем 2-й закон расщепления. В соответствии с этим законом:

$x_1 \mathrm{V} x_2 \equiv (x_1 \mathrm{V} x_2 \mathrm{V} x_3)\ \&\ (x_1 \mathrm{V} x_2 \mathrm{V} \neg x_3)$.

$x_1 \mathrm{V} x_3 \equiv (x_1 \mathrm{V} x_3 \mathrm{V} x_2) \& (x_1 \mathrm{V} x_3 \mathrm{V} \neg x_2)$.

$\neg x_2\ \mathrm{V} x_3 \equiv (\neg x_2 \mathrm{V} x_3 \mathrm{V} x_1)\ \&\ (\neg x_2 \mathrm{V} x_3 \mathrm{V} \neg x_1)$.

Поэтому имеем:

$f(x_1, x_2, x_3) \equiv (x_1 \mathrm{V} x_2 \mathrm{V} x_3) \& (x_1 \mathrm{V} x_2 \mathrm{V} \neg x_3) \& (x_1 \mathrm{V} x_3 \mathrm{V} x_2) \& (x_1 \mathrm{V} x_3 \mathrm{V} \neg x_2) \&$
$\& (\neg x_2 \mathrm{V} x_3 \mathrm{V} x_1) \& (\neg x_2\ \mathrm{V}\ x_3 \mathrm{V} \neg x_1)$.

3. Применив шаг 5, получим:

$f(x_1, x_2, x_3) \equiv \equiv (x_1 \mathrm{V} x_2 \mathrm{V} x_3) \& (x_1 \mathrm{V} x_2 \mathrm{V} \neg x_3) \& (x_1 \mathrm{V} x_2 \mathrm{V} x_3) \& (x_1 \mathrm{V} \neg x_2 \mathrm{V} x_3) \&$
$\& (x_1 \mathrm{V} \neg x_2 \mathrm{V} x_3) \& (\neg x_1\ \mathrm{V} \neg x_2 \mathrm{V} x_3)$.

4. Замечаем, что 1-й и 3-й, а также 4-й и 5-й дизъюнктивные члены полученного выражения совпадают, применяем шаг 6 и получим окончательно СКНФ формулы $f(x_1, x_2, x_3)$:

$f(x_1, x_2, x_3) \equiv (x_1 \mathrm{V} x_2 \mathrm{V} x_3) \& (x_1 \mathrm{V} x_2 \mathrm{V} \neg x_3) \& (x_1 \mathrm{V} \neg x_2 \mathrm{V} x_3) \& (\neg x_1 \mathrm{V} \neg x_2 \mathrm{V} x_3)$.

Упражнения

1. Составьте таблицы истинности формул:

а) $(x \vee y) \leftrightarrow (y \downarrow \bar{x})$; б) $(x \,\&\, y) \rightarrow (z \oplus \neg(x \,\&\, y))$;

2. Проверьте двумя способами (составлением таблиц истинности; приведением формул к СДНФ или СКНФ с помощью эквивалентных преобразований), будут ли эквивалентны следующие формулы:

а) $x \leftrightarrow (y|z)$ и $(x \leftrightarrow y)|(x \leftrightarrow z)$; б) $x \downarrow (y|z)$ и $(x \downarrow y)|(x \downarrow z)$.

3. С помощью эквивалентных преобразований приведите формулу к ДНФ, КНФ, СДНФ, СКНФ:

а) $\neg((x \downarrow y) \rightarrow \bar{z})$; б) $\neg(\neg(x \downarrow y) \rightarrow (z \leftrightarrow \bar{y}))$.

2. ЛОГИКА ПРЕДИКАТОВ

Логика предикатов изобретена Готлобом Фреге. Он считал её универсальным языком, в котором можно было бы представить систематически и математически точным образом любую возможную форму рационального мышления, которая могла бы стать частью дедуктивного мышления. Логика предикатов, как и традиционная формальная логика, расчленяет элементарное высказывание на субъект (буквально – подлежащее, хотя оно и может играть роль дополнения) и предикат (буквально – сказуемое, хотя оно может играть и роль определения).

2.1. Определение предикатов. Кванторы

Логика предикатов – это расширение возможностей логики высказываний, позволяющее строить высказывания с учётом свойств изучаемых объектов или отношений между ними. Многие утверждения, имеющие форму высказываний, содержат переменные, конкретные значения которых не указаны, им не может быть предписано истинностное значение, называются предикатами.

Одноместный предикат $P(x)$ – это утверждение об объекте x, где x рассматривается как переменная. Иначе говоря, если в $P(x)$ вместо x подставить конкретный изучаемый объект a, то получаем высказывание, принадлежащее алгебре высказываний. В таком случае

$P(a)=1$ или $P(a)=0$. Двуместным предикатом $P(x,y)$ называется функция двух переменных x,y, определённая на множестве $M=M_1\times M_2$ и принимающая значения из множества $\{1,0\}$.

Определение 2.1.1. *Предикатом* $P(x_1, x_2, \ldots, x_n)$ называется функция, аргументы которой определены на некотором множестве M, $x_1, x_2, \ldots, x_n \in M$, а сама она принимает два значения: 1 (истина) и 0 (ложь). Таким образом, предикат осуществляет отображение $M^n \to \{0, 1\}$.

Переменные $x_1, x_2, \ldots, x_n$ называются *предметными переменными*, а множество *M – предметной областью*.

Если все переменные $x_1, x_2, \ldots, x_n$ принимают конкретные значения, то предикат есть не что иное, как высказывание. Таким образом, высказывание является частным случаем предиката. Можно сказать, что предикат есть высказывание, зависящее от параметров. Множество, на котором предикат P принимает значение 1, называется множеством истинности предиката P.

Примеры 2.1.1.

а) $P(x)$ = «x – чётное число». Здесь M – множество целых чисел, $x\in M$.

б) $A(x, y, z)$ = «точки x, y, z лежат на одной окружности». Здесь M – множество точек плоскости, $x, y, z \in M$.

в) $B(x, y)$ = «x старше y». Здесь M – множество людей, $x, y\in M$.

Предикат от n переменных называется *n-местным предикатом.* Высказывание есть 0-местный предикат.

Как видно из примера а), одноместный предикат отражает свойство некоторого объекта, а многоместный предикат выражает отношение между многими объектами.

Расширение логики высказываний до логики предикатов получается за счёт включения в формулы утверждений, являющихся предикатами. Но при этом, поскольку предикаты относятся к изучаемым объектам, мы обязаны включить в теорию и сами объекты $a_1,\ldots,a_n,\ldots$. Поэтому, чтобы дать определение формул в логике предикатов, необходимо уточнить принципы описания в логике предикатов объектов. Делается это с помощью понятия «терм».

Можно дать следующее определение терма:

1) Предметные переменные $x_1,\ldots,x_n,\ldots$ – это термы;

2) если $f(,\ldots,)$ функция n переменных, ставящая в соответствие изучаемым объектам объект, и $t_1,\ldots,t_n$ термы, то $f(t_1,\ldots,t_n)$ – терм.

Теперь можно дать определение формулы:

1) если $P(\cdot,...,\cdot)$ – n-местный предикат, а $t_1,...,t_n$ – термы, то $P(t_1,...,t_n)$ (атомарная) формула;
2) если A и B – формулы, то $(A \,\&\, B),(A \vee B),(A \to B)$ – формулы;
3) если A – формула, то $\neg A$ – формула;
4) если $A(x)$ – формула, содержащая переменную x, то

$$\forall x A(x),\ \exists x A(x) - \qquad (2.1.1)$$

формулы.

Обращают на себя внимание необычные значки в формуле (2.1.1). Это кванторы. Знак $\forall$ называется квантором всеобщности, а знак $\exists$ – квантором существования. Ввёл их в логику Ч.С. Пирс, хотя идея квантификации принадлежит Г. Фреге.

Квантор общности. Пусть $P(x)$ – некоторый предикат, определённый для каждого $x \in M$. Тогда выражение $\forall x P(x)$ является истинным высказыванием, если $P(x)$ истинно для всякого $x \in M$, и ложным в противном случае. Символ $\forall x$ называется *квантором общности.* Выражение $\forall x P(x)$ читается: «Для всех x имеет место $P(x)$». В обычной речи квантору общности соответствуют слова: все, всякий, каждый, любой. Возможно отрицание квантора общности: $\neg \forall x P(x)$: «Не для всех x имеет место $P(x)$».

Пример 2.1.2.

Пусть $P(x)$ – предикат «x – чётное число». Тогда $\forall x P(x)$ есть высказывание «Всякое x – четное число» = «Все числа – чётные», которое истинно на множестве M чётных чисел и ложно, если M содержит хотя бы одно нечётное число, например, если M – множество целых чисел. Отрицание $\neg \forall x P(x)$ есть высказывание «Не всякое x – чётное число» = «Не все числа – чётные», которое истинно на множестве целых чисел и ложно на множестве чётных чисел.

Квантор существования. Пусть $P(x)$ – некоторый предикат, $x \in M$. Тогда выражение $\exists x P(x)$ является истинным высказыванием, если $P(x)$ истинно хотя бы для одного $x \in M$, и ложным в противном случае. Символ $\exists x$ называется *квантором существования.* Выражение $\exists x P(x)$ читается: «Существует x, для которого имеет место $P(x)$». В обычной речи квантору существования соответствуют слова: некоторый, несколько. Возможно отрицание квантора существования: $\neg \exists x P(x)$: «Не существует x, для которого имеет место $P(x)$».

Кванторы существования и общности называются *двойственными* кванторами.

Пример 2.1.3.

Пусть, как и в предыдущем примере, $P(x)$ – предикат «x – чётное число». Тогда $\exists x P(x)$ есть высказывание «Некоторые x – чётные числа» = «Существуют чётные числа», которое истинно на множестве M, содержащем хотя бы одно чётное число и ложно, если M содержит только нечётные числа. Высказывание $\neg \exists x P(x)$ = «Неверно, что некоторые x – чётные числа» = «Не существует чётных чисел» истинно на множестве M, содержащем только нечётные числа и ложно, если M содержит хотя бы одно чётное число.

Буква x, стоящая справа от квантора, называется *кванторной переменной* и должна присутствовать обязательно. Переменная, стоящая под знаком квантора, называется также *связанной переменной.* Несвязанная переменная называется *свободной.* Выражения $\forall x P(x)$ и $\exists x P(x)$ не зависят от x и имеют вполне определённые значения. Поэтому переименование связанной переменной, т.е. переход от выражения $\forall x P(x)$ к $\forall y P(y)$, не меняет его истинностного значения.

Кванторы могут применяться и к многоместным предикатам. При этом число свободных переменных уменьшается на единицу. Одноместный предикат при связывании переменной квантором становится 0-местным предикатом, т.е. высказыванием.

Дизъюнкцией двух предикатов $P(x)$ и $Q(x)$ называется новый предикат $P(x) \vee Q(x)$, который принимает значение «ложь» при тех и только тех значениях $x \in M$, при которых каждый из предикатов принимает значение «ложь», и принимает значение «истина» от всех остальных случаях.

Отрицанием предиката $P(x)$ называется новый предикат $\neg P(x)$, который принимает значение «истина» при всех значениях $x \in M$, при которых предикат $P(x)$ принимает значения «ложь», и принимает значения «ложь» при тех значениях $x \in M$, при которых предикат $P(x)$ принимает значения «истина».

Импликацией предикатов $P(x)$ и $Q(x)$ называется новый предикат $P(x) \to Q(x)$, который является ложным при тех и только тех значениях $x \in M$, при которых одновременно $P(x)$ принимает значение «истина», а $Q(x)$ – значение «ложь», и принимает значение «истина» во всех остальных случаях.

Из элементарных предикатов с помощью логических операций можно получить сложные предикаты.

В качестве системы аксиом исчисления предикатов могут быть использованы следующие формулы:

1. $P \to (Q \to P)$.
2. $(P \to (Q \to C)) \to ((P \to Q) \to (P \to C))$.
3. $P \,\&\, Q \to P$.
4. $P \,\&\, Q \to Q$.
5. $(P \to Q) \to ((P \to C) \to (P \to (Q \,\&\, C)))$.
6. $P \to P \vee Q$.
7. $Q \to P \vee Q$.
8. $(P \to C) \to ((Q \to C) \to ((P \vee Q) \to C))$.
9. $(P \to Q) \to (\neg Q \to \neg P)$.
10. $P \to (\neg\neg P)$.
11. $(\neg\neg P) \to P$.
12. $(\forall x P(x)) \to P(y)$.
13. $P(y) \to (\exists x P(x))$.

В аксиомах (12) и (13) предполагается, что терм y свободен по переменной x в формуле P. О логическом значении формулы логики предикатов можно говорить лишь тогда, когда задано множество M, на котором определены входящие в эту формулу предикаты.

Формально-логический анализ рассуждений в логике предикатов, безотносительно к содержанию отдельных предикатов, приводит к исчислению предикатов первого рода как формальной теории.

2.2. Интерпретации

За каждой формулой скрывается нечто, в связи с чем она была написана и о чём она повествует, т.е. скрывается её содержание. Содержательная часть формул, их смысл, относится к специальному разделу логики, называемому семантикой. Выяснить содержание формулы можно, обращаясь к реальному миру предметов. Делается это посредством интерпретации формулы.

Интерпретация состоит в указании конкретного непустого множества M, называемого областью интерпретации, и некоторого правила, по которому каждому n-местному предикату $P(\cdot,...,\cdot)$ ставится в соответствие отношение $\overline{P} \subset M^n$ в M, а функциям (операциям) n переменных $f(\cdot,...,\cdot)$, участвующим в определении термов, конкретные

функции (операции) $\bar{f} : M^n \to M$ на M. При заданной интерпретации переменные, входящие в формулы, мыслятся пробегающими область M. Формально интерпретация – это структура вида:

$$\mathrm{M} = \left\langle M; \overline{P}_1,..., \overline{P}_p; \bar{f}_1,..., \bar{f}_q \right\rangle.$$

Содержанием при интерпретации формулы наполняются благодаря тому, что элементы множества M уже привязаны к конкретной реальности, знакомы и понятны. Например, в случае, когда $M = R$, т.е. является множеством действительных чисел, у нас не появляется чувства беспокойства; под формулами мы понимаем сведения из математического анализа, который прочно привязан к практической деятельности инженеров, физиков, химиков и т.д. Следовательно, нам становится ясным, когда формула при определённой фиксации своих переменных истинна, когда – ложна.

Пусть задано множество M изменения предметных переменных формулы $A(x_1, x_2, \ldots, x_n)$, т.е. $(x_1, x_2, \ldots, x_n) \in M$.

Определение 2.2.1. Формула A называется *выполнимой в данной интерпретации*, если существует набор значений переменных $(a_1, a_2, \ldots, a_n) \in M$, для которого $A(a_1, a_2, \ldots, a_n) = \text{И}$.

Определение 2.2.2. Формула A называется *истинной в данной интерпретации*, если $A(x_1, x_2, \ldots, x_n) = \text{И}$ на любом наборе своих переменных $(x_1, x_2, \ldots, x_n) \in M$.

Определение 2.2.3. Формула A называется *общезначимой или тождественно-истинной*, если она истинна в каждой интерпретации.

Определение 2.2.4. Формула A называется *выполнимой*, если существует интерпретация, для которой она выполнима.

2.3. Равносильность формул логики предикатов

Определение 2.3.1. Формулы F и G, определённые на некотором множестве M, называются *равносильными на этом множестве*, если при любых подстановках констант вместо переменных они принимают одинаковые значения.

Определение 2.3.2. Формулы, равносильные на любых множествах, будем называть просто *равносильными.*

Переход от одних формул к равносильным им другим формулам логики предикатов может быть произведён по следующим правилам:

1. Все равносильности, имеющие место для логики высказываний, переносятся на логику предикатов.

Примеры 2.3.1.

а) $\exists x(A(x) \to \forall yB(y)) \sim \exists x(\neg A(x) \vee \forall yB(y))$.

б) $\forall xA(x) \to (B(z) \to \forall xC(x)) \sim \neg(\forall xA(x)) \vee \neg B(z) \vee \forall xC(x)$.

в) $(\exists xA(x) \to \forall yB(y)) \to C(z) \sim \neg(\exists xA(x) \to \forall yB(y)) \vee C(z) \sim$

$\sim \neg (\neg(\exists xA(x)) \vee \forall yB(y) \vee C(z) \sim \exists xA(x) \& \neg(\forall yB(y)) \vee C(z)$.

2. Перенос квантора через отрицание.

Пусть A – формула, содержащая свободную переменную x. Тогда

$$\neg(\forall xA(x) \sim \exists x(\neg A(x)), \qquad (2.3.1)$$

$$\neg(\exists xA(x)) \sim \forall x \neg(A(x)). \qquad (2.3.2)$$

Правило переноса квантора через знак отрицания можно сформулировать так: знак отрицания можно ввести под знак квантора, заменив квантор на двойственный.

Справедливость равносильностей (2.3.1) и (2.3.2) вытекает из смысла кванторов. Так, левая часть (2.3.1) может быть прочитана следующим образом: «Неверно, что для всякого x имеет место $A(x)$». В правой же части (2.3.1) утверждается: «Существует x, для которого $A(x)$ не имеет места». Очевидно, что оба утверждения одинаковы. В левой и правой частях (2.3.2) соответственно содержатся одинаковые утверждения: «Неверно, что существует x, для которого имеет место $A(x)$» и «Для всех x не имеет места A(x)».

Пользуясь равносильностями (2.3.1) и (2.3.2), а также равносильностями логики высказываний, можно для каждой формулы найти такую равносильную ей формулу, в которой знаки отрицания относятся к элементарным высказываниям и элементарным предикатам.

Например, с помощью элементарных преобразований получаем:

$\neg(\exists x(A(x) \to \forall yB(y)) \sim \neg(\exists x(\neg A(x) \vee \forall yB(y)) \sim \forall x(\neg(\neg A(x) \vee \forall yB(y))) \sim \forall x(A(x) \& \neg \forall yB(y)) \sim \forall x(A(x) \& \exists y \neg B(y))$.

3. Вынос квантора за скобки.

Пусть формула $A(x)$ содержит переменную x, а формула B не содержит переменной x, и все переменные, связанные в одной формуле, связаны в другой. Тогда имеют место соотношения:

$$\forall xA(x) \vee B \sim \forall x(A(x) \vee B), \qquad (2.3.3)$$

$$\forall xA(x) \& B \sim \forall x(A(x) \& B), \qquad (2.3.4)$$

$$\exists xA(x) \vee B \sim \exists x(A(x) \vee B), \qquad (2.3.5)$$

$$\exists xA(x) \& B \sim \exists x(A(x) \& B). \qquad (2.3.6)$$

Докажем формулу (2.3.3). Пусть формула $\forall xA(x) \vee B$ истинна на некотором множестве изменения переменных M и при некоторых фиксированных значениях свободных переменных. Тогда либо формула $\forall xA(x)$, либо формула B истинна. Если истинна формула $\forall xA(x)$,

то формула A(x) истинны для всякого x, принадлежащего M и, следовательно, формула $A(x)$ V B тоже истинна для всякого x из M. Но тогда истинна формула $\forall x(A(x) \vee B)$.

Если формула $\forall xA(x) \vee B$ ложна, то ложны формулы $\forall xA(x)$ и B. Следовательно, так как B не зависит от x, для всякого $x \in M$ формула $A(x)$ V B ложна. Но тогда ложна формула $\forall x(A(x)$ V $B)$.

Равносильности (2.3.4) – (2.3.6) доказываются аналогично.

4. Дистрибутивность квантора общности относительно конъюнкции и квантора существования относительно дизъюнкции.

Пусть формула B, так же как и формула A, зависит от x. Тогда

$$\forall xA(x) \& \forall xB(x) \sim \forall x(A(x) \& B(x)), \qquad (2.3.7)$$

$$\exists xA(x) \vee \exists xB(x) \sim \exists x(A(x) \vee B(x)). \qquad (2.3.8)$$

Докажем (2.3.7). Пусть правая часть (2.3.7) истинна, т.е. $\forall x(A(x)$ & $B(x))$ – истинно. Тогда для любого $x_0 \in M$ истинно значение $A(x_0)$ & $B(x_0)$. Поэтому значения $A(x_0)$ и $B(x_0)$ одновременно истинны для любого x_0. Следовательно, истинна формула $\forall xA(x)$ & $\forall xB(x)$.

Если же правая часть (2.3.7) ложна, то для некоторого $x_0 \in M$ либо значение $A(x_0)$, либо значение $B(x_0)$ ложно. Значит, ложно либо $\forall xA(x_0)$, либо $\forall xB(x_0)$. Следовательно, $\forall xA(x)$ & $\forall xB(x)$ ложно.

Равносильность (2.3.8) доказывается аналогично.

Дистрибутивные законы для квантора общности относительно дизъюнкции и квантора существования относительно конъюнкции, вообще говоря, не имеют места, т.е. формулы $\forall xA(x)$ V $\forall xB(x)$ и $\forall x(A(x)$ V $B(x))$, а также

$$\exists xA(x) \,\&\, \exists xB(x) \text{ и } \exists x(A(x) \,\&\, B(x))$$

не являются равносильными, хотя они могут быть равносильными на некоторых множествах M.

Примеры 2.3.2.

1. Показать, что формулы $\forall x(A(x)$ V $B(x))$ и $\forall xA(x)$ V $\forall xB(x))$ не равносильны.

Пусть M – множество натуральных чисел, $A(x)$ = «x – чётное число», $B(x)$ = «x – нечётное число». Тогда $\forall x(A(x)$ V $B(x))$ = «Всякое натуральное число чётное или нечётное» = 1.

$\forall xA(x)$ = «Всякое натуральное число – чётное» = 0,

$\forall xB(x)$ = «Всякое натуральное число – нечётное» = 0,

$\forall xA(x)$ V $\forall xB(x)$ = «Всякое натуральное число чётное или всякое натуральное число нечётное» = 0, т.е. формулы $\forall x(A(x)$ V $B(x))$ и $\forall xA(x)$ V $\forall xB(x)$ не равносильны.

2. Показать, что формулы $\exists x(A(x) \,\&\, B(x))$ и $\exists xA(x) \,\&\, \exists xB(x)$ не равносильны.

Пусть A(x) = «У x голубые глаза», $B(x)$ = «У x чёрные глаза». Тогда

$\exists x(A(x) \,\&\, B(x))$ = «У некоторых голубые и чёрные глаза» = 0,

$\exists xA(x)$ = «У некоторых голубые глаза» = 1,

$\exists xB(x)$ = «У некоторых чёрные глаза» = 1,

$\exists xA(x) \,\&\, \exists xB(x)$ = «У некоторых голубые, и у некоторых чёрные глаза» = 1, т.е. формулы $\exists x(A(x) \,\&\, B(x))$ и $\exists xA(x) \,\&\, \exists xB(x)$ не равносильны.

5. Перестановка одноимённых кванторов.

$$\forall x\forall yA(x,y) \sim \forall y\forall xA(x,y), \qquad (2.3.9)$$

$$\exists x\exists yA(x,y) \sim \exists y\exists xA(x,y). \qquad (2.3.10)$$

Разноимённые кванторы переставлять, вообще говоря, нельзя.

Примеры 2.3.3.

Пусть M – множество натуральных чисел, $A(x, y)$ = «$x > y$».

а) $\forall x\forall yA(x, y)$ = «Для всех x и y имеет место $x > y$» = 0;

$\forall y\forall xA(x, y)$ = «Для всех y и x имеет место $x > y$» = 0;

$\forall x\forall yA(x, y) \equiv \forall y\forall xA(x, y)$.

б) $\exists x\exists yA(x, y)$ = «Существуют такие x и y, что $x > y$» = 1;

$\exists y\exists xA(x, y)$ = «Существуют такие y и x, что x > y» = 1;

$\exists x\exists yA(x, y) \equiv \exists\, y\exists xA(x, y)$.

в) $\exists x\forall yA(x, y)$ = «Существует такое x, что для всякого y имеет место $x > y$» = 0 (утверждается существование максимального числа на множестве натуральных чисел);

$\forall y\exists xA(x, y)$ = «Для всякого y существует такое x, что $x > y$» = 1;

$\exists x\forall yA(x, y) \neq \forall y\exists xA(x, y)$.

г) $A(x,y)$ = «Книгу x читал человек y».

$\forall x\exists yA(x, y)$ = «Каждую книгу читал кто-нибудь» = 1 (например, автор книги читал свою книгу);

$\exists y\forall xA(x, y)$ = «Существует человек, который читал все книги» = 0;

$\forall x\exists yA(x, y) \neq \exists y\forall xA(x, y)$.

Очень полезной при выполнении элементарных преобразований формул, содержащих предикаты, является следующее преобразование формул логики предикатов.

6. Переименование связанных переменных.

Заменяя связанную переменную формулы A другой переменной, не входящей в эту формулу, всюду: в кванторе и в области действия квантора получим формулу, равносильную A.

Пример 2.3.4.

Пусть $A = \forall x F(x) \to \exists x G(x)$.

Заменяя связанную переменную x на y в первом члене импликации и на z – во втором, получим равносильную формулу:

$B = \forall y F(y) \to \exists z G(z)$, т.е. $A \equiv B$.

7. Выше была доказана дистрибутивность квантора общности относительно конъюнкции и квантора существования относительно дизъюнкции (тождества (2.3.7) и (2.3.8)). Этот факт означает, что в вышеуказанных случаях соответствующие кванторы могут быть вынесены за скобки и помещены впереди формулы, что и демонстрируют тождества (2.3.7) и (2.3.8).

Рассмотрим теперь случай, когда закон дистрибутивности, вообще говоря, не применим. Сначала рассмотрим формулу $\forall x A(x)$ V $\forall x B(x)$ и применим правило переименования переменных. Получим:

$$\forall x A(x) \mathrm{V} \forall x B(x) \sim \neg \forall x A(x) \mathrm{V} \forall y B(y). \qquad (2.3.11)$$

Так как $\forall y B(y)$ не зависит от x, справедлива равносильность (2.3.3), причём $B = \forall y B(y)$. Поэтому в соответствии с (2.3.3) можно вынести за скобки $\forall x$:

$$\forall x A(x) \mathrm{V} \forall y B(y) \sim \neg \forall x (A(x) \mathrm{V} \forall y B(y)). \qquad (2.3.12)$$

Так как $A(x)$ не зависит от y, справедлива равносильность (2.3.3), причём на этот раз $B = A(x)$. Поэтому в соответствии с (2.3.3) можно вынести за скобки $\forall y$:

$$A(x) \mathrm{V} \forall y B(y) \sim \neg \forall y (A(x) \mathrm{V} B(y)). \qquad (2.3.13)$$

Учитывая (2.3.11), (2.3.12), (2.313), получим:

$$\forall x A(x) \mathrm{V} \forall x B(x) \sim \forall x \forall y (A(x) \mathrm{V} B(y)). \qquad (2.3.14)$$

Таким образом, за скобки выносятся два квантора $\forall x$ и $\forall y$, а выражение в скобках не содержит знаков квантора.

Проведём аналогичные выкладки для формулы $\exists x A(x)$ & $\exists x B(x)$:

$$\exists x A(x) \& \exists x B(x) \sim \exists x A(x) \ \& \ \exists y B(y) \sim \exists x (A(x) \& \exists y B(y)) \sim \exists x \exists y (A(x) \& B(y)). \qquad (2.3.15)$$

Аналогично можно доказать следующие равносильности:

$$\forall x A(x) \mathrm{V} \exists x B(x) \sim \forall x \exists y (A(x) \mathrm{V} B(y)), \qquad (2.3.16)$$

$$\forall x A(x) \& \exists x B(x) \sim \forall x \exists y (A(x) \& B(y)). \qquad (2.3.17)$$

2.4. Приведённые и нормальные формулы

Определение 2.4.1. Формула алгебры предикатов называется приведённой, если она является атомарной или образована из атомарных формул и их отрицаний только с помощью навешивания кванторов, конъюнкции и дизъюнкции, т.е. в приведённых формулах из логических символов имеются только символы &, V и $\neg$, причём символ $\neg$ встречается лишь перед символами предикатов.

Пример 2.4.1.

1. $A(x)\&B(x, y)$.
2. $\forall xA(x)$ V $\exists x\neg B(x, y)$.
3. $\forall x\forall yF_1(x, y, z)\wedge\exists x\exists y\neg F_2(x, y)$.
4. $\neg$ $(A(x)\&B(x, y))$.
5. $\forall xA(x)\rightarrow\exists x\neg B(x, y)$.
6. $\neg$ $(\forall xA(x)\rightarrow\exists x\neg B(x, y))$.
7. $\forall x(F_1(x, y)\rightarrow F_2(x, y, z))\vee\neg\vee xF_3(x)$.

Первые три формулы в соответствии с определением являются приведёнными, остальные не являются приведёнными. В третьей формуле знак отрицания стоит перед формулой, а не перед символами предикатов. В остальных формулах используется недопустимый для приведённой формулы символ импликации $\rightarrow$.

Теорема 2.4.1. Для каждой формулы существует равносильная ей приведённая формула, причём множества свободных и связанных переменных этих формул совпадают.

Действительно, пользуясь равносильностями логики высказываний, можно получить формулу, содержащую только символы &, V и $\neg$. Применяя затем правило переноса квантора через знак отрицания, можно получить равносильную приведённую формулу. Такая приведённая формула называется приведённой формулой данной формулы.

Примеры 2.4.2.

Рассмотрим четвёртую, пятую и шестую формулы предыдущего примера и получим для них приведённые формулы.

Для четвёртой формулы по закону де Моргана:

$$\neg\ (A(x)\&B(x, y))\sim\neg A(x)\ \text{V}\ \neg B(x, y).$$

Для пятой формулы:

$$\forall xA(x)\rightarrow\exists x\neg B(x, y)\sim\neg\forall xA(x)\ \text{V}\ \exists x\neg B(x, y)\sim\exists x\neg A(x)\ \text{V}\ \exists x\neg B(x, y).$$

Для шестой формулы:

$$\neg\ (\forall xA(x)\rightarrow\exists x\neg B(x, y))\sim\neg\exists x\neg A(x)\ \text{V}\ \exists x\neg B(x, y))\sim\neg(\exists x\neg A(x))\ \&$$
$$\&\ \neg(\exists x\neg B(x, y))\sim\forall xA(x)\ \&\ \forall xB(x, y).$$

Определение 2.4.2. Предварённой нормальной формой (ПН-формой) называется всякая формула вида

$$Q_1x_1...Q_kx_kA,\qquad(2.4.1)$$

где Q_i обозначает квантор общности или существования, а формула A не содержит кванторов и является приведённой формулой, т.е. ПН-форма не содержит символов кванторов или все символы кванторов стоят впереди. При этом выражение $Q_1x_1...Q_kx_k$ и формула A называются, соответственно, приставкой и матрицей формулы (2.4.1).

Примеры 2.4.3.

1. $\forall x\exists y(\&A(x) \vee B(x, y))$ – ПН-форма.
2. $\exists x\exists y\exists z\exists u(F_1(x, y)\vee F_2(x, y, z)\& F_3(y, u))$ – ПН-форма.
3. $\forall x(\&A(x)) \ \& \ \exists yB(x, y)$ – приведённая формула, не являющаяся ПН-формой.

Теорема 2.4.2. Для каждой приведённой формулы существует равносильная ей нормальная ПН-форма.

Алгоритм, позволяющий из приведённой формулы получить равносильную ей ПН-форму, основан на правиле переименования связанных переменных и использовании равносильностей (2.3.3) – (2.3.8), (2.3.14) и (2.3.17).

Пусть Q – любой из кванторов $\forall$, $\exists$.

Имеют место равносильные преобразования:

$$QxA(x)\vee B\sim Qx(A(x)\vee B), \qquad (2.4.2)$$

$$QxA(x)\&B \sim Qx(A(x)\&B). \qquad (2.4.3)$$

В тождествах (2.4.2), (2.4.3) формула B не зависит от x.

$$Q_1xA(x)\&Q_2xB(x) \sim Q_1xQ_2z(A(x)\&B(z)), \qquad (2.4.4)$$

$$Q_1xA(x)\vee Q_2xB(x) \sim Q_1xQ_2z(A(x)\vee B(z)). \qquad (2.4.5)$$

Тождества (2.4.2) и (2.4.3) есть обобщённая запись равносильных преобразований (2.3.3) – (2.3.6), а тождества (2.4.4) и (2.4.5) обобщают равносильности (2.3.14) – (2.3.17).

Мы видим, что тождества (2.4.2) – (2.4.5) позволяют поместить кванторы впереди формулы, что и требуется для нормальной формулы логики предикатов.

Пример 2.4.4.

Найти равносильную ПН-форму для приведённой формулы: $\forall x\exists yA(x, y) \ \& \ \exists x\exists u(x, u)$.

В формуле $\exists yA(x, y)$ переменная y связана, поэтому $\exists yA(x, y)$ не зависит от y. Обозначим $D(x) = \exists yA(x, y)$.

В формуле $\exists uB(x, u)$ переменная u связана, поэтому $\exists uB(x, u)$ не зависит от u. Обозначим $F(x) = \exists uB(x, u)$. Тогда

$$\forall x\exists yA(x,y)\&\exists x\exists uB(x,u)\sim\forall xD(x)\&\exists xF(x). \qquad (2.4.6)$$

Применим равносильность (2.4.4), имея в виду, что Q_1x есть $\forall x$, а Q_2x есть $\exists x$. Получим:

$$\forall xD(x)\&\exists xF(x)\sim\forall x\exists z(D(x)\&F(z)).$$

Рассмотрим формулу $D(x) \ \& \ F(z) \sim \exists yA(x, y) \ \& \ \exists uB(z, u)$. Применив два раза равносильность (2.4.3), получим:

$$\exists yA(x,y)\&\exists uB(z,u) \sim \exists y(A(x,y)\&\exists uB(z,u)) \sim \exists y\exists u(A(x, y)\&B(z, u)). \qquad (2.4.7)$$

Учитывая (2.4.5), (2.4.6), (2.4.7), получим окончательно:

$$\forall x\exists y A(x,y)\&\exists x\exists u B(x,u) \sim \forall x\exists z\exists y\exists u(A(x,y)\&B(z,u)). \qquad (2.4.8)$$

В тождестве (2.4.8) в левой части – исходная формула, а в левой части её нормальная формула.

Теорема 2.4.3. Для каждой формулы существует равносильная ей нормальная формула.

Теорема 2.4.3 является очевидным следствием теорем 2.4.1 и 2.4.2.

Пример 2.4.5.

Найти равносильную нормальную формулу для формулы:

$$\forall x\exists y A(x, y) \to \exists x\exists u B(x, u).$$

1. Найдём вначале приведённую формулу, равносильную данной. Избавимся от символа $\to$:

$$\forall x\exists y A(x, y) \to \exists x\exists B(x, u) \sim \neg(\forall x\exists y A(x, y)) \vee \exists x\exists u B(x, u).$$

Применим равносильности (2.3.1) и (2.3.2) (перенос квантора через отрицание):

$$\neg\ (\forall x\exists y A(x, y)) \sim \exists x\forall y\neg A(x, y).$$

Следовательно,

$$\forall x\exists y A(x,y)\to\exists x\exists u B(x,u)\sim\exists x\forall y\neg A(x,y)\vee\exists x\exists u B(x,u). \qquad (2.4.9)$$

Правая часть тождества (2.4.9) есть приведённая формула.

2. Найдём теперь ПН-форму, равносильную приведённой формуле $\exists x\forall y\neg A(x,\ y)\ \vee\ \exists x\exists u B(x,\ u)$. Выполним преобразование этой формулы, аналогично предыдущему примеру:

$$\exists x\forall y\neg A(x,y)\vee\exists x\exists u B(x,u) \sim \exists x\forall y\neg A(x, y)\vee\exists z\exists u B(z, u) \sim$$

$$\forall x\exists z(\forall y\neg A(x,y)\vee\exists u B(z,u)) \sim \forall x\exists z\forall y\exists u(\neg A(x,y)\vee B(z,u)). \qquad (2.4.10)$$

В правой части (2.4.10) – ПН-форма, равносильная исходной формуле.

2.5. Выражение суждения в виде формулы логики предикатов

Существуют две задачи, определяющие связь между суждениями и формулами логики предикатов:

1) выражение суждения в виде формулы логики предикатов;

2) интерпретация формулы логики предикатов.

Рассмотрим первую задачу.

Суждение (в том числе и в математике) – это мысль, в которой утверждается наличие или отсутствие свойств предметов, отношений между предметами.

Простым суждением назовём суждение, в котором нельзя выделить часть, в свою очередь являющуюся суждением. Среди простых суждений выделяют *атрибутивные суждения* и *суждения об отношениях*.

В атрибутивных суждениях выражается наличие или отсутствие у предметов некоторых свойств. Например, «Иванов – спортсмен», «Все сладкоежки любят конфеты», «Ни один студент нашей группы не знает испанский язык», «Некоторые океаны имеют пресную воду».

Все атрибутивные суждения можно разделить на следующие типы: «*a* есть *P*», «Все *S* есть *P*», «Ни один *S* не есть *P*», «Некоторые *S* есть *P*», «Некоторые *S* не есть *P*». Эти суждения следующим образом переводятся на язык логики предикатов:

«*a* есть *P*» – $P(a)$;

«Все *S* есть *P*» – $\forall x(S(x) \to P(x))$;

«Ни один *S* не есть *P*» – $\forall x(S(x) \to \neg P(x))$;

«Некоторые *S* есть *P*» – $\exists x(S(x) \,\&\, P(x))$;

«Некоторые *S* не есть *P*» – $\exists x(A(x) \,\&\, \neg P(x))$.

Полезно понять и запомнить следующее правило: если кванторная переменная связана квантором общности ($\forall$), то в формуле используется знак импликации ($\to$), а если кванторная переменная связана квантором существования ($\exists$), то – знак конъюнкции ($\&$). Рассмотрим соответствующие ситуации на конкретных примерах.

Примеры 2.5.1.

Перевести на язык логики предикатов следующие суждения:

а) «Веста – собака».

Заменим имя «Веста» символом «*в*» и введём соответствующий предикат $P(x)$ = «x – собака».

Это суждение можно выразить формулой: $P(\text{в})$.

б) «Всякая логическая функция может быть задана таблицей».

Введём предикаты $S(x)$ = «x – логическая функция»; $P(x)$ = «x может быть задана таблицей».

Наше суждение можно выразить формулой: $\forall x(S(x) \to P(x))$.

в) «Ни один народ не хочет войны».

Введём предикаты $S(x)$ = «x – народ»; $P(x)$ = «x хочет войны».

Наше суждение можно выразить формулой: $\forall x(S(x) \to \neg P(x))$.

г) «Некоторые журналисты были в космосе».

Введём предикаты $S(x)$ = «x – журналист»; $P(x)$ = «x был в космосе».

Наше суждение можно выразить формулой: $\exists x(S(x) \,\&\, P(x))$.

д) «Некоторые современники динозавров не вымерли».

Введём предикаты $S(x)$ = «x – современник динозавров»; $P(x)$ = «x вымер».

Наше суждение можно выразить формулой: $\exists x(A(x) \,\&\, \neg P(x))$.

Суждения об отношениях выражают определённые отношения между двумя, тремя и большим числом объектов. При переводе этих суждений в формулы используют многоместные предикаты и правила, рассмотренные выше. При переводе отрицаний суждений на язык формул применяется правило переноса квантора через знак отрицания и другие равносильные преобразования.

Пример 2.5.2.

Суждение «Некоторые студенты сдали все экзамены» записать в виде формулы логики предикатов. Построить отрицание данного суждения в виде формулы, не содержащей внешних знаков отрицания. Перевести на естественный язык.

Введём предикаты: $A(x)$ = «x – студент»; $B(y)$ = «y – экзамен», $C(x, y)$ = «x сдал экзамен y». Тогда предложение «Некоторые студенты сдали все экзамены» можно записать в виде следующей формулы:

$$\exists x \forall y(A(x) \& B(y) \rightarrow C(x, y)).$$

Построим отрицание этой формулы, применяя равносильные преобразования:

$$\neg \exists x \forall y(A(x) \& B(y) \supset C(x,y))) \sim \forall x \exists y(\neg\ (A(x) \& B(y) \supset C(x,y)) \sim$$
$$\sim \forall x \exists y(A(x) \& B(y) \& \neg C(x, y)).$$

Это предложение можно прочитать следующим образом:

«Каждый студент не сдал хотя бы один экзаме».

Язык логики предикатов удобен для записи математических предложений: определений, теорем, следствий, в частности, необходимых и достаточных условий. При этом предварительно определяются все необходимые предикаты.

Пример 2.5.3.

Записать на языке логики предикатов следующее определение предела числовой последовательности: «Число a называется пределом числовой последовательности $\{a_n\}$, если для любого положительного числа ε существует такой номер n_0, зависящий от ε, что для всех натуральных чисел n, больших или равных n_0, справедливо неравенство: $|a_n - a| < \varepsilon$».

Введём предикаты: $P(\varepsilon)$ = «$\varepsilon > 0$»; $Q(n)$ = «n – натуральное число»; $R(n, n_0)$ = «$n \geq n_0$»; $S(n, \varepsilon)$ = «$|a_n - a| < \varepsilon$».

Определение предела последовательности может быть записано следующей формулой:

$\forall \varepsilon \exists n_0(\varepsilon) \forall n(P(\varepsilon) \& Q(n) \& Q(n_0) \& R(n, n_0) \rightarrow S(n, \varepsilon))$.

Пример 2.5.4.

Записать в виде формулы логики предикатов великую теорему П. Ферма: «Для любого целого $n > 2$ не существует натуральных чисел x, y, z, удовлетворяющих равенству: $x^n + y^n = z^n$».

Введём предикаты: $N(x)$ = «x – натуральное число»; $M(x)$ = «$x > 2$»; $P(x, y, z, n)$ = «$x^n + y^n = z^n$».

Для любых чисел x, y, z, n условие (посылка) теоремы Ферма есть конъюнкция $N(x) \& N(y) \& N(z) \& N(n) \& M(n)$, а заключение есть $\neg P(x, y, z, n)$. Поэтому теорема Ферма формулируется следующим образом:

$$\forall x \forall y \forall z \forall n(N(x) \& N(y) \& N(z) \& N(n) \& M(n) \rightarrow \neg P(x, y, z, n)).$$

Если теорема имеет вид $\forall x(P(x) \rightarrow Q(x))$, то предикат $Q(x)$ является следствием предиката $P(x)$. При этом предикат $Q(x)$ называется необходимым условием предиката $P(x)$, а предикат $P(x)$ – достаточным условием предиката $Q(x)$.

Пример 2.5.5.

Запишем в виде формулы логики предикатов утверждение: «Если число делится на 6, то оно делится на 3».

Введём предикаты $P(x)$ = «x делится на 6»; $Q(x)$ = «x делится на 3». Наше утверждение формулируется следующим образом: $\forall x(P(x) \rightarrow \rightarrow Q(x))$.

Предикат $P(x)$ (делимость на 6) является достаточным условием предиката $Q(x)$ (делимость на 3). Предикат $Q(x)$ (делимость на 3) является необходимым условием предиката $P(x)$ (делимость на 6).

Установление истинности произвольной формулы исчисления предикатов может быть связано с серьёзными трудностями, имеющими принципиальный характер. Действительно, в общем случае отсутствует конструктивный алгоритм, который бы за конечное, хотя и возможно большое число шагов, позволил бы установить тождественную истинность или ложность произвольной формулы. Хотя, казалось бы, используемый для аналогичной цели в исчислении высказываний алгоритм построения таблицы истинности может быть обобщён для формулы исчисления предикатов, тем не менее, он совершенно не годится в случае предметной области с более чем счётной мощностью. Действительно, в этом случае потребовалось бы по-

строить таблицу с бесконечным числом строк, что уже в принципе не позволит установить истинность произвольной формулы для подобной предметной области.

Поиск выхода из создавшегося положения послужил движущей силой развития математической логики. Были предложены различные формальные системы, в рамках которых удалось разработать алгоритмы (например, метод резолюций), предназначенные для поиска доказательств общезначимости формул исчисления предикатов первого порядка.

3. ЭЛЕМЕНТЫ ТЕОРИИ ВЫВОДА

3.1. Логическое следствие

Задача математической логики – дать принцип рассуждения, т.е. теорию вывода, критерии для решения механическим путём вопроса о том, можно ли некоторую цепочку рассуждений считать правильной. При этом важна только форма высказываний, составляющих цепочку, а не их содержание и смысл. Основной инструмент построения правильного доказательства – тавтологические импликации.

Высказывание B называется логическим следствием высказываний $A_1, A_2, ..., A_m$, если B истинно всякий раз, когда каждое $A_1, A_2, ..., A_m$ истинно. Это записывается так:

$$A_1, A_2, ..., A_m \Rightarrow B \text{, или } A_1, A_2, ..., A_m \mid -B .$$

Высказывания $A_1, A_2, ..., A_m$ называются посылками логического следствия, а высказывание B – заключением.

Кратко это можно сформулировать так: «Пусть $A_1, A_2, ..., A_m$. Тогда B». Большинство теорем в математике имеют именно такую структуру. Доказать теорему – значит доказать, что заключение действительно является логическим слествием посылок. Самый простой способ доказательства логического следствия – составить совместную таблицу истинности посылок и заключения, найти в ней строки, в которых посылки истинны, и убедиться, что заключение в этих строках также истинно.

Пример 3.1.1.

Доказать логическое следствие $A, C, A \& C \Rightarrow \neg B$.

Для доказательства составим таблицу истинности:

A	B	C	A&B$\Rightarrow\neg$C	$\neg B$
0	0	0	1	1
0	0	1	1	1
0	1	0	1	0
0	1	1	1	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	0
1	1	1	0	0

В таблице нашлась только одна строка (третья снизу), в которой все посылки принимают значение истины. Видно, что заключение при этом также истинно. Таким образом, логическое следствие доказано. Этим доказывается любая теорема, которая по форме совпадает с данным логическим следствием. Чтобы подчеркнуть независимость доказательства от содержания высказываний, приведём две формулировки теорем данной логической структуры.

1. $A =$«x делится на 5»; $B =$«x – нечётное»; $C =$«x делится на 4». «Пусть x делится на 5 и на 4. Известно, что нечётные числа, кратные 5, не делятся на 4. Значит, x чётно».

2. $A =$«Треугольник ABC – равнобедренный»; $B =$«Треугольник ABC – прямоугольный»; $C =$«Треугольник ABC – равносторонний».

«Пусть треугольник ABC равнобедренный и равносторонний. Если равнобедренный треугольник является прямоугольным, то он – не равносторонний. Значит, треугольник ABC – не прямоугольный».

Приведём пример нелогичного рассуждения:

$$A \to B, C \to D, A \vee C \Rightarrow B \,\&\, D .$$

Пусть компоненты принимают следующие значения: $A = 1, B = 1, C = 0, D = 0$. В этом случае все посылки истинны, а заключение ложно. Содержательный пример: «Если идёт дождь (A), то возьмём зонт (B). Если светит солнце (C), то наденем футболку (D). Пусть идёт дождь или светит солнце $(A \vee C)$. Тогда возьмём зонт и наденем футболку $(B \,\&\, D)$».

Нет необходимости при доказательстве теорем (логических следствий) каждый раз строить таблицы истинности. Обычно доказательство теоремы $A_1, A_2, ..., A_m \Rightarrow B$ представляет собой цепочку (конечную последовательность) высказываний $E_1, E_2, ..., E_r$, последнее из которых совпадает с доказываемым заключением: $E_r = B$. Какую же цепочку доказательства считать правильной? Представляется оправданным следующее правило.

Любая формула в цепочке $E_1, E_2, ..., E_r$ доказательства теоремы $A_1, A_2, ..., A_m \Rightarrow B$ есть:

а) либо посылка (одно из высказываний $A_1, A_2, ..., A_m$);

б) либо логическое следствие из некоторых предыдущих высказываний.

Нетрудно убедиться, что в цепочку доказательств можно ввести любую тавтологию, а также любое высказывание, логически эквивалентное какому-либо высказыванию цепочки.

Логическое следствие можно построить с помощью набора правил (правил вывода). Математически эти правила основаны на применении тождественно истинных импликаций. В самом деле, если высказывание вида «если A, то B» является тавтологией, ясно, что высказывание B логически следует из высказывания A.

3.2. Основные правила вывода

Непосредственными умозаключениями называются дедуктивные умозаключения, делаемые из одной посылки. К ним в традиционной логике относятся следующие: превращение, обращение, противопоставление предикату и умозаключения по «логическому квадрату».

1. Утверждающий модус (modus ponens) задаётся формулой $((a \to b) \& a) \to b$. Для построения примера воспользуемся интересным высказыванием великого русского педагога К.Д. Ушинского: «Если человек избавлен от физического труда и не приучен к умственному, зверство овладевает им». Использовав это высказывание, построим умозаключение:

«Если человек избавлен от физического труда и не приучен к умственному, то им овладевает зверство».

«Этот человек избавлен от физического труда и не приучен к умственному».

«Этим человеком овладевает зверство».

2. Отрицательный модус (modus tollens) определяется формулой $((a \to b) \& \neg b) \to \neg a$.

Можно строить достоверные умозаключения от отрицания следствия к отрицанию основания. Рассмотрим соответствующее конкретное умозаключение.

Пример 3.2.1.

«Если река выходит из берегов, то вода заливает прилежащие территории».

«Вода реки не залила прилежащие территории».

«Река не вышла из берегов».

3.3. Автоматическое доказательство теорем. Метод резолюций

Наиболее известный классический алгоритм автоматического доказательства теорем называется методом резолюций (МР). Для любого прикладного исчисления предикатов 1-го порядка он даёт ответ «ДА», если Г |- S, и «НЕТ» или не выдает ответа в противном случае.

В основе МР лежит идея доказательства от противного.

Теорема 3.3.1. Если $\Gamma, \neg S \mid -F$, где F противоречие, то Γ |- S.

Пустая формула не является истинной или ложной ни в какой интерпретации и, по определению, является противоречием. В качестве формулы F при доказательстве от противного по МР принято использовать пустую формулу (□).

Пусть имеется множество формул $\Gamma = \{A_1, A_2, \ldots, A_n\}$ и формула B. *Автоматическим доказательством теоремы* называют алгоритм, который проверяет вывод:

$$A_1, A_2, \ldots, A_n \Rightarrow B. \qquad (3.3.1)$$

Выражение (3.3.1) можно прочитать следующим образом:

Если посылки $A_1, A_2, \ldots, A_n$ истинны, то истинно заключение B или если причины $A_1, A_2, \ldots, A_n$ имели место, то будет иметь место следствие B.

Проблема доказательства в логике состоит в том, чтобы установить, что если истинны формулы $A_1, A_2, \ldots, A_n$, то истинна формула B.

В общем случае такой алгоритм построить нельзя. Но для некоторых частных случаев такие алгоритмы существуют. Доказательство теоремы равносильно доказательству общезначимости некоторой формулы. Наиболее эффективно доказательство общезначимости формул осуществляется методом резолюций. Процедура поиска доказательства методом резолюций фактически является процедурой поиска опровержения, т.е. вместо доказательства общезначимости формулы доказывается, что отрицание формулы противоречиво: $A \equiv 1 \leftrightarrow \neg A \equiv 0$.

Введём терминологию, обычно употребляемую при изложении метода резолюций.

Литерой будем называть выражения A или $\neg A$. Литеры A и $\neg A$ называются *контрарными*, а множество $\{A, \neg A\}$ – *контрарной парой*.

Дизъюнкт – это дизъюнкция литер (или элементарная дизъюнкция).

Пример 3.3.1.

$A \vee B \vee C$, $A \vee \neg B$, $\neg A$ – дизъюнкты; $A \vee B \,\&\, C$ – не дизъюнкт.

Дизъюнкт называется *пустым* (обозначается $\square$), если он не содержит литер. Пустой дизъюнкт всегда ложен, так как в нём нет литер, которые могли бы быть истинными при любых наборах переменных.

Рассмотрим применение метода резолюций к исчислению высказываний.

Правилом резолюции называют следующее правило вывода:

$$\frac{A \vee B,\ \neg A \vee C}{B \vee C}. \qquad (3.3.2)$$

Правило (3.3.2) можно также записать в следующем виде:

$$A \vee B,\ \neg A \vee C \to B \vee C. \qquad (3.3.3)$$

Правило резолюций можно доказать, используя равносильности логики высказываний:

$$(A \vee B) \,\&\, (\neg A \vee C) \to (B \vee C) = \neg((A \vee B) \,\&\, (\neg A \vee C)) \vee (B \vee C) =$$
$$= \neg(A \vee B) \vee \neg(\neg A \vee C) \vee (B \vee C) = \neg A \,\&\, \neg B \vee A \,\&\, \neg C \vee (B \vee C) =$$
$$= (\neg A \vee A) \,\&\, (\neg A \vee \neg C) \,\&\, (\neg B \vee A) \,\&\, (\neg B \vee \neg C) \vee (B \vee C) =$$
$$= (\neg A \vee \neg C) \,\&\, (\neg B \vee A) \,\&\, (\neg B \vee \neg C) \vee (B \vee C) = (\neg A \vee \neg C \vee B \vee C) \,\&$$
$$\&\, (\neg B \vee A \vee B \vee C) \,\&\, (\neg B \vee \neg C \vee B \vee C) = 1.$$

Итак, при истинных посылках истинно заключение.

Правило (3.3.2) – единственное правило, применяемое в методе резолюций, что позволяет не запоминать многочисленных аксиом и правил вывода.

Дизъюнкт $B \vee C$ называется *резольвентой* дизъюнктов $A \vee B$ и $\neg A \vee C$ по литере A: $B \vee C = res_A(A \vee B \text{ и } \neg A \vee C)$.

Если дизъюнкты не содержат контрарных литер, то резольвенты у них не существует.

Для дизъюнктов A и $\neg A$ резольвента есть пустой дизъюнкт: $res_A(A, \neg A) = 0$.

Пример 3.3.2.

Пусть $F = A \vee B \vee C$, $G = \neg A \vee \neg B \vee D$.

Тогда:

$res_A(F, G) = B \vee C \vee \neg B \vee D$.

$res_B(F, G) = A \vee C \vee \neg A \vee D$.

$res_C(F, G)$ не существует.

Метод резолюций соответствует методу доказательства от противного. Действительно, условие $A_1, A_2, \ldots, A_n \to B$ равносильно условию $A_1, A_2, \ldots, A_n, \neg B \to 0$. Метод резолюций относится к методам *непрямого вывода*.

Изложим процедуру вывода $A_1, A_2, \ldots, A_n \Rightarrow B$ в виде алгоритма.

Алгоритм построения вывода методом резолюций

Шаг 1. Формулы $A_1, A_2, \ldots, A_n$ и формулу $\neg B$ привести к КНФ.

Шаг 2. Составить множество S дизъюнктов формул $A_1, A_2, \ldots, A_n$ и B.

Шаг 3. Вместо пары дизъюнктов, содержащих контрарные литеры, записать их резольвенту по правилу (3.3.2).

Шаг 4. Процесс продолжаем. Если он заканчивается пустым дизъюнктом, то вывод обоснован.

Изложенный алгоритм называется *резолютивным выводом из S.*

Возможны три случая:

1. Среди множества дизъюнктов нет содержащих контрарные литеры. Это означает, что формула B не выводима из множества формул $A_1, A_2, \ldots, A_n$.

2. В результате очередного применения правила резолюции получен пустой дизъюнкт. Это означает, что формула B выводима из множества формул $A_1, A_2, \ldots, A_n$.

3. Процесс зацикливается, т.е. получаются всё новые и новые резольвенты, среди которых нет пустых. Это ничего не означает.

Пример 3.3.3.

Обосновать вывод $A \to (B \to C), A\&B \Rightarrow C$.

Применим метод резолюций. Для этого нужно проверить вывод: $A \to (B \to C), A\&B, \neg C \Rightarrow 0$.

Будем действовать в соответствии с алгоритмом.

Шаг 1. Нужно привести к КНФ формулы $A \to (B \to C), A\&B, \neg C$.

$A \to (B \to C) \sim \neg A \vee (B \to C) \equiv \neg A \vee \neg B \vee C$.

Формулы $A\&B, \neg C$ уже находятся в КНФ.

Шаг 2. Составим множество S дизъюнктов:

$S = \{\neg A \vee \neg B \vee C, A, B, \neg C\}$.

Шаг 3. Построим резолютивный вывод из S. Для этого выпишем по порядку все дизъюнкты из S:

1) $\neg A \vee \neg B \vee C$;
2) A;
3) B;
4) $\neg C$.

Вместо пары дизъюнктов, содержащих контрарные литеры, запишем их резольвенту (в скобках указаны номера формул, образующих резольвенту):

5) $\neg B \vee C$. (1, 2)

6) C. (3, 5)

7) □. (4, 6)

Вывод заканчивается пустым дизъюнктом, что является обоснованием вывода $A \rightarrow (B \rightarrow C), A\&B \Rightarrow C$.

Пример 3.3.4.

Записать с помощью формул логики высказываний и решить методом резолюций следующую задачу:

«Чтобы хорошо учиться, надо прикладывать усилия. Тот, кто хорошо учится, получает стипендию. В данный момент студент прикладывает усилия. Будет ли он получать стипендию?»

Введём следующие высказывания:

A = «студент хорошо учится».

B = «студент прикладывает усилия»

C = «студент получает стипендию».

Чтобы утвердительно ответить на вопрос задачи: «Будет ли студент получать стипендию?», нужно проверить вывод:

$B \rightarrow A, A \rightarrow C, B \Rightarrow C$.

Будем действовать в соответствии с алгоритмом.

Шаг 1. Нужно привести к КНФ формулы $B \rightarrow A, A \rightarrow C, B, \neg C$.

$B \rightarrow A = \neg B \vee A$,

$A \rightarrow C = \neg A \vee C$.

Формулы B и $\neg C$ уже находятся в КНФ.

Шаг 2. Составим множество S дизъюнктов:

$S = \{\neg B \vee A, \neg A \vee C, B, \neg C\}$.

Шаг 3. Построим резолютивный вывод из S. Сначала перепишем по порядку дизъюнкты из S:

1) $\neg B \vee A$.

2) $\neg A \vee C$.

3) B.

4) $\neg C$.

Затем вместо пары дизъюнктов, содержащих контрарные литеры, запишем их резольвенту:

5) $\neg B \vee C$. (1, 2)

6) C. (3, 5)

7) 0. (4, 6)

Таким образом, на вопрос задачи можно ответить утвердительно: «Студент будет получать стипендию».

Правило резолюций более общее, чем правило *modus ponens* и производные правила, рассмотренные в п. 3.2. Докажем методом резолюций правило *modus ponens*. Необходимо построить вывод:

$A, A \to B \Rightarrow B$.

Построим резолютивный вывод.

$A, \neg A \vee B \Rightarrow B$.

$A, \neg A \vee B, \neg B \Rightarrow 0$.

$S = \{A, \neg A \vee B, \neg B\}$.

1) A.
2) $\neg A \vee B$.
3) $\neg B$.
4) B. (1, 2)
5) 0. (3, 4)

Метод резолюций легко поддаётся алгоритмизации. Это позволяет использовать его в логических языках, в частности в ПРОЛОГе. Недостатком этого метода является необходимость представления формул в КНФ. Кроме того, автоматическое доказательство теорем методом резолюций основано на переборе, и этот перебор может быть настолько большим, что затраты времени на него практически неосуществимы. Эти обстоятельства стимулируют поиски различных модификаций метода резолюций. Метод резолюций позволяет обосновать некоторые другие правила вывода.

Приведём ещё один пример применения метода резолюций, основанного на попарном переборе дизъюнктов.

Пример 3.3.5.

Построим с помощью метода резолюций следующий вывод:

$\neg A \to B, C \vee A, B \to \neg C \Rightarrow A$.

Или, что то же:

$\neg A \to B, C \vee A, B \to \neg C, \neg A \Rightarrow 0$.

Перепишем все посылки в виде дизъюнктов:

$A \vee B, C \vee A, \neg B \vee \neg C, \neg A \Rightarrow 0$.

Выпишем по порядку все посылки и начнём их по очереди склеивать по правилу резолюций:

1) $A \vee B$
2) $C \vee A$
3) $\neg B \vee \neg C$
4) $\neg A$
5) $A \vee \neg C$ (1, 3)

6) B (1, 4)
7) $A \vee \neg B$ (2, 3)
8) C (2, 4)
9) A (2, 5)
10) $\neg C$ (3, 6)
11) $\neg B$ (3, 8)
12) $\neg C$ (4, 5)
13) $\neg B$ (4, 7)
14) 0 (4, 9)

Мы видим, что такая стратегия перебора неэффективна. В данном случае существует более быстрый вывод. Например:

5) B (1, 4)
6) C (2, 4)
7) $\neg B$ (3, 6)
8) □ (5, 7)

К прямым выводам относится также простая контрапозиция, сложная контрапозиция, правило импортации, правило экспортации и некоторые другие правила.

Правило простой контрапозиции имеет следующий вид:

$$\frac{a \to b}{\neg b \to \neg a}.$$

Это правило читается так: «Если a имплицирует b, то отрицание b имплицирует отрицание a». Здесь a и b – переменные, обозначающие произвольные высказывания, или пропозициональные переменные.

Правило сложной контрапозиции имеет следующий вид:

$$\frac{(a \,\&\, b) \to c}{(a \,\&\, \neg c) \to \neg b}.$$

Правило импортации (конъюнктивного объединения условий). Видный математик П.С. Новиков назвал это правило правилом соединения посылок:

$$\frac{a \to (b \to c)}{(a \,\&\, b) \to c}.$$

Это правило читается так: «Если a имплицирует, что b имплицирует c, то a и b имплицируют c».

Правило экспортации (разъединения условий):

$$\frac{(a \,\&\, b) \to c}{a \to (b \to c)}.$$

Это правило читается так: «Если a и b имплицируют c, то a имплицирует, что b имплицирует c». Это правило обратно предыдущему.

4. ЭЛЕМЕНТЫ ТЕОРИИ АЛГОРИТМОВ

4.1. Основные понятия теории алгоритмов

Век бурного развития науки и техники привёл к наличию большого количества устройств (различных машин, роботов, ЭВМ и других), выполняющих различные действия, заданные им каким-либо способом. Различные действия по некоторому плану может выполнять и сам человек. Назовём исполнителем и человека, и устройство, работающих по некоторому плану (алгоритму), и остановимся на понятии алгоритма и предъявляемых к нему требованиях.

Алгоритм является объектом исследования научной дисциплины ***теория алгоритмов***, пограничной между математикой и информатикой. Теория алгоритмов базируется на разделе математики «Математическая логика». Примерами алгоритмов являются:

- правило нахождения наибольшего общего делителя двух натуральных чисел;
- правила умножения двух натуральных чисел в столбик;
- правило построения перпендикуляра к прямой из заданной точки;
- перевод с одного языка на другой;
- правила игры в шахматы или иную игру;
- работа диспетчера, преобразующего поступающую информацию о движении поездов в приказы.

Алгоритм – это раз и навсегда составленная программа, в которой всё заранее предусмотрено. Иначе говоря, алгоритм – это точное воспроизводимое, поддающееся исполнению предписание, определяющее – шаг за шагом, – как надлежит решать данную задачу. Алгоритмом является любое формализованное доказательство математической теоремы, равно как и программа цифровой вычислительной машины, переводящей с одного языка на другой.

Понятие алгоритма связано с конструктивным направлением в математике. Это направление возникло в связи с тем, что теория множеств, разработанная в конце XIX в., обнаружила противоречия – антиномии. Возникла мысль о преодолении этих противоречий за счёт сужения множества рассматриваемых объектов и перехода к так называемым конструктивным объектам.

Говоря о конструктивном объекте, имеют в виду существование способа алгоритмического построения этого объекта. При этом не всякое множество является конструктивным объектом. Конструктивный объект является результатом конструктивного процесса, т.е. результатом применения алгоритма определённого типа.

Теоретико-множественное и конструктивное направления в математике по-разному интерпретируют доказательства теорем, в частности, теорем существования (речь идёт о теоремах, в которых доказывается существование некоторого объекта, обладающего данными свойствами).

Доказательство в конструктивном направлении означает построение алгоритма для нахождения объекта (алгоритмическое доказательство).

Доказательство в теоретико-множественном направлении означает логический вывод из аксиом без явного построения соответствующего алгоритма (неалгоритмическое доказательство). Например, доказательство теоремы М. Ролля, в котором утверждается существование на отрезке [a; b] некоторой точки, обладающей определёнными свойствами, но не даётся алгоритм нахождения такой точки.

Алгоритм – это план действий, состоящий из последовательности понятных исполнителю операций, приводящих к искомому результату. Алгоритм должен состоять из конечного числа шагов и быть понятным исполнителю. Понятие «исполнитель» невозможно определить с помощью какой-либо формализации. Исполнителем может быть человек, группа людей, робот, станок, компьютер, язык программирования и т.д. Одно из принципиальных обстоятельств состоит в том, что исполнитель не вникает в смысл того, что он делает, но получает необходимый результат – исполнитель действует формально, т.е. отвлекаясь от содержания поставленной задачи, и только строго выполняет некоторые правила и инструкции. Это важная особенность алгоритма. Наличие алгоритма формализует процесс решения задачи, исключая рассуждения исполнителя. Целесообразность предусматриваемых алгоритмом действий обеспечивается точным анализом со стороны того, кто составляет этот алгоритм. Интуитивное определение алгоритма хотя и не строгое, но настолько ясное, что не даёт оснований для сомнений в тех случаях, когда речь идёт о найденном алгоритме решения конкретной задачи. Положение существенно меняется, когда возникает алгоритмическая проблема, решение которой не найдено, и требуется установить, имеет ли она решение. Действительно, в этом случае нужно либо доказать существование алгоритма,

либо доказать его отсутствие. В первом случае достаточно дать описание фактического процесса, решающего задачу. При этом достаточно и интуитивного понятия алгоритма, чтобы удостовериться в том, что описанный процесс есть алгоритм. Во втором случае нужно доказать несуществование алгоритма, а для этого нужно точно знать, что такое алгоритм. Между тем для общего понятия алгоритма точного определения до 1930-х гг. не было, и поэтому выработка такого определения стала одной из важных задач современной математики.

Чтобы алгоритм приводил к решению задачи исполнителем, он (алгоритм) должен обладать определёнными свойствами. Перечислим их.

1. Свойство дискретности. Процесс, для которого создан алгоритм, должен быть разбит на последовательность отдельных шагов (действий), образующих прерывную (дискретную) структуру. Только выполнив очередное действие, можно переходить к следующему. Возникающая в результате такого разбиения запись представляет собой упорядоченную совокупность чётко разделённых друг от друга предписаний (директив, команд, операторов), образующих дискретную структуру алгоритма.

2. Свойство определённости (детерминированности). Это свойство означает однозначность толкования каждого шага алгоритма. При одних и тех же исходных данных алгоритм должен каждый раз приводить к повторяющимся результатам. Алгоритм не должен содержать указаний, смысл которых может восприниматься исполнителем неоднозначно, например, «варить до готовности», «солить по вкусу». Кроме того, при исполнении алгоритма не должно возникать вопросов и разночтений относительно того, что выполнять на следующем шаге – не должно возникать потребностей в принятии решений, не предусмотренных алгоритмом.

3. Свойство массовости алгоритма предполагает возможность его использования при любых допустимых исходных данных. Алгоритм составляется для решения класса задач одного типа, а не для решения одной из таких задач.

4. Свойство конечности и результативности. При точном исполнении всех действий алгоритма процесс должен завершаться за конечное число шагов, и при этом должен получаться какой-либо ответ на вопрос решаемой задачи. Вывод о том, что решения не существует – тоже результат.

5. Свойство понятности. У каждого исполнителя имеется своя система команд. Составляя запись алгоритма для определённого исполнителя, следует использовать лишь известные (понятные) этому исполнителю команды.

Алгоритмы, в которых основную роль играют математические действия, называются численными алгоритмами.

Для создания алгоритма используется набор операций $O=\{O_1, O_2, \ldots, O_n\}$.

Операции выполняют над объектами, называемыми ***операндами***. Операнды, объединённые знаком операции, называют операторами.

Операция, выполняемая над двумя операндами, называется двуместной (***бинарной***), над одним оператором – одноместной (***унарной***).

Операторы (как сингулярные, так и унарные) обозначают буквами, например, $A_1, A_2, A_3, \ldots$.

Алгоритм можно представить в виде последовательности операторов $A_1, A_2, A_3, \ldots, A_n$, которые исполняются дискретно в порядке их записи. Такой алгоритм называют ***линейным:***

$$A_1 A_2 A_3 A_4 \ldots A_n . \qquad (4.1.1)$$

Для алгоритма, в котором возможно изменение порядка выполнения операторов в зависимости от итогов на предыдущем шаге, вводятся операции отношения – ***предикаты*** P_i, называемые также ***условными операторами, операторами ветвления***. В зависимости от истинностного значения (Истина или Ложь) предиката P_i, выполняется переход к следующему за P_i оператору или к оператору, указанному в P_i. Алгоритмы с предикатами называют ***разветвленными:***

$$A_1 A_2 A_3 P_1 \overset{7}{\uparrow} A_4 A_5 A_6 A_7 A_8 . \qquad (4.1.2)$$

Если условие P_1 истинное, то после P_1 выполнится оператор A_4, если же P_1 окажется ложным, то произойдёт передача управления оператору A_7. Очевидно, что среди операторов, предшествующих оператору P_1, должен быть оператор или группа операторов, вырабатывающие для P_1 те условия, которые P_1 должен анализировать. С помощью оператора P_i можно организовать группу операторов, выполняющихся многократно. Такая группа именуется ***циклом,*** а сам алгоритм – циклическим:

$$A_1 A_2 A_3 A_4 A_5 P_1 \overset{3}{\uparrow} A_6 A_7 . \qquad (4.1.3)$$

В алгоритме (4.1.3) операторы A_3, A_4, A_5, P_1 могут повторяться многократно, именно они составляют цикл. Внутри цикла должно быть предусмотрено изменение того условия, которое проверяет P_1, иначе цикл будет выполняться бесконечно. Явление бесконечного движения по циклу называют зацикливанием; обязанность составителя алгоритма предусмотреть конечность выполнения операций в цикле.

При работе с алгоритмом рассматривают два этапа работ:

– *составление алгоритма*;

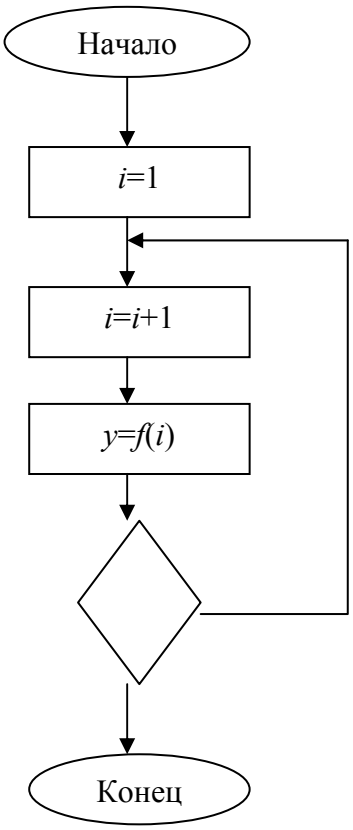
– *реализация алгоритма*.

Алгоритм можно описать различными средствами: на естественном языке, в виде структурной схемы, в виде блок-схемы, в виде последовательности формул.

1. Запись на естественном языке. Описание считается корректным, если воспринимается однозначно исполнителем.

2. В виде структурной схемы, например (4.1.1), (4.1.2), (4.1.3).

3. В виде блок-схемы (ориентированного графа), когда каждый оператор или блок операторов изображают некоторой геометрической фигурой, в ней компактно указывают суть операторов, а последовательность их выполнения указывают стрелками от очередного блока к следующему блоку:



Графический способ описания алгоритма имеет ряд преимуществ благодаря наглядности и высокой «читаемости» алгоритма, а также благодаря явному отображению управления в нём.

Далее мы рассмотрим три конкретные реализации алгоритма: а) математическая модель идеализированного вычислительного устройства (машина Тьюринга); б) рекурсивные функции; в) нормальные алгоритмы Маркова.

4.2. Машина Тьюринга

Машина Тьюринга, описанная А. Тьюрингом в 1936 г., представляет собой теоретическую модель вычислительной машины. Машину Тьюринга (МТ) следует рассматривать как одну из возможных формализаций понятия алгоритм. Её работу можно описать следующим образом. Рассмотрим ленту, разделённую на отдельные ячейки; эта лента потенциально бесконечна влево и вправо. В каждую ячейку можно записать определённый символ некоторого внешнего алфавита A. Машина Тьюринга в любой момент времени находится в определённом состоянии (множество состояний Q – конечно) и указывает на определённую ячейку.

Машина Тьюринга в зависимости от символа, на который она указывает, может записать на его место какой-либо иной символ (он может совпадать со старым), сместиться на один символ влево или вправо, заменить её содержимое или остановиться (часто считают, что машина останавливается автоматически, если нет никакой инструкции, которую она могла бы выполнить). Работа машины Тьюринга определяется её программой.

Программа машины Тьюринга представляет последовательность инструкций (команд), каждая из которых имеет вид $a_i q_j \to a_k q_l I$, где: $a_i, a_k \in \mathrm{A}; q_j, q_l \in Q$, $I = \{R, L, H\}$.

Смысл команды состоит в следующем: если машина находится в состоянии q_j и считывает символ a_i, то она должна записать в текущую ячейку символ a_k, перейти в состояние q_l и сдвинуться вправо (это соответствует букве R), влево (это соответствует букве L) или остановиться (это соответствует букве H).

Считается, что в начале работы машина находится на левом конце ленты в начальном состоянии q_1. Она выполняет операции, которые определяются её программой. Если она в некоторый момент останавливается, результатом работы алгоритма считается последовательность символов, записанных на ленте в момент остановки.

Машина Тьюринга (МТ) – это математическая модель идеализированного вычисляемого устройства. Для построения МТ надо задать:

1. Конечный алфавит $A = \{a_1, ...a_n, \lambda\}$, где λ – пустой символ.
2. Конечное множество внутренних состояний:

$$Q = \{q_0, q_1, ...q_m\},$$

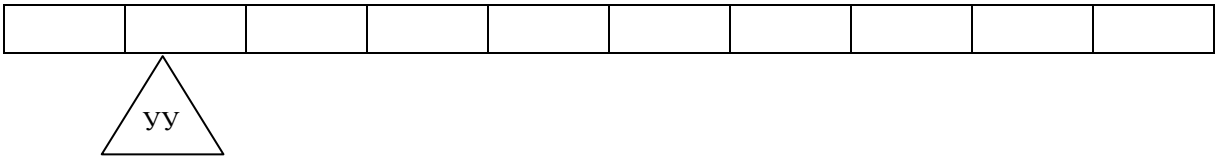
где q_0 – заключительное состояние, q_1 – начальное состояние.

МТ представляет собой:

- Бесконечную ленту, разделённую на ячейки. В каждый момент времени в ячейке записана буква $a_i \in A$. В процессе работы в ячейку может быть записан другой символ $a_j \in A$.

- По ячейкам передвигается управляющее устройство (УУ). В каждый момент времени оно находится напротив какой-то ячейки и имеет некоторое состояние $q_l \in Q$.

Машина действует дискретно, т.е. в определённые моменты времени.



Если в какой-то момент времени УУ воспринимает ячейку, содержащую символ a_i, и МТ находится в состоянии q_l, то МТ может совершить следующие действия:

1. Стереть символ a_i, и записать на его место символ a_j.
2. Переместиться в ячейку слева (Л).
3. Переместиться в ячейку справа (П).
4. Остаться на месте (С).

Эти действия определяются программой.

Таким образом, M=<A,Q, П>.

Программу МТ можно представить в виде последовательности команд вида: $q_l a_i \to a_j D q_k$, где D={Л, П, С}. (Л – переход влево, П – переход вправо, С – остаться на месте).

Программу также можно представить в виде таблицы:

	q_1	q_2	….	q_n
a_1				
a_2				
….		$a_j D q_k$		
a_m				

Пример 4.2.1.

Составить МТ, которая добавляет к машинному слову единицу.

λ	λ	1	1	1	λ	λ

Программа:

$q_1\lambda \to q_1\lambda П$ (Если в воспринимаемой ячейке указан символ λ и МТ находится в состоянии q_1, то состояние не меняется, символ не меняется, УУ сдвигается вправо).

$q_1 1 \to q_2 1 П$ (Если в воспринимаемой ячейке указан символ 1 и МТ находится в состоянии q_1, то это значит, что УУ находится на начале слова, состояние меняется на q_2, символ не меняется, УУ сдвигается вправо).

$q_2 1 \to q_2 1 П$ (Если в воспринимаемой ячейке указан символ 1 и МТ находится в состоянии q_2, то это значит, что УУ передвигается по слову, состояние не меняется, символ не меняется, УУ сдвигается вправо).

$q_2\lambda \to q_0 1 C$ (Если в воспринимаемой ячейке указан символ λ и МТ находится в состоянии q_2, то это значит, что УУ дошло до конца слова, состояние меняется на заключительное, символ меняется на 1, УУ останавливается).

В виде таблицы эту программу можно записать следующим образом (с учётом того, что внутренний алфавит состоит из двух символов $\{q_1, q_2\}$, а внешний алфавит состоит из пустого символа λ и 1):

	q_1	q_2
λ	$q_1\lambda П$	$q_0 1 C$
1	$q_2 1 П$	$q_2 1 П$

Конфигурация МТ (машинное слово) – это слово вида $p_1 q_s a_i p_2$, где:

p_1 – слово в алфавите МТ (может быть пустое),

q_s – внутреннее состояние М,

a_i – воспринимаемый символ,

p_2 – слово в алфавите МТ.

МТ переводит конфигурацию α в конфигурацию β ($\overset{\alpha \to \beta}{T}$), если α имеет вид $p_1 q_j a_i p_2$, β имеет вид $p_1 q_r a_k p_2$, $q_j a_i \to a_k D q_r$ – одна из команд МТ.

Тезис Тьюринга. Любой интуитивный алгоритм может быть реализован с помощью некоторой машины Тьюринга.

Задана программа машины Тьюринга (q_1 – начальное состояние, q_0 – заключительное состояние управляющего устройства, A – внешний алфавит машины). Запишите последовательность конфигураций машины для заданного слова P. Определите, какие действия выполняет эта машина с произвольным (в том числе и пустым) словом W, заданного вида, в алфавите A, если в начальной конфигурации головка устанавливается в заданном положении.

A={0,1}, P=110111, $W = \underbrace{1\ldots1}_{x}0\underbrace{1\ldots1}_{y}$, где $x, y \in \{0,1,2,3,\ldots\}$, и в начальной конфигурации головка устанавливается за последним символом слова.

	q_1	q_2	q_3	q_4	q_5
0	$q_2 0L$	$q_3 1L$	$q_4 0R$	–	$q_0 0C$
1	–	$q_2 1L$	$q_3 1L$	$q_5 0R$	$q_5 1R$

Решение.

Данная машина выполняет «сложение x и y». Последовательность конфигураций машины для слова $P = 110111$ следующая:

1) $P = 1101\underline{1}10$ $q_1 0 \to q_2 0L$, 2) $P = 11011\underline{1}$ $q_2 1 \to q_2 1L$,

3) $P = 1101\underline{1}11$ $q_2 1 \to q_2 1L$, 4) $P = 110\underline{1}111$ $q_2 1 \to q_2 1L$,

5) $P = 11\underline{0}1111$ $q_2 0 \to q_3 1L$, 6) $P = 1\underline{1}11111$ $q_3 1 \to q_3 1L$,

7) $P = \underline{1}111111$ $q_3 1 \to q_3 1L$, 8) $P = \underline{0}111111$ $q_3 0 \to q_4 0R$,

9) $P = 0\underline{1}11111$ $q_4 1 \to q_5 0R$ 10) $P = 00\underline{1}1111$ $q_5 1 \to q_5 1R$,

11) $P = 001\underline{1}111$ $q_5 1 \to q_5 1R$ 12) $P = 0011\underline{1}11$ $q_5 1 \to q_5 1R$,

13) $P = 00111\underline{1}1$ $q_5 1 \to q_5 1R$, 14) $P = 0011111\underline{}$ $q_5 1 \to q_5 1R$,

15) $P = 00111110\underline{}$ $q_5 0 \to q_0 0$.

$W = 1\ldots101\ldots1$, $M(W) = 1\ldots1$, где число единиц равно $(x+y)$. Если слово W – пустое, то оно не изменится и на ленте останется пустое слово.

4.3. Рекурсивные функции

Будем рассматривать только числовые функции, т.е. функции, аргументы и значения которых принадлежат множеству натуральных чисел:

$$\underset{q_1}{\underline{0}} \vdash \underset{q_2}{\underline{0}0} \vdash \underset{q_3}{\underline{0}10} \vdash \underset{q_4}{0\underline{1}0} \vdash \underset{q_5}{00\underline{0}} \vdash \underset{q_0}{00\underline{0}}$$

N (N=0,1,2,…).

Если область определения функции совпадает со множеством $N^n(f: N^n \to N)$, то функция называется всюду определённой, иначе – частично определённой.

Примеры 4.3.1.

$f(x,y) = x + y$ – всюду определённая функция,

$f(x,y) = x - y$ – частично определённая функция, т.к. она определена только для $x \geq y$.

Рекурсивное определение функции – это такое определение, при котором значение функции для данных аргументов определяется значениями той же функции для более простых аргументов или значениями более простых функций.

Примеры 4.3.2.

1. Числа Фибоначчи (1, 1, 2, 3, 5, 8, …) – это последовательность чисел $f(n)$, где $f(0)=1,\ f(1)=1,\ f(n+2)=f(n)+f(n+1)$.

2. Факториал $(n!=1\cdot 2\cdot 3\cdot\cdots\cdot n)$ $f(0)=1,\ f(n+1)=f(n)\cdot(n+1)$.

Рекурсивные функции строятся на основе трёх примитивных (заведомо однозначно понимаемых и реализуемых) функций. Их также называют простейшими.

1. $S(x) = x+1$ – функция следования.

Например, $S(0)=1,\ S(1)=2,\ S(-5)$ – не определена.

2. $O(x) = 0$ – нуль-функция;

Например, $O(0)=0,\ O(1)=0,\ O(-5)$ – не определена.

3. $I_m(x_1, x_2, \cdots, x_n) = x_m,\ (m = 1,2,\cdots,n)$ – функция проектирования (выбора аргумента).

Например, $I_2(1,2,3,4,\cdots,n) = 2$.

С примитивными функциями можно производить различные манипуляции, используя три оператора: суперпозиции, примитивной рекурсии и минимизации.

1. Оператор суперпозиции (подстановки).

Пусть f – m-местная функция, $g_1,\cdots,g_m$ – n-местные операции на множестве N. Оператор суперпозиции S ставит в соответствие операциям f и $g_1,\cdots,g_m$ – n-местную функцию h.

$h(x_1,...,x_n)=f(g_1((x_1,...,x_n),...,g_m(x_1,...,x_n))$.

Примеры 4.3.3.

Используя оператор суперпозиции, можно получить любую константу:

$S(O(x))0+1=1$,

$S(S(O(x)))=0+1+1=0+2=2$,

$S(S\cdots(O(x))\cdots)=0+n$, где число вложений функций следования n.

1) Используя оператор суперпозиции, можно выполнить сдвиг на константу n:

$S(x)=x+1,\ S(S(x))=x+1+1=x+2,\ S(S\cdots(S(x))\cdots)=x+n$.

2. Оператор примитивной рекурсии

Оператор R каждой $(n+2)$-местной операции f и n-местной операции g ставит в соответствие $(n+1)$-местную операцию $h=R(f,g)$, удовлетворяющую следующей схеме:

$h(x_1,...x_n,0)=g(x_1,...x_n)$,

$h(x_1,...,x_n,y+1)=f(x_1,...,x_n,y,h(x_1,...,x_n,y))$.

Для $n=0$ схема примитивной рекурсии имеет вид:

$h(0)=a$, где a – константа,

$h(y+1)=f(y,h(y))$.

Пример 4.3.4.

Вычисление факториала с использованием оператора примитивной рекурсии будет выглядеть следующим образом:

$h(0)=1$,

$h(y+1)=h(y)*(y+1)=f(h(y),y+1)$.

Схема примитивной рекурсии образует процесс построения функции h, при котором на нулевом шаге используется функция g, а на каждом последующем шаге значение функции f от аргументов $x_1,...,x_n$, номера предыдущего шага и значения функции h, вычисленного на предыдущем шаге.

Функция называется примитивно-рекурсивной (ПРФ), если она может быть получена из простейших функций с помощью оператора суперпозиции или оператора примитивной рекурсии.

Примеры 4.3.5.

1) $f(x,y)=x+y$ – примитивно-рекурсивная функция.

Схема примитивной рекурсии:

$x+0=I_1(x)$,

$x+(y+1)=S(x,y)$.

2) $f(x,y)=x*y$ – примитивно-рекурсивная функция.

$x*0=O(x)$,

$x*(y+1)=x*y+x$.

3. Оператор минимизации (μ-оператор)

$f(x_1,...,x_n)=\mu_y[g(x_1,...,x_n,y)=0]$, где y – выделенная переменная.

Работу μ-оператора можно описать следующим образом. Выделяется переменная y, затем фиксируются остальные переменные $x_1,x_2,..x_n$. Значение y последовательно увеличивается, начиная с 0. Значением μ-оператора будет то значение y, при котором функция впервые обратилась в 0.

Если функция не обратилась в 0 или принимает отрицательное значение, то значение μ-оператора считается неопределённым.

Пример 4.3.6.

Пусть $g(x,y)=x-y+3$.

Зафиксируем $x=1$ и будем менять y.

$\mu_y[g(1,1)=0]=3$, т.к. 1-1+3=3

$\mu_y[g(1,2)=0]=2$, т.к. 1-2+3=2

$\mu_y[g(1,3)=0]=1$, т.к. 1-3+3=1

$\mu_y[g(1,4)=0]=0$, т.к. 1-1+3=0

Следовательно, $\mu_y=4$.

Функция $f(x_1,x_2,...,x_n)$ называется частично рекурсивной (ЧРФ), если она может быть получена из простейших функций с помощью конечного числа применений операций суперпозиции, примитивной рекурсии и минимизации.

Пример 4.3.7.

$f(x,y)=x-y$ – частична, т.к. она не определена, если $x<y$. Чтобы сделать эту функцию полностью определённой на множестве натуральных чисел N, рассматривают усечённую разность.

$$x\div y=\begin{cases}x-y, x\geq y\\ 0, x<y\end{cases}$$

Свойства усечённой разности.

1) $x \div 0 = x$

2) $x \div (y+1) = x \div y \div 1$

3) $(x \div y) \div z = x \div (y \div z)$

Докажем, что $f(x,y) = x \div 1$ – примитивно-рекурсивная функция.

Функция $x \div 1$ примитивно рекурсивна, т.к. по схеме примитивной рекурсии:

1) При $x = 0$ $0 \div 1 = 0 = O(x)$,

2) $(x+1) \div 1 = x = I_1(x,y)$.

Таким образом, её можно получить из простейших функций $O(x)$ и $I_m(x_1,\cdots,x_n)$ с помощью оператора простейшей рекурсии R.

Докажем, что $x \div y$ – примитивно-рекурсивная функция.

По схеме примитивной рекурсии:

1) $x \div 0 = x = I_1(x)$,

2) $x \div (y+1) = (\underbrace{x \div y}_{z}) \div 1 = z \div 1$.

Таким образом, функцию $x \div y$ можно получить с помощью операции примитивной рекурсии из функций $I_1(x)$ и $h(x,y,z) = z \div 1$.

Функция $|x-y| = |x \div y| + |y \div x|$ также является примитивно-рекурсивной.

$f(x,y,z) = |x - (z+y)|$ – примитивно-рекурсивная функция.

Функцию $f(x,y) = x - y$ можно получить с помощью оператора минимизации: $f(x,y) = \mu_z[|x-(z+y)| = 0]$.

Следовательно, функция $f(x,y) = x - y$ является частично-рекурсивной функцией.

Всюду определённая частично-рекурсивная функция является общерекурсивной (ОРФ).

Для алгоритмических проблем типичной является ситуация, когда требуется найти алгоритм для вычисления числовой функции $f(x_1,\cdots x_n)$. Числовые функции, значения которых можно вычислить с помощью некоторого алгоритма, называются вычислимыми функциями. Это понятие интуитивно, т.к. интуитивно понятие алгоритма.

Функция $f(x_1,\cdots x_n)$ эффективно вычислима, если существует алгоритм, с помощью которого можно найти $f(k_1,\cdots k_n)$, если известны $k_1,\cdots,k_n$.

Тезис Черча. Всякая эффективно вычислимая функция является частично-рекурсивной функцией.

В формулировку тезиса Черча входит понятие эффективной вычислимости. Поэтому его нельзя ни доказать, ни опровергнуть в математическом смысле.

Частичная рекурсивность – это уточнение понятия вычислимой функции. С его помощью можно уточнять или опровергать вычислимость.

Любая частично-рекурсивная функция является вычислимой по Тьюрингу, и наоборот.

4.4. Нормальные алгоритмы Маркова

Алгоритмическая система Маркова строится по тем же принципам, что и МТ, но носит более простой и интуитивно понятный характер.

Нормальные алгоритмы Маркова (НАМ) представляются нормальной схемой подстановок, которая состоит из совокупности подстановок, расположённых в определённом порядке.

Пусть X – некоторый конечный алфавит, $F(X)$ – слова алфавита, $e \in F(X)$ – пустое слово. Если $p,q \in F(x)$, то выражения $p \to q$ и $p \to .q$ называются формулами подстановки в алфавите X:

$p \to q$ – простая подстановка;

$p \to .q$ – окончательная подстановка.

Символы $\to$ и . не принадлежат X.

Слова p и q могут быть пустыми.

Строка R входит в строку L, если L имеет вид L_1RL_2. Подстановка применима к слову, если строка, соответствующая левой части подстановки, входит в слово. Применение заключается в замене в преобразующем слове левой строки – правой:

$L_1pL_2 \to L_1qL_2$.

Механизм работы НАМ

Дано преобразуемое слово – цепочка символов фиксированного алфавита и нормальная схема подстановок, содержащая фиксированную последовательность простых и заключительных подстановок.

1) Слово всегда просматривается слева направо. Схема подстановок просматривается, начиная с первой подстановки, и, если подстановку можно применить, то она применяется к самому левому вхождению этой строки в преобразуемое слово.

2) Работа алгоритма заканчивается, если:

– ни одна из подстановок не применима,

– использована заключительная подстановка.

Может возникнуть ситуация, когда процесс не закончится никогда. В этом случае считают, что алгоритм не применим к слову.

Пример 4.4.1.

Пусть $X = \{x, y, x\}$.

Нормальная схема подстановок следующая:

$xx \to y$,

$xy \to x$,

$yzy \to x$,

$zz \to z$,

$yy \to x$.

Преобразуемое слово:

$xxxyyyzzz \to yxyyyzzz \to yxyyzzz \to yxyzzz \to yxzzz \to yxzz$.

Например, если X={А,Б,В,Г,…,Я}.

Нормальная схема подстановок следующая:

Х→К

М→Р

КА→ЛОН

РУ→.С

Преобразуемое слово:

МУХА→МУКА→РУКА→РУЛОН→СЛОН

Пример 4.4.2.

Пусть $X = \{a, b\}$.

Нормальная схема подстановок:

$a \to .e$ ($e-$ пустой символ).

$b \to b$.

Преобразуемое слово:

$bbbbbb$ – схема не применима.

Преобразуемое слово:

$abab \to bab$.

Пример 4.4.3.

Пусть $X = \left\{x, y, x^{-1}, y^{-1}\right\}$.

Нормальная схема подстановок:

$xx^{-1} \to e,\ x^{-1}x \to e,\ yy^{-1} \to e,\ y^{-1}y \to e$.

Преобразуемое слово: $xxyxyy^{-1}x^{-1}yx \to xxyxx^{-1}yx \to xxyyx$.

Пример 4.4.4.

Пусть $X = \{x_1, \cdots, x_n\}$.

Нормальная схема подстановок: $x_1 \to e,\ x_2 \to e,\ \cdots,\ x_n \to e$.

Преобразуемое слово переписывается в пустое.

Пусть R и Q нормальные алгоритмы над алфавитом X и $p \in F(x)$. Запись $R(p) \approx Q(p)$ означает, что или оба алгоритма R и Q не применимы к слову p, или оба применимы и $R(p) = Q(p)$.

Два алгоритма R и Q называют эквивалентными относительно алфавита X, если $R(p) \approx Q(p)$ каждый раз, когда $p \in F(x)$ и хотя бы одно из слов $R(p)$ или $Q(p)$ определено и тоже принадлежит $F(x)$.

Если для всех $R(p) \approx Q(p)$, то R и Q называются полностью эквивалентными.

Пусть $X = \{1\}$, а $X' = \{1,*\}$. Тогда всякое натуральное число n можно записать в виде слова в алфавите X': $\overline{0} = 1,\ \overline{1} = 11,\ \overline{2} = 111,\ \overline{3} = 1111, \ldots$.

Поставим в соответствие вектору $(n_1, n_2, ..., n_k)$, где $n_1, n_2, \cdots, n_k$ – натуральные числа, слово в алфавите X' вида $\overline{n_1} * \overline{n_2} * ... * \overline{n_k}$, которое обозначим $\overline{(n_1, n_2, ..., n_k)}$. Например, вектору $\overline{(4,0,2)}$ соответствует слово 11111*1*111.

Пусть $f : N^k \to N$ – некоторая частичная функция и R_f обозначает алгоритм в алфавите X' такой, что $R_f(\overline{k_1, k_2, ...k_n}) = f(k_1, k_2, ...k_n)$ тогда и только тогда, когда хотя бы одна из частей равенства определена. При этом считается, что алгоритм R_f не применим к словам, отличным от вида $\overline{(n_1, n_2, ..., n_k)}$.

Функция f называется частично определимой по Маркову, если существует нормальный алгоритм Q над X', полностью эквивалентный R_f над X'. Если функция определена полностью, то её называют просто вычислимой по Маркову.

Теорема 4.4.1. Простейшие функции $O(x) = 0,\ S(x) = x + 1$ и $I_m(x_1, x_2, \cdots, x_n) = x_m$ вычислимы по Маркову.

Теорема 4.4.2. Всякая частично рекурсивная функция частично рекурсивна по Маркову.

Обратная теорема: всякая частично вычислимая по Маркову функция является частично рекурсивной.

Благодаря компактным правилам замены алгоритм Маркова представляет собой мощное средство описания алгоритмов. На сегодняшний день нет таких алгоритмов над последовательностями знаков, для которых не существовало бы алгоритмов Маркова, выполняющих ту же самую обработку сообщений.

Заметим, что по завершении работы каждый алгоритм должен выдавать некоторые результаты, которые называют выходными данными. Одним из требований, предъявляемых к алгоритмам в настоящее время, является наличие памяти.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Палий, И.А. Дискретная математика : курс лекций / И.А. Палий. – М. : Эксмо, 2008. – 352 с.
2. Шапорев, С.Д. Математическая логика / С.Д. Шапорев. – СПб. : БХВ-Петербург, 2005. – 410 с.
3. Новиков, Ф.А. Дискретная математика для программистов / Ф.А. Новиков. – СПб., 2005. – 304 с.
4. Судоплатов, С.В. Математическая логика и теория алгоритмов : учебник / С.В. Судоплатов, Е.В. Овчинникова. – М. : ИНФРА-М ; Новосибирск : Изд-во НГТУ, 2004. – 224 с.
5. Судоплатов, С.В. Элементы дискретной математики / С.В. Судоплатов, Е.В. Овчинникова. – М. : ИНФРА-М ; Новосибирск : Изд-во НГТУ, 2003. – 360 с.
6. Редькин, Н.П. Дискретная математика : курс лекций для студентов-механиков / Н.П. Редькин. – СПб. : Лань, 2003. – 93 с.
7. Гуц, А.К. Математическая логика и теория алгоритмов : учеб. пособие / А.К. Гуц. – Омск : Наследие. Диалог – Сибирь, 2003. – 108 с.
8. Сдвижков, О.А. Математика на компьютере: Maple 8 / О.А. Сдвижков. – М. : СОЛОН-Пресс, 2003. – 176 с.
9. Ерусалимский, Я.М. Дискретная математика: теория, задачи, приложения / Я.М. Ерусалимский. – М. : Вузовская книга, 2000. – 280 с.
10. Яблонский, С.В. Введение в дискретную математику / С.В. Яблонский. – М. : Наука, 1986. – 384 с.
11. Методические указания к практическим занятиям по дисциплине «Математическая логика и теория алгоритмов» / сост. М.Г. Пантелеев, А.С. Календарев. – СПб. : ГЭТУ, 1997. – 32 с.

ПРИЛОЖЕНИЕ

ЛАБОРАТОРНАЯ РАБОТА

Тема: **Алгебра логики.**

Зададим процедурами программирования наиболее часто используемые элементарные функции двузначной логики и для проверки вычислим по одному их значению.

Конъюнкция:

```
> con := proc(x, y) min(x, y) end;
                    con := proc(x, y) min(x, y) end proc
> con(0,1);
                                    0
```

Дизъюнкция:

```
> diz := proc(x, y) max(x, y) end;
                    diz := proc(x, y) max(x, y) end proc
> diz(0,1);
                                    1
```

Отрицание:

```
> no := proc(x) 1 - x end;
                       no := proc(x) 1 - x end proc
> no(1);
                                    0
```

Сложение по модулю 2:

```
> mo := proc(x) x mod 2 end;
                     mo := proc(x) x mod 2 end proc
> mo(3);
                                    1
```

Импликация:

```
> im := proc(x, y) `if`(x <= y,1,0) end;
                  im := proc(x, y) `if`(x <= y,1,0) end proc
> im(0,1);
                                    1
```

Эквивалентность:

```
> iqv := proc(x, y) `if`(x = y,1,0) end;
                   iqv := proc(x, y) `if`(x = y,1,0) end proc
> iqv(0,1);
                                    0
```

Задача 1. Проверить равенство $x \sim y = (\bar{x} \vee y) \wedge (x \vee \bar{y})$.

Решение. Таблица истинности левой части равенства очевидна. Достаточно составить таблицу истинности правой части:

```
> g := proc(x, y) con(diz(no(x), y), diz(x, no(y))) end;
          g := proc(x, y) con(diz(no(x), y), diz(x, no(y))) end proc
```

> $g(0,0);\ g(0,1);\ g(1,0);\ g(1,1);$

$$1$$
$$0$$
$$0$$
$$1$$

Ответ: равенство верно.

Задача 2. Применяя таблицу истинности, доказать тождественную истинность формулы: $((x \sim y) \wedge \bar{x}) \to \bar{y}$.

Решение.

> $g := proc(x,y)\ im(con(iqv(x,y),no(x)),no(y))\ end;$

$$g := proc(x,y)\ im(con(iqv(x,y),no(x)),no(y))\ end\ \ proc$$

> $g(0,0);\ g(0,1);\ g(1,0);\ g(1,1);$

$$1$$
$$1$$
$$1$$
$$1$$

Тождественная истинность формулы доказана.

Задача 3. Составить таблицу истинности функции $x \oplus (y \vee \bar{z})$.

Решение. Введём заданную функцию:

> $f := proc(x,y,z)\ mo(x + diz(y,no(z)))\ end;$

$$f := proc(x,y,z)\ mo(x + diz(y,no(z)))\ end\ \ proc$$

Таблица истинности функции алгебры логики (булевой функции) трёх переменных содержит 8 строк и 4 столбца. Первые три столбца имеют стандартный вид, соответствующий записи чисел от 0 до 7 в двоичной системе исчисления. Поэтому вводим:

> $a := [0,0,0,0,1,1,1,1];$

$$a := [0,0,0,0,1,1,1,1]$$

> $b := [0,0,1,1,0,0,1,1];$

$$b := [0,0,1,1,0,0,1,1]$$

> $c := [0,1,0,1,0,1,0,1];$

$$c := [0,1,0,1,0,1,0,1]$$

Вывод списка элементов четвёртого столбца – значений заданной функции:

> $p := [seq(f(a[i],b[i],c[i]),i = 1..8)];$

$$p := [1,0,1,1,0,1,0,0]$$

Вывод таблицы истинности:

> $evalm(transpose(matrix([a,b,c,p])));$

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Теперь найдём совершенную дизъюнктивную нормальную форму (СДНФ) последней функции f. Определим степень с булевым показателем:

```
> s := proc(x, y) `if`(y = 1, x, no(x)) end;
```

$$s := proc(x, y)\ `if`(y = 1, x, no(x))\ end\ \ proc$$

Находим СДНФ заданной функции:

```
> add(add(add(f(i, j, k) * (s(x, i) * s(y, j) * s(z, k)), i = 0..1), j = 0..1), k = 0..1);
```

$$(1-x)(1-y)(1-z)+(1-x)y(1-z)+x(1-y)z+(1-x)yz,$$

где произведение следует понимать как конъюнкцию, а сумму как дизъюнкцию.

Аналогично составляется совершенная конъюнктивная нормальная форма.

Учебное издание

ЭЛЕМЕНТЫ МАТЕМАТИЧЕСКОЙ ЛОГИКИ И ТЕОРИИ АЛГОРИТМОВ

Учебно-методические указания

СОСТАВИТЕЛИ:
ФЕТИСОВ Валерий Георгиевич
ФИЛИППЕНКО Виктор Игнатьевич

Ответственный за выпуск Н.В. Ковбасюк

ИД № 06457 от 19.12.01 г. Издательство ЮРГУЭС.
Подписано в печать 27.04.11 г.
Формат бумаги 60х90/16. Усл. п. л. 4,25. Тираж 55 экз. Заказ № 242.

ПЛД № 65-175 от 05.11.99 г.
Типография Издательства ЮРГУЭС.
346500, г. Шахты, Ростовская обл., ул. Шевченко, 147