

МИНОБРНАУКИ РОССИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»  
БОРИСОГЛЕБСКИЙ ФИЛИАЛ  
(БФ ФГБОУ ВО «ВГУ»)

**МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО УЧЕБНОЙ ДИСЦИПЛИНЕ**  
**Математическая логика и теория алгоритмов**

## Методические указания для обучающихся по освоению дисциплины

Приступая к изучению учебной дисциплины, прежде всего обучающиеся должны ознакомиться с программой дисциплины. Электронный вариант рабочей программы размещён на сайте БФ ВГУ.

Знание основных положений, отраженных в рабочей программе дисциплины, поможет обучающимся ориентироваться в изучаемом курсе, осознавать место и роль изучаемой дисциплины в подготовке будущего педагога, строить свою работу в соответствии с требованиями, заложенными в программе.

Основными формами контактной работы по дисциплине являются лекции и практические занятия, посещение которых обязательно для всех студентов.

В ходе лекционных занятий необходимо критически осмысливать предлагаемый материал, задавать вопросы как уточняющего характера, помогающие уяснить отдельные излагаемые положения, так и вопросы продуктивного типа, направленные на расширение и углубление сведений по изучаемой теме, на выявление недостаточно освещенных вопросов, слабых мест в аргументации и т.п.

На практических занятиях необходимо активно участвовать в решении предлагаемых задач, начиная уже с этапа анализа условия и поиска путей решения. Студенту, вызванному для решения задачи к доске, следует подробно комментировать ход решения задачи, а стальным студентам — выполнять основные этапы решения предложенной задачи самостоятельно, но при этом контролируя ход решения на доске.

При подготовке к промежуточной аттестации необходимо повторить пройденный материал в соответствии с программой дисциплины, примерным перечнем вопросов, выносящихся на экзамен. Рекомендуется использовать конспекты лекций и источники, перечисленные в списке литературы в рабочей программе дисциплины, а также ресурсы электронно-библиотечных систем. Необходимо обратить особое внимание на темы учебных занятий, пропущенных по разным причинам. При необходимости можно обратиться за консультацией и методической помощью к преподавателю.

## Методические материалы для обучающихся по освоению теоретических вопросов дисциплины

№	Темы	Рассматриваемые вопросы
1	Алгебра высказываний	Высказывания, логические операции над ними. Совершенные нормальные формы. Логическое следствие. Прямая и обратная теоремы, противоположная и обратная теоремы; закон контрапозиции. Методы математических доказательств. Применение алгебры высказываний к описанию релейно-контактных схем. Исчисление высказываний. Формулы исчисления высказываний. Свойства формализованного исчисления высказываний.
2	Логика предикатов	Предикат. Логические операции над предикатами. Кванторные операции. Формулы логики предикатов, их классификация. Интерпретация формул логики предикатов. Выполнимость и общезначимость формул логики предикатов. Исчисление предикатов.
3	Булевы функции	Булевы функции от одной и двух переменных. Булевы функции от $n$ переменных. Системы булевых функций. Классы Поста. Применение булевых функций к описанию релейно-контактных схем.

4	Машины Тьюринга и поста как модели алгоритма. Нормальные алгоритмы Маркова (НАМ) и частично рекурсивные функции (ЧРФ) как модели алгоритма. Равносильность различных определений алгоритма.	Интуитивное понятие алгоритма. Необходимость уточнения понятия алгоритма. Различные подходы к определению алгоритма. Устройство и принципы работы машины Тьюринга (МТ). Решение задач синтеза и анализа на МТ. Операции с МТ. Функции, вычислимые по Тьюрингу. Тезис Черча-Тьюринга. Устройство и принципы работы машины Поста. Марковские подстановки. НАМы и их применение к словам. Нормально вычислимые функции и принцип нормализации Маркова. Операторы подстановки, примитивной рекурсии, минимизации. Классы ЧРФ, общерекурсивных и примитивно рекурсивных функций и их взаимосвязь. Тезис Черча для ЧРФ. Понятие вычислимой функции. Разрешимые и перечислимые множества. Характеристические функции. Понятие об эффективной нумерации. Нумерация машин Тьюринга.
5	Алгоритмически неразрешимые проблемы Классы сложности P и NP.	Теорема Клини. Недетерминированная машина Тьюринга (НДТ). Основные алгоритмически неразрешимые проблемы. Алгоритмическая сводимость. Характеристики сложности вычислений. Полиномиально и экспоненциально решаемые задачи. Полиномиальная сводимость. Теорема Кука. Основные NP-полные задачи
6	Оптимизационные задачи теории алгоритмов.	Общая характеристика «жадных» алгоритмов. Условия оптимальности «жадных» алгоритмов.

## Методические материалы для обучающихся по подготовке к практическим занятиям

Содержание дисциплины «Математическая логика и теория алгоритмов» представлено следующими разделами:

- алгебра высказываний;
- логика предикатов;
- булевы функции;
- машины Тьюринга и поста как модели алгоритма. Нормальные алгоритмы Маркова (НАМ) и частично рекурсивные функции (ЧРФ) как модели алгоритма. Равносильность различных определений алгоритма;
- алгоритмически неразрешимые проблемы. Классы сложности P и NP;
- оптимизационные задачи теории алгоритмов.

### 1. Алгебра высказываний

#### 1.1. Высказывания и операции над ними

Под *высказыванием* понимают повествовательное предложение, про которое можно однозначно утверждать, истинно оно или ложно. Высказывание относится к основным неопределяемым понятиям математической логики.

Высказывания обозначают заглавными буквами латинского алфавита:

$A, B, C, \dots, X, Y, Z, \dots$

С точки зрения математической логики не важны структура и конкретное содержание высказывания, а важен лишь тот факт, истинно оно или ложно. Значения «истина» и «ложь» называются *истинностными значениями*. В литературе имеются следующие обозначения для истинных высказываний: 1, И, t (от англ. true – истинный), и для ложных высказываний: 0, Л, f (от англ. false – ложный). Из этих обозначений будем использовать 1 и 0.

Считается, что есть непустое первоначально заданное множество высказываний, которые называются элементарными. Из элементарных высказываний можно строить более сложные с помощью *логических операций* или *логических связей*.

*Отрицанием* высказывания  $A$  называется новое высказывание, обозначаемое  $\neg A$  или  $\bar{A}$  (читается «не  $A$ » или «*неверно, что  $A$* »), которое истинно тогда и только тогда, когда само  $A$  ложно.

*Конъюнкцией* двух высказываний  $A$  и  $B$  называется новое высказывание, обозначаемое  $A \wedge B$ , или  $A \& B$ , или  $A \cdot B$  (читается « *$A$  и  $B$* »), которое истинно тогда и только тогда, когда истинны оба высказывания  $A$  и  $B$  одновременно.

*Дизъюнкцией* двух высказываний  $A$  и  $B$  называется новое высказывание, обозначаемое  $A \vee B$  или  $A + B$  (читается « *$A$  или  $B$* »), которое ложно тогда и только тогда, когда ложны оба высказывания  $A$  и  $B$  одновременно.

*Импликацией* двух высказываний  $A$  и  $B$  называется новое высказывание, обозначаемое  $A \rightarrow B$  (читается «*если  $A$ , то  $B$* », «*из  $A$  следует  $B$* »), которое ложно тогда и только тогда, когда высказывание  $A$  истинно, а  $B$  – ложно.

*Эквивалентностью* двух высказываний  $A$  и  $B$  называется новое высказывание, обозначаемое  $A \leftrightarrow B$  (читается « *$A$  эквивалентно  $B$* », « *$A$  тогда и только тогда, когда  $B$* »), которое истинно тогда и только тогда, когда высказывания  $A$  и  $B$  принимают одинаковые значения истинности.

Логические операции удобно задавать с помощью *таблиц истинности*.

*Таблица истинности операции отрицания*

$A$	$\neg A$
0	1
1	0

*Таблица истинности операции конъюнкции*

$A$	$B$	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

*Таблица истинности операции дизъюнкции*

$A$	$B$	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

*Таблица истинности операции импликации*

$A$	$B$	$A \rightarrow B$
0	0	1
0	1	1
1	0	0
1	1	1

*Таблица истинности операции эквивалентности:*

$A$	$B$	$A \leftrightarrow B$
0	0	1
0	1	0
1	0	0

1	1	1
---	---	---

С помощью логических операций из элементарных высказываний можно строить более сложные. При этом если отсутствуют скобки, то операции выполняются в следующем порядке:  $\neg$ ,  $\wedge$ ,  $\vee$ ,  $\rightarrow$ ,  $\leftrightarrow$ .

Пусть из элементарных высказываний  $A$ ,  $B$ ,  $C$  построено сложное высказывание, например,  $(A \rightarrow B) \wedge (\neg C)$ . Если использовать это выражение как схему построения других сложных высказываний, то вместо элементарных высказываний  $A$ ,  $B$ ,  $C$  в него можно подставить *переменные*  $X$ ,  $Y$ ,  $Z$ :

$$(X \rightarrow Y) \wedge (\neg Z) \quad (1)$$

Отличие нового выражения от первоначального в том, что переменные можно заменять как на другие буквы, так и на сложные высказывания, например, если  $X$  заменить на  $U \vee V$ ,  $Y$  – на  $\neg R$ ,  $Z$  – на  $Q \rightarrow P$ , то получим новое высказывание  $((U \vee V) \rightarrow (\neg R)) \wedge (\neg(Q \rightarrow P))$ . Таким образом, выражение (1) следует понимать как формулу – схему конструирования новых высказываний.

Переменные, вместо которых можно подставлять высказывания, называют *пропозициональными переменными*.

Дадим строгое определение формулы исчисления высказываний.

1. Всякая пропозициональная переменная есть формула.
2. Если  $F$  и  $H$  – формулы, то  $(\neg F)$ ,  $(F \wedge H)$ ,  $(F \vee H)$ ,  $(F \rightarrow H)$ ,  $(F \leftrightarrow H)$  – также формулы.
3. Никаких других формул алгебры высказываний, кроме полученных согласно пунктам 1-2 этого определения, нет.

Истинностные значения формул исчисления высказываний можно определять с помощью таблиц истинности.

Если формула содержит  $n$  пропозициональных переменных, каждая из которых может независимо остальных принимать одно из значений «истина» или «ложь», то таблица истинности такой формулы будет содержать  $2^n$  строк.

Формула исчисления высказываний  $F(X_1, X_2, \dots, X_n)$  называется *выполнимой* (*опровержимой*), если она принимает значение «истина» («ложь») на некотором наборе истинностных значений входящих в нее пропозициональных переменных  $X_1, X_2, \dots, X_n$ .

Формула исчисления высказываний  $F(X_1, X_2, \dots, X_n)$  называется *тавтологией* или *тождественно истинной* (*противоречием* или *тождественно ложной*), если она принимает значение «истина» («ложь») на любом наборе истинностных значений входящих в нее пропозициональных переменных  $X_1, X_2, \dots, X_n$ .

Очевидно, что по последнему столбцу в таблице истинности формулы  $F(X_1, X_2, \dots, X_n)$  можно определить, какого рода формулой она является. Если формула является тавтологией, то в последнем столбце будут стоять только значения «1», если противоречием, то, соответственно, только значения «0», если выполнима или опровержима, то будут встречаться как одни, так и другие значения.

**Пример.** Построить таблицу истинности для формулы

$$F = ((X \rightarrow Y) \wedge (\neg Z \vee \neg X)) \rightarrow (Y \vee Z)$$

и определить, является она тавтологией, противоречием, выполнимой или опровержимой.

Таблица истинности формулы F содержит  $2^3 = 8$  строк.

В первых трех столбцах таблицы выпишем всевозможные тройки значений переменных X, Y и Z. В последующих столбцах выпишем логические значения формул  $X \rightarrow Y = P$ ,  $\neg Z$ ,  $\neg X$ ,  $\neg Z \vee \neg X = Q$ ,  $(X \rightarrow Y) \wedge (\neg Z \vee \neg X) = P \wedge Q = S$ ,  $Y \vee Z = R$ ,  $F = S \rightarrow R$ , руководствуясь определениями соответствующих логических операций. Обозначения P, Q, R, S введены для удобства записи, чтобы таблица не была очень громоздкой.

X	Y	Z	$X \rightarrow Y = P$	$\neg Z$	$\neg X$	$\neg Z \vee \neg X = Q$	$P \wedge Q = S$	$Y \vee Z = R$	F
0	0	0	1	1	1	1	1	0	0
0	0	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	1
0	1	1	1	0	1	1	1	1	1
1	0	0	0	1	0	1	0	0	1
1	0	1	0	0	0	0	0	1	1
1	1	0	1	1	0	1	1	1	1
1	1	1	1	0	0	0	0	1	1

Так как в последнем столбце формулы F стоят все значения «1» и «0», то эта формула является выполнимой и опровержимой.

## 1.2. Тавтологии. Основные тавтологии. Равносильность формул. Теорема о замене

Две формулы исчисления высказываний  $F(X_1, X_2, \dots, X_n)$  и  $H(X_1, X_2, \dots, X_n)$  называются *логически равносильными*, если они принимают одинаковые истинностные значения на любых одинаковых наборах истинностных значений входящих в них пропозициональных переменных  $X_1, X_2, \dots, X_n$ .

Обозначение:  $F \cong H$  или  $F \equiv H$ .

Существует тесная связь между понятиями равносильности и тавтологии:

*Формулы  $F(X_1, X_2, \dots, X_n)$  и  $H(X_1, X_2, \dots, X_n)$  являются логически равносильными тогда и только тогда, когда формула  $F \leftrightarrow H$  является тавтологией.*

В алгебре высказываний справедливы следующие основные равносильности, выражающие свойства логических операций.

1.  $P \wedge Q \equiv Q \wedge P$ ;  $P \vee Q \equiv Q \vee P$  – законы коммутативности конъюнкции и дизъюнкции.

2.  $P \wedge (Q \wedge R) \equiv (P \wedge Q) \wedge R$ ;  $P \vee (Q \vee R) \equiv (P \vee Q) \vee R$  – законы ассоциативности конъюнкции и дизъюнкции.

3.  $P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$ ;  $P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$  – законы дистрибутивности конъюнкции и дизъюнкции друг относительно друга.

4.  $P \wedge (Q \vee P) \equiv P$ ;  $P \vee (Q \wedge P) \equiv P$  – законы поглощения.

5.  $\neg(P \wedge Q) \equiv \neg P \vee \neg Q$ ;  $\neg(P \vee Q) \equiv \neg P \wedge \neg Q$  – законы де Моргана.

6.  $\neg(P \wedge Q) \equiv \neg P \vee \neg Q$ ;  $P \wedge P \equiv P$  – законы идемпотентности.
7.  $\neg\neg P \equiv P$  – закон двойного отрицания.
8.  $P \vee \neg P \equiv 1$  – закон исключенного третьего.
9.  $P \wedge \neg P \equiv 0$  – закон отрицания противоречия.
10.  $P \wedge 1 \equiv P$ ;  $P \wedge 0 \equiv 0$ ;  $P \vee 1 \equiv 1$ ;  $P \vee 0 \equiv P$ .
11.  $P \rightarrow Q \equiv \neg P \vee Q$  – закон исключения импликации.
12.  $P \leftrightarrow Q \equiv (P \rightarrow Q) \wedge (Q \rightarrow P) \equiv (\neg P \vee Q) \wedge (\neg Q \vee P)$  – закон исключения эквивалентности.

Учитывая связь между понятиями равносильности и тавтологии, каждый из перечисленных законов можно было бы сформулировать следующим образом.

*Эквивалентность формул, записанных в левой и правой частях каждого закона есть тавтология исчисления высказываний.*

Например, формулы  $(\neg(P \wedge Q)) \leftrightarrow (\neg P \vee \neg Q)$  и  $(P \rightarrow Q) \leftrightarrow (\neg P \vee Q)$  есть тавтологии.

Каждую из рассмотренных выше тавтологий можно воспринимать не как одну отдельную формулу, а как своего рода схему получения формул, если допустить, что одинаковые переменные можно заменять на новые переменные или целые формулы. Это допущение основано на так называемой теореме о замене.

**Теорема о замене.** Если формула  $F$ , содержащая пропозициональную переменную  $X$ , является тавтологией, то замена в формуле  $F$  переменной  $X$  на любую формулу  $H$  снова приводит к тавтологии.

С помощью приведенных выше свойств логических операций над формулами исчисления высказываний можно выполнять *равносильные преобразования*, переходя от левой части соответствующего свойства к правой, или наоборот.

**Пример.** Используя основные законы логических операций, упростить формулу  $F$  исчисления высказываний:

$$F = ((X \rightarrow Y) \wedge (\neg Z \vee \neg X)) \rightarrow (Y \vee Z)$$

1.  $F = ((\neg X \vee Y) \wedge (\neg Z \vee \neg X)) \rightarrow (Y \vee Z)$  [по закону исключения импликации, примененному к формуле  $X \rightarrow Y$ ].

2.  $F \equiv \neg(((\neg X \vee Y) \wedge (\neg Z \vee \neg X))) \vee (Y \vee Z)$  [по закону исключения импликации, примененному к формуле  $((\neg X \vee Y) \wedge (\neg Z \vee \neg X)) \rightarrow (Y \vee Z)$ . Здесь роль условия импликации играет формула  $(\neg X \vee Y) \wedge (\neg Z \vee \neg X)$ , роль заключения импликации – формула  $Y \vee Z$ ].

3.  $F \equiv (\neg(\neg X \vee Y) \vee \neg(\neg Z \vee \neg X)) \vee (Y \vee Z)$  [по закону де Моргана, примененному к формуле  $\neg(((\neg X \vee Y) \wedge (\neg Z \vee \neg X)))$ ].

4.  $F \equiv (\neg\neg X \wedge \neg Y) \vee (\neg\neg Z \wedge \neg\neg X) \vee (Y \vee Z)$  [по закону де Моргана, примененному к формулам  $\neg(\neg X \vee Y)$  и  $\neg(\neg Z \vee \neg X)$ ].

5.  $F \equiv ((X \wedge \neg Y) \vee (Z \wedge X)) \vee (Y \vee Z)$  [по закону двойного отрицания, примененному к формулам  $\neg\neg X$  и  $\neg\neg Z$ ].

6.  $F \equiv (X \wedge \neg Y) \vee (Z \wedge X) \vee Y \vee Z$  [по закону ассоциативности].

7.  $F \equiv (X \wedge \neg Y) \vee (X \wedge Z) \vee Z \vee Y$  [по закону коммутативности].

8.  $F \equiv (X \wedge \neg Y) \vee ((X \wedge Z) \vee Z) \vee Y$  [по закону ассоциативности].

9.  $F \equiv (X \wedge \neg Y) \vee Z \vee Y$  [по закону поглощения, примененному к формуле  $(X \wedge Z) \vee Z$ ].

10.  $F \equiv ((X \wedge \neg Y) \vee Y) \vee Z$  [по законам коммутативности и ассоциативности].

11.  $F \equiv ((X \vee Y) \wedge (\neg Y \vee Y)) \vee Z$  [по закону дистрибутивности].

12.  $F \equiv ((X \vee Y) \wedge 1) \vee Z$  [по закону исключенного третьего].

13.  $F \equiv X \vee Y \vee Z$  [по 10 закону и закону ассоциативности].

Так как все преобразования были равносильными, то получаем:

$$F = ((X \rightarrow Y) \wedge (\neg Z \vee \neg X)) \rightarrow (Y \vee Z) \equiv X \vee Y \vee Z.$$

### 1.3. Понятие логического следования

Формула  $H(X_1, X_2, \dots, X_n)$  называется *логическим следствием* формул  $F_1(X_1, X_2, \dots, X_n), F_2(X_1, X_2, \dots, X_n), \dots, F_k(X_1, X_2, \dots, X_n)$ , если формула  $H(X_1, X_2, \dots, X_n)$  обращается в истинное высказывание на всяком наборе значений пропозициональных переменных  $X_1, X_2, \dots, X_n$ , на котором в истинные высказывания обращается и каждая из формул  $F_1(X_1, X_2, \dots, X_n), F_2(X_1, X_2, \dots, X_n), \dots, F_k(X_1, X_2, \dots, X_n)$ .

Обозначение:  $F_1, F_2, \dots, F_k \models H$ .

Формулы, которые стоят до знака следования называют *условиями* или *посылками* или *гипотезами* логического следствия.

Определить, является ли формула  $H$  логическим следствием формул  $F_1, F_2, \dots, F_k$ , можно, например, по таблице истинности. Нужно построить таблицу, которая содержала бы все формулы, и выбрать из нее только те строки, в которых все формулы-посылки одновременно принимают значение «ИСТИНА». Если в каждой из выбранных строк формула  $H$  также принимает значение «ИСТИНА», то, согласно определению, она будет логически следовать из формул  $F_1, F_2, \dots, F_k$ .

Однако, если формулы зависят от большого числа переменных, построение таблицы истинности нерационально. Поэтому в ряде случаев для проверки того, является ли некоторая формула логическим следствием данных формул, можно воспользоваться признаком логического следствия.

**Теорема (признак логического следствия).** Формула  $H$  будет логическим следствием формулы  $F$  тогда и только тогда, когда формула  $F \rightarrow H$  является тавтологией:

$$F \models H \Leftrightarrow \models F \rightarrow H.$$

**Доказательство**

1. Необходимость. Пусть формула  $H(X_1, X_2, \dots, X_n)$  есть логическое следствие формулы  $F(X_1, X_2, \dots, X_n)$ . Тогда по определению, формула  $H$  принимает значение «ИСТИНА» на тех наборах значений переменных  $X_1, X_2, \dots, X_n$ , на которых формула  $F$  принимает значение «ИСТИНА». Тогда по определению операции импликация, формула  $F \rightarrow H$  всегда будет принимать значение «ИСТИНА», и, следовательно, являться тавтологией.

2. Достаточность. Пусть по условию формула  $F \rightarrow H$  является тавтологией, то есть принимает истинное значение при любых наборах значений входящих в нее переменных. Из таблицы истинности для импликации следует, что если формула  $H$  истинна, то формула  $F$  может быть либо истинной, либо ложной. А значит, всякий раз, когда формула  $F$  истинна,  $H$  – тоже истинна, поэтому  $H$  – логическое следствие  $F$ .

#### 1.4. Совершенные дизъюнктивная и конъюнктивная нормальные формы

Для каждой формулы алгебры высказываний можно построить равносильную ей формулу, содержащую только логические операции конъюнкцию, дизъюнкцию и отрицание. Для этого нужно, используя соответствующие тавтологии (равносильности), раскрыть все имеющиеся в формуле импликации и эквивалентности. Однако таких выражений одной и той же формулы через эти три операции можно построить множество. Среди них некоторые формулы играют в алгебре высказываний особую роль.

*Конъюнктивным (дизъюнктивным) одночленом от переменных  $X_1, X_2, \dots, X_n$*  называется конъюнкция (дизъюнкция) этих переменных или их отрицаний.

Союз «или» в этом определении употребляется в неисключающем смысле, то есть в такой одночлен может входить одновременно и переменная, и ее отрицание, например:

$\neg X \wedge Y \wedge Z$  и  $\neg X \wedge Y \wedge \neg Z \wedge X$  - конъюнктивные одночлены.

$X \vee \neg Y \vee Z$  и  $X \vee \neg Y \vee Z \vee Y \vee \neg Z$  - дизъюнктивные одночлены.

*Дизъюнктивной нормальной формой (ДНФ)* называется дизъюнкция конъюнктивных многочленов.

Аналогично, *конъюнктивной нормальной формой (КНФ)* называется конъюнкция дизъюнктивных многочленов. Например:

$(\neg X \wedge Y \wedge Z) \vee (\neg X \wedge Y \wedge \neg Z \wedge X)$  – ДНФ,

$(X \vee \neg Y \vee Z) \wedge (X \vee \neg Y \vee Z \vee Y \vee \neg Z)$  – КНФ.

Среди множества нормальных форм, которые можно построить для каждой формулы исчисления высказываний, существует такая, которая для данной формулы единственна. Такие формы называют *совершенными*.

Одночлен (конъюнктивный или дизъюнктивный) от переменных  $X_1, X_2, \dots, X_n$  называется *совершенным*, если от каждой пары переменных  $X_i$  и  $\neg X_i$ ,  $i \in \{1, \dots, n\}$  входит одна и только одна из букв.

Нормальная форма (конъюнктивная или дизъюнктивная) называется *совершенной*, если в нее входят лишь совершенные одночлены от переменных  $X_1, X_2, \dots, X_n$ .

Например:

$(\neg X \wedge Y \wedge Z) \vee (\neg X \wedge Y \wedge \neg Z)$  – совершенная дизъюнктивная нормальная форма (СДНФ),

$(X \vee \neg Y \vee Z) \wedge (X \vee \neg Y \vee \neg Z)$  – совершенная конъюнктивная нормальная форма (СКНФ).

**Теорема.** Каждая не тождественно ложная формула алгебры высказываний от  $n$  переменных имеет единственную (с точностью до порядка следования дизъюнктивных членов) совершенную дизъюнктивную нормальную форму.

**Теорема.** Каждая не тождественно истинная формула алгебры высказываний от  $n$  переменных имеет единственную (с точностью до порядка следования конъюнктивных членов) совершенную конъюнктивную нормальную форму.

Данные теоремы дают алгоритм представления формул алгебры высказываний в СДНФ и СКНФ с помощью построения таблицы истинности.

### Правило 1.

Чтобы привести не тождественно ложную формулу к СДНФ, нужно:

- выбрать из таблицы истинности этой формулы все те наборы значений, входящих в нее переменных, на которых формула принимает значение «ИСТИНА»;
- для каждого такого набора построить совершенный конъюнктивный одночлен (СКО), принимающий значение «ИСТИНА» на этом наборе и только на нем;
- полученные совершенные конъюнктивные одночлены соединить знаками дизъюнкции.

### Правило 2.

Чтобы привести не тождественно истинную формулу к СКНФ, нужно:

- выбрать из таблицы истинности этой формулы все те наборы значений, входящих в нее переменных, на которых формула принимает значение «ЛОЖЬ»;
- для каждого такого набора построить совершенный дизъюнктивный одночлен (СДО), принимающий значение «ЛОЖЬ» на этом наборе и только на нем;
- полученные совершенные дизъюнктивные одночлены соединить знаками конъюнкции.

**Пример.** Привести формулу  $F = (X \rightarrow Y) \wedge \neg Z$  к СДНФ и СКНФ

$X$	$Y$	$Z$	$X \rightarrow Y$	$\neg Z$	$F$
<b>0</b>	<b>0</b>	<b>0</b>	1	1	1
0	0	1	1	0	0
<b>0</b>	<b>1</b>	<b>0</b>	1	1	1
0	1	1	1	0	0
1	0	0	0	1	0
1	0	1	0	0	0
<b>1</b>	<b>1</b>	<b>0</b>	1	1	1
1	1	1	1	0	0

1. Строим для формулы  $F = (X \rightarrow Y) \wedge \neg Z$  совершенную дизъюнктивную нормальную форму.

Выберем из таблицы истинности те наборы значений переменных  $X, Y, Z$ , на которых формула принимает значение «ИСТИНА» (они выделены в таблице жирным шрифтом).

Строим СКО для каждого из трех выбранных наборов значений переменных. Так как на наборе  $X = 1, Y = 1, Z = 0$ , то конъюнктивный одночлен на этом наборе примет значение «ИСТИНА» тогда и только тогда, когда переменные  $X$  и  $Y$  войдут в него без отрицания, а переменная  $Z$  – с отрицанием:  $X \wedge Y \wedge \neg Z$  (2).

На наборе  $X = 0, Y = 1, Z = 0$  совершенный конъюнктивный одночлен будет выглядеть следующим образом:  $\neg X \wedge Y \wedge \neg Z$  (3), на наборе  $X = 0, Y = 1, Z = 0$  –  $\neg X \wedge \neg Y \wedge \neg Z$  (4).

Чтобы получить СДНФ для формулы  $F$ , соединим три полученных СКО дизъюнкцией:

$$(X \wedge Y \wedge \neg Z) \vee (\neg X \wedge Y \wedge \neg Z) \vee (\neg X \wedge \neg Y \wedge \neg Z) - \text{СДНФ.}$$

2. Строим для формулы  $F = (X \rightarrow Y) \wedge \neg Z$  совершенную КНФ.

Выберем из таблицы истинности те наборы значений переменных  $X, Y, Z$ , на которых формула принимает значение «ЛОЖЬ» (они выделены в таблице курсивом).

Строим СДО для каждого из трех выбранных наборов значений переменных. Так как на наборе  $X = 1, Y = 1, Z = 1$ , то дизъюнктивный одночлен на этом наборе примет значение «ЛОЖЬ» тогда и только тогда, когда все три переменные войдут в него с отрицанием:  $\neg X \vee \neg Y \vee \neg Z$  (5).

На наборе  $X = 1, Y = 0, Z = 1$ , совершенный дизъюнктивный одночлен будет выглядеть следующим образом:  $\neg X \vee Y \vee \neg Z$  (6), на наборе  $X = 1, Y = 0, Z = 0$ :  $\neg X \vee Y \vee Z$  (7), на наборе  $X = 0, Y = 1, Z = 1$ :  $X \vee \neg Y \vee \neg Z$  (8), и, наконец, на наборе  $X = 0, Y = 0, Z = 1$ :  $X \vee Y \vee \neg Z$  (9).

Чтобы получить СКНФ для формулы  $F$ , соединим пять полученных СДО конъюнкцией:

$$(\neg X \vee \neg Y \vee \neg Z) \wedge (\neg X \vee Y \vee \neg Z) \wedge (\neg X \vee Y \vee Z) \wedge (X \vee \neg Y \vee \neg Z) \wedge (X \vee Y \vee \neg Z)$$

Есть и еще один способ приведения формулы алгебры высказываний к СДНФ и СКНФ, который заключается в использовании равносильных преобразований.

Для приведения формулы к совершенной нормальной форме нужно сначала привести ее к дизъюнктивной или конъюнктивной нормальной форме.

Получим СДНФ

$$1. F = (X \rightarrow Y) \wedge \neg Z \equiv (\neg X \vee Y) \wedge \neg Z \text{ [по закону исключения импликации].}$$

$$2. F \equiv (\neg X \wedge \neg Z) \vee (Y \wedge \neg Z) \text{ [по закону дистрибутивности получили ДНФ].}$$

3.  $F \equiv (\neg X \wedge \neg Z \wedge 1) \vee (Y \wedge \neg Z \wedge 1)$  [не нарушая равносильности формул добавили в каждую скобку единицу, чтобы ввести недостающие переменные].

4.  $F \equiv (\neg X \wedge \neg Z \wedge (Y \vee \neg Y)) \vee (Y \wedge \neg Z \wedge (X \vee \neg X))$  [расписали единицу по закону исключенного третьего].

$$5. F \equiv (\neg X \wedge \neg Z \wedge Y) \vee (\neg X \wedge \neg Z \wedge \neg Y) \vee (Y \wedge \neg Z \wedge X) \vee (Y \wedge \neg Z \wedge \neg X) \text{ [по закону дистрибутивности].}$$

6.  $F \equiv (\neg X \wedge Y \wedge \neg Z) \vee (\neg X \wedge \neg Y \wedge \neg Z) \vee (X \wedge Y \wedge \neg Z) \vee (X \wedge \neg Y \wedge \neg Z)$  – СДНФ [по закону идемпотентности и по коммутативному закону].

Получим СКНФ

1.  $F = (X \rightarrow Y) \wedge \neg Z \equiv (\neg X \vee Y) \wedge \neg Z$  [по закону исключения импликации получили КНФ].

2.  $F \equiv (\neg X \vee Y \vee 0) \wedge (\neg Z \vee 0 \vee 0)$  [не нарушая равносильности формул добавили в каждую скобку нули, чтобы ввести недостающие переменные].

3.  $F \equiv (\neg X \vee Y \vee (Z \wedge \neg Z)) \wedge (\neg Z \vee (X \wedge \neg X) \vee (Y \wedge \neg Y))$  [по закону отрицания противоречия].

4.  $F \equiv (\neg X \vee Y \vee Z) \wedge (\neg X \vee Y \vee \neg Z) \wedge (\neg Z \vee X \vee Y) \wedge (\neg Z \vee X \vee \neg Y) \wedge (\neg Z \vee \neg X \vee Y) \wedge (\neg Z \vee \neg X \vee \neg Y)$  [по закону дистрибутивности].

5.  $F \equiv (\neg X \vee Y \vee Z) \wedge (\neg X \vee Y \vee \neg Z) \wedge (X \vee Y \vee \neg Z) \wedge (X \vee \neg Y \vee \neg Z) \wedge (\neg X \vee \neg Y \vee \neg Z)$  – СКНФ [по закону идемпотентности и по коммутативному закону].

**Пример.** Получить аналитическое выражение для формулы  $F(X, Y, Z)$  алгебры высказываний, которая задана своей таблицей значений.

$X$	$Y$	$Z$	$F$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Данная формула от трех переменных принимает значение 1 тогда и только тогда, когда все ее аргументы принимают одинаковое значение, а в остальных случаях формула принимает значение 0.

Запишем для данной формулы СДНФ, и получим аналитическое выражение для формулы  $F(X, Y, Z) = (X \wedge Y \wedge Z) \vee (\neg X \wedge \neg Y \wedge \neg Z)$ .

Одним из приложений СДНФ и СКНФ является получение аналитического выражения для формулы алгебры высказываний, которая задана своей таблицей значений.

## 2. Логика предикатов

### 2.1. Основные понятия логики предикатов

Предикатом является предложение, о котором нельзя судить, истинно оно или ложно.

$n$ -местным предикатом, заданным на множествах  $M_1, M_2, \dots, M_n$ , называется предложение с  $n$  переменными  $x_1, x_2, \dots, x_n$ , которое при подстановке вместо этих переменных конкретных элементов из множеств  $M_1, M_2, \dots, M_n$  соответственно,

обращается в высказывание, истинное или ложное. Переменные  $x_1, x_2, \dots, x_n$  называются *предметными переменными*.

Обозначение:  $P(x_1, x_2, \dots, x_n)$  – предикат от  $n$  переменных.

С каждым предикатом  $P(x_1, x_2, \dots, x_n)$  связаны три множества:

1. Область определения  $D = M_1 \times M_2 \times \dots \times M_n$  (множества  $M_1, M_2, \dots, M_n$  не обязательно все различны между собой), из которой предметные переменные принимают свои значения.

2. Множество значений – двухэлементное множество  $E = \{1, 0\}$ .

3. Область истинности  $I = \{ \langle a_1, a_2, \dots, a_n \rangle \mid P(a_1, a_2, \dots, a_n) \text{ – истинное высказывание} \}$  – часть области определения, состоящая из тех и только тех наборов значений переменных, на которых предикат  $P(x_1, x_2, \dots, x_n)$  обращается в истинное высказывание.

Над предикатами можно выполнять все те же пять логических операций, что и над высказываниями. Ограничимся определением операций над одноместными предикатами.

*Отрицанием* одноместного предиката  $P(x)$ , заданного на множестве  $D$ , называется новый одноместный предикат  $\overline{P(x)}$ , заданный на том же множестве, обозначаемый  $\neg P(x)$  или  $\overline{P(x)}$  (читается «не  $P(x)$ » или «*неверно, что  $P(x)$* »), который обращается в истинное высказывание тогда и только тогда, когда  $P(x)$  обращается в ложное высказывание.

*Конъюнкцией* двух одноместных предикатов  $P(x)$  и  $Q(x)$ , заданных на множествах  $D_1$  и  $D_2$  соответственно, называется новый двухместный предикат, заданный на множестве  $D_1 \times D_2$ , обозначаемый  $P(x) \wedge Q(x)$  (читается « $P(x)$  и  $Q(x)$ »), который обращается в истинное высказывание тогда и только тогда, когда  $P(x)$  и  $Q(x)$  одновременно обращаются в истинные высказывания.

*Дизъюнкцией* двух одноместных предикатов  $P(x)$  и  $Q(x)$ , заданных на множествах  $D_1$  и  $D_2$  соответственно, называется новый двухместный предикат, заданный на множестве  $D_1 \times D_2$ , обозначаемый  $P(x) \vee Q(x)$  (читается « $P(x)$  или  $Q(x)$ »), который обращается в ложное высказывание тогда и только тогда, когда  $P(x)$  и  $Q(x)$  одновременно обращаются в ложные высказывания.

*Импликацией* двух одноместных предикатов  $P(x)$  и  $Q(x)$ , заданных на множествах  $D_1$  и  $D_2$  соответственно, называется новый двухместный предикат, заданный на множестве  $D_1 \times D_2$ , обозначаемый  $P(x) \rightarrow Q(x)$  (читается «если  $P(x)$ , то  $Q(x)$ »), такой, что  $(\forall a \in D_1)(\forall b \in D_2)$  высказывание  $P(a) \rightarrow Q(b)$  является импликацией высказываний  $P(a)$  и  $Q(b)$ .

*Эквивалентностью* двух одноместных предикатов  $P(x)$  и  $Q(x)$ , заданных на множествах  $D_1$  и  $D_2$  соответственно, называется новый двухместный предикат, заданный на множестве  $D_1 \times D_2$ , обозначаемый  $P(x) \leftrightarrow Q(x)$  (читается « $P(x)$  равносильно  $Q(x)$ »), такой, что  $(\forall a \in D_1)(\forall b \in D_2)$  высказывание  $P(a) \leftrightarrow Q(b)$  является эквивалентностью высказываний  $P(a)$  и  $Q(b)$ .

Все свойства логических операций, доказанные для высказываний справедливы и для операций над предикатами.

Для сложных предикатов справедливы следующие свойства.

Пусть  $P(x)$  и  $Q(x)$  – два предиката, заданные на одном и том же множестве  $D$ .

Область истинности предиката  $\neg P(x)$  совпадает с дополнением области истинности предиката  $P(x)$  до области его определения  $D$ :

$$I_{\neg P} = D - I_P \quad (10)$$

Область истинности предиката  $P(x) \wedge Q(x)$  совпадает с пересечением областей истинности предикатов  $P(x)$  и  $Q(x)$ :

$$I_{P \wedge Q} = I_P \cap I_Q \quad (11)$$

Область истинности предиката  $P(x) \vee Q(x)$  совпадает с объединением областей истинности предикатов  $P(x)$  и  $Q(x)$ :

$$I_{P \vee Q} = I_P \cup I_Q \quad (12)$$

Область истинности предиката  $P(x) \rightarrow Q(x)$  совпадает с областью истинности предиката  $\neg P(x) \vee Q(x)$ , поэтому, учитывая (10) и (12):

$$I_{P \rightarrow Q} = I_{\neg P \vee Q} = I_{\neg P} \cup I_Q \quad (13)$$

Область истинности предиката  $P(x) \leftrightarrow Q(x)$  совпадает с областью истинности предиката  $(\neg P(x) \vee Q(x)) \wedge (\neg Q(x) \vee P(x))$ , поэтому, учитывая (10), (11) и (12):

$$I_{P \leftrightarrow Q} = I_{(\neg P \vee Q) \wedge (\neg Q \vee P)} = (I_{\neg P} \cup I_Q) \cap (I_{\neg Q} \cup I_P) \quad (14).$$

**Пример.** Найти область истинности предиката  $P(x) = (x : 6) \rightarrow (x > 3)$ , заданного на множестве  $Z$ .

Предикат  $P(x) : (x : 6) \rightarrow (x > 3)$  можно рассматривать как импликацию предикатов  $R(x) : (x : 6)$  и  $Q(x) : (x > 3)$ :

$$P(x) = R(x) \rightarrow Q(x).$$

Чтобы найти его область истинности, воспользуемся формулой (13), которая в наших обозначениях примет вид:

$$I_{R \rightarrow Q} = I_{\neg R \vee Q} = I_{\neg R} \cup I_Q \quad (15).$$

Отрицанием предиката  $R(x) = (x : 6)$  является предикат  $\neg R(x) = \neg(x : 6)$ , область истинности которого составляют все целые числа, не кратные 6:

$$I_{\neg R} = \{x \in Z \mid x \text{ не делится на } 6\}.$$

Область истинности предиката  $Q(x) = (x > 3)$  составляют все целые числа, большие 3:

$$I_Q = \{x \in Z \mid x > 3\}.$$

Согласно равенству (15), область истинности предиката  $P(x)$  совпадает с объединением областей истинности предикатов  $\neg R(x)$  и  $Q(x)$ , и, следовательно, состоит из тех и только тех целых чисел, которые либо не делятся на 6, либо больше 3:

$$I_p = \{x \in \mathbb{Z} \mid x \text{ не делится на } 6 \text{ или } x > 3\}.$$

Для предиката  $S(x, y)$  от двух переменных область определения и область истинности состоят из упорядоченных пар предметных переменных  $(x, y)$ , в которых  $x \in M_1, y \in M_2, M_1$  и  $M_2$  – множества, на которых задан предикат  $S(x, y)$ .

Рассмотрим, как найти область истинности сложного двухместного предиката можно по тем же формулам, которые были приведены для одноместных предикатов.

**Пример.** Изобразить графически область истинности предиката от двух переменных:

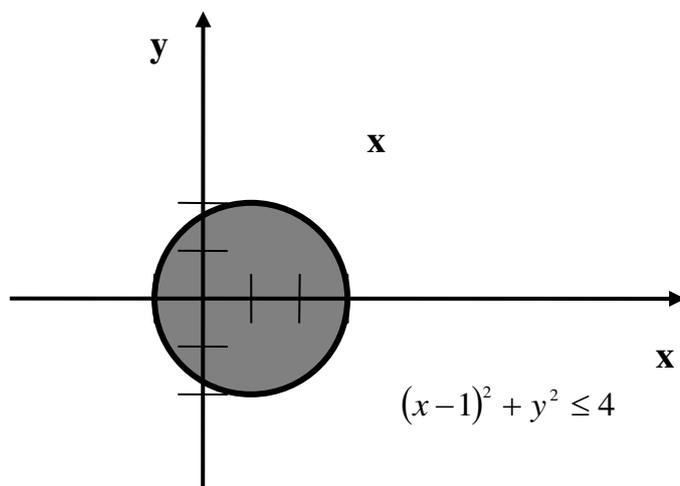
$$S(x, y): ((x-1)^2 + y^2 > 4) \rightarrow (x + y < -1).$$

Предикат  $S(x, y)$ , заданный на множестве  $D = R \times R$ , является импликацией двух предикатов:

$$P(x, y): (x-1)^2 + y^2 > 4 \text{ и } Q(x, y): x + y < -1$$

Отрицанием предиката  $P(x, y)$  будет предикат  $\neg P(x, y): (x-1)^2 + y^2 \leq 4$ , область истинности которого состоит из всех пар действительных чисел  $(x, y)$ , удовлетворяющих неравенству  $(x-1)^2 + y^2 \leq 4$ .

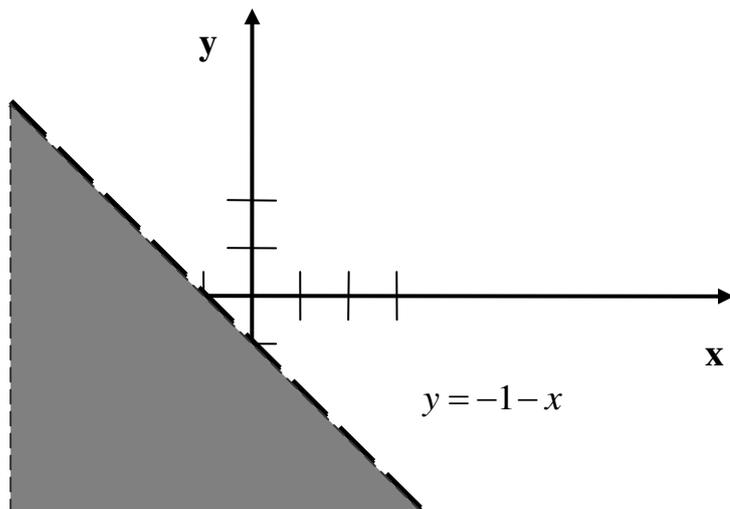
Уравнение  $(x-1)^2 + y^2 = 4$  есть уравнение окружности с центром в точке  $(1, 0)$  и радиусом, равным 2. Точки, удовлетворяющие соответствующему неравенству, лежат, очевидно, внутри и на этой окружности:



**Рис. 1.** Область истинности предиката  $\neg P(x, y)$

Область истинности предиката  $Q(x, y): x + y < -1$  есть множество всех пар действительных чисел, удовлетворяющих неравенству  $x + y < -1$ . Перепишем это неравенство в виде:  $y < -1 - x$ . Множество точек, удовлетворяющих уравнению  $y = -1 - x$  представляет собой прямую, проходящую, например, через точки  $(0, -1)$  и  $(-1, 0)$ . Тогда точки, удовлетворяющие соответствующему неравенству, будут

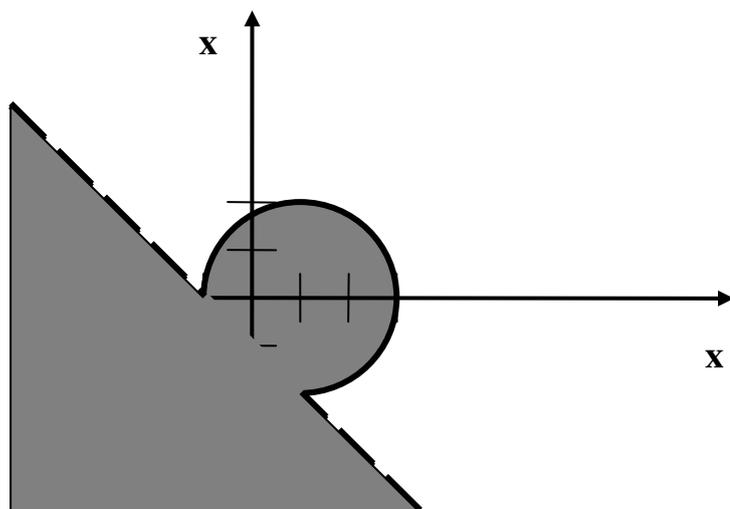
расположены ниже этой прямой. Так как неравенство строгое, то очки, лежащие на самой прямой, ему не удовлетворяют.



**Рис. 2. Область истинности предиката  $Q(x, y)$**

Учитывая формулу (13), получим область истинности предиката  $S(x, y)$ :  

$$I_S = I_{P \rightarrow Q} = I_{\neg P \vee Q} = I_{\neg P} \cup I_Q$$



**Рис. 3. Область истинности предиката  $S(x, y)$**

Предикат  $P(x_1, x_2, \dots, x_n)$ , заданный на области определения  $D$ , называется:

– *тождественно истинным*, если при любой подстановке вместо переменных  $x_1, x_2, \dots, x_n$  конкретных значений из множества  $D$  он обращается в истинное высказывание;

– *тождественно ложным*, если при любой подстановке вместо переменных  $x_1, x_2, \dots, x_n$  конкретных значений из множества  $D$  он обращается в ложное высказывание;

– *выполнимым (опровержимым)* на этом множестве, если существует хотя бы один набор значений переменных  $x_1, x_2, \dots, x_n$ , на котором он обращается в истинное (ложное) высказывание.

Справедливы следующие утверждения.

1. Предикат  $P(x_1, x_2, \dots, x_n)$ , заданный на области определения  $D$ , является тождественно истинным, тогда и только тогда, когда его область истинности совпадает с областью определения:  $I_P = D$ .

2. Предикат  $P(x_1, x_2, \dots, x_n)$ , заданный на области определения  $D$ , является тождественно ложным, тогда и только тогда, когда его область истинности есть пустое множество:  $I_P = \emptyset$ .

3. Предикат  $P(x_1, x_2, \dots, x_n)$ , заданный на области определения  $D$ , выполним тогда и только тогда, когда его область истинности не пуста:  $I_P \neq \emptyset$ .

4. Предикат  $P(x_1, x_2, \dots, x_n)$ , заданный на области определения  $D$ , опровержим тогда и только тогда, когда его область истинности не совпадает с областью определения:  $I_P \neq D$ .

## 2.2. Равносильность и следование предикатов

Два  $n$ -местных предиката  $P(x_1, x_2, \dots, x_n)$  и  $Q(x_1, x_2, \dots, x_n)$ , заданные на одной и той же области определения  $D$ , называются *равносильными*, если предикат  $P(x_1, x_2, \dots, x_n)$  обращается в истинное высказывание на тех и только тех наборах значений переменных  $x_1, x_2, \dots, x_n$ , на которых в истинное высказывание обращается предикат  $Q(x_1, x_2, \dots, x_n)$ .

Очевидно, что предикаты равносильны тогда и только тогда, когда их области истинности совпадают:  $I_P = I_Q$ .

Предикат  $Q(x_1, x_2, \dots, x_n)$ , заданный на области определения  $D$ , называется *логическим следствием* предиката  $P(x_1, x_2, \dots, x_n)$  с той же областью определения, если он обращается в истинное высказывание на всех тех наборах значений переменных  $x_1, x_2, \dots, x_n$ , на которых в истинное высказывание обращается предикат  $P(x_1, x_2, \dots, x_n)$ .

Очевидно, что предикат  $Q(x_1, x_2, \dots, x_n)$  будет логическим следствием предиката  $P(x_1, x_2, \dots, x_n)$  тогда и только тогда, когда область истинности предиката  $P(x_1, x_2, \dots, x_n)$  содержится в области истинности предиката  $Q(x_1, x_2, \dots, x_n)$ :  $I_P \subseteq I_Q$ .

Учитывая приведенную выше классификацию предикатов, можно сформулировать следующие утверждения.

1. Каждые два тождественно истинных (тождественно ложных) предиката с одной и той же областью определения равносильны.

2. Каждый тождественно истинный  $n$ -местный предикат является следствием любого другого  $n$ -местного предиката, заданного на том же множестве.

3. Каждый  $n$ -местный предикат является следствием любого другого тождественно ложного  $n$ -местного предиката, заданного на том же множестве.

**Пример.** Определить, являются ли предикаты  $P(x, y): \sqrt{xy} = \sqrt{x} \cdot \sqrt{y}$  и  $Q(x, y): \frac{\sqrt{x}}{\sqrt{y}} = \sqrt{\frac{x}{y}}$  равносильными или один из них есть следствие другого.

Если рассматривать эти предикаты на множестве всех действительных чисел, то область истинности первого есть множество всех неотрицательных действительных чисел, второго – только положительных действительных чисел ( $y \neq 0$ ). Поэтому  $I_P \subset I_Q$  и  $Q(x, y)$  есть следствие  $P(x, y)$ .

Если же рассматривать эти предикаты на множестве всех положительных действительных чисел, то на этой области определения каждый из предикатов будет тождественно истинным, поэтому они равносильны.

### 2.3. Кванторные операции над предикатами

Известно, что для превращения одноместного предиката в высказывание нужно подставить вместо переменной некоторое ее значение из области определения. Однако существует еще один способ превращения предиката в высказывание. Этот способ связан с применением, так называемых *кванторных операций*.

Символ « $\forall$ » принято называть *квантором общности*, а символ « $\exists$ » – *квантором существования*. Выражение « $(\forall x)(P(x))$ » читается как: «для любого  $x$  выполняется  $P(x)$ », выражение « $(\exists x)(P(x))$ » – «существует  $x$ , для которого выполняется  $P(x)$ ».

Операцией связывания квантором общности называется *правило*, по которому каждому одноместному предикату  $P(x)$ , заданному на множестве  $D$ , сопоставляется высказывание  $(\forall x)(P(x))$ , которое истинно в том и только том случае, когда предикат  $P(x)$  тождественно истинен, и ложно, когда  $P(x)$  – опровержим:

$$(\forall x)(P(x)) = \begin{cases} \text{истинное высказывание, если } P(x) \text{ – тождественно истинен;} \\ \text{ложное высказывание.} \end{cases}$$

Операцией связывания квантором существования называется *правило*, по которому каждому одноместному предикату  $P(x)$ , заданному на множестве  $D$ , сопоставляется высказывание  $(\exists x)(P(x))$ , которое ложно в том и только том случае, когда предикат  $P(x)$  тождественно ложен, и истинно, когда  $P(x)$  – выполним:

$$(\exists x)(P(x)) = \begin{cases} \text{ложное высказывание, если } P(x) \text{ – тождественно ложен;} \\ \text{истинное высказывание.} \end{cases}$$

Например, пусть предикат  $P(x)$ , заданный на множестве натуральных чисел, имеет вид:  $x > 12$ .

Тогда высказывание  $(\forall x)(P(x))$  можно записать как  $(\forall x)(x > 12)$ , а высказывание  $(\exists x)(P(x))$  как  $(\exists x)(x > 12)$ . Очевидно, что первое высказывание будет ложно, так как предикат  $P(x): x > 12$  опровержим на множестве натуральных чисел – существуют натуральные числа, которые меньше либо равны 12. Этот же предикат будет выполним на множестве натуральных чисел, так как существуют натуральные числа, меньшие 12.

Для операций, связанных с кванторами также справедливы законы де Моргана:

1.  $\neg[(\forall x)(P(x))] \equiv (\exists x)(\neg P(x));$
2.  $\neg[(\exists x)(P(x))] \equiv (\forall x)(\neg P(x)).$

С помощью кванторной символики можно записывать на языке логики предикатов различные предложения. Проанализировав строение простых высказываний, можно сделать вывод о том, что содержание любого из них может быть сведено к утверждению о наличии или отсутствии у предметов определенных свойств.

При этом такие утверждения могут относиться не только к отдельным предметам, но и к классам предметов.

Высказывание, в котором утверждается, что все предметы класса обладают или не обладают определенным свойством, называется *общеутвердительным* или *общеотрицательным*.

Высказывание, в котором утверждается, что некоторые предметы класса обладают или не обладают определенным свойством, называется *частноутвердительным* или *частноотрицательным*.

Пусть класс предметов обозначается буквой  $S$ , свойство – буквой  $P$ . Тогда общеутвердительное суждение «Все прямоугольники – параллелограммы» на языке логики предикатов будет выглядеть следующим образом:  $(\forall x)(S(x) \rightarrow P(x))$ . Общеотрицательное суждение «Никакой треугольник не является окружностью» на языке логики предикатов будет записано так:  $(\forall x)(S(x) \rightarrow \neg P(x))$ .

Частноутвердительному суждению «Некоторые функции – периодические» соответствует следующая формула логики предикатов  $(\exists x)(S(x) \wedge P(x))$ . Частноотрицательному суждению «Некоторые ромбы нельзя вписать в окружность» –  $(\exists x)(S(x) \wedge \neg P(x))$ .

Отрицанием общеутвердительного суждения является суждение частноотрицательное, а отрицанием общеотрицательного суждения является суждение частноутвердительное, и наоборот.

**Пример.** Записать символически на языке логики предикатов предложение «**Все математики знают математическую логику**», построить его отрицание и перевести полученное высказывание на русский язык.

Обозначим через  $P(x)$ : « $x$  – математик», через  $Q(x)$ : « $x$  знает математическую логику». Тогда на языке логики предикатов данное предложение будет записано следующим образом:

$$(\forall x)(P(x) \rightarrow Q(x)).$$

Строим отрицание полученного высказывания по первому закону де Моргана:

$$\begin{aligned} \neg((\forall x)(P(x) \rightarrow Q(x))) &\equiv (\exists x)(\neg(P(x) \rightarrow Q(x))) \equiv (\exists x)\neg(\neg P(x) \vee Q(x)) \equiv \\ &\equiv (\exists x)(P(x) \wedge \neg Q(x)) \end{aligned}$$

Переведем полученное высказывание на русский язык, получим предложение «**Некоторые математики не знают математической логики**».

Операции квантификации можно применять к предикатам любой размерности. При этом если исходный предикат был  $n$ -местным то после связывания одной из переменных любым квантором получим предикат, зависящий уже от  $n - 1$  переменной, то есть  $(n - 1)$ -местный предикат. Таким образом, можно связать все переменные кванторами и получить из предиката любой размерности высказывание.

Если в  $n$ -местном предикате кванторами связаны не все переменные, то переменные с кванторами так и называют *связанными*, а без кванторов – *свободными*. При этом значение предиката от связанной переменной уже не зависит.

## 2.4. Формулы логики предикатов

Прежде чем дать строгое определение формулы логики предикатов, определим множество символов, образующих алфавит, из которого формулы будут строиться.

В алфавит логики предикатов входят:

– переменные  $x, y, z, \dots$  и эти же переменные с натуральными индексами.

Природа этих переменных в логике предикатов не рассматривается. Считается, что существует некоторое непустое множество, откуда эти переменные могут принимать свои значения. Множество называют предметной областью, а сами переменные – предметными переменными;

– нульместные предикатные переменные (высказывания)  $P, Q, R, \dots$  и эти же переменные с натуральными индексами;

–  $n$ -местные предикатные переменные ( $n \geq 1$ )  $P(-), Q(-), \dots, P(-, -), Q(-, -), \dots, P(\underbrace{-, -, \dots, -}_{n \text{ раз}}, -), Q(\underbrace{-, -, \dots, -}_{n \text{ раз}}, -), \dots$  с указанием мест в них;

– символы логических и кванторных операций  $\forall, \exists, \neg, \wedge, \vee, \rightarrow, \leftrightarrow$ ;

– вспомогательные символы  $(, )$  – открывающая и закрывающая скобки,  $=$  – знак равенства и  $,$  – запятая.

Дадим строгое определение формулы логики предикатов.

1. Всякая нульместная предикатная переменная есть формула.

2. Если  $P(\underbrace{-, -, \dots, -}_{n \text{ раз}})$   $n$ -местная предикатная переменная, то  $P(x_1, x_2, \dots, x_n)$  –

формула со свободным вхождением переменных  $x_1, x_2, \dots, x_n$ .

3. Если  $F$  и  $H$  – формулы, то  $\neg F, F \wedge H, F \vee H, F \rightarrow H, F \leftrightarrow H$  – тоже формулы. При этом предметные переменные, свободные (связанные) хотя бы в одной из формул  $F, H$ , являются свободными (связанными) и в новых формулах.

4. Если  $F$  – формула,  $x$  – предметная переменная, входящая в  $F$  свободно, то  $(\forall x)(F(x))$  и  $(\exists x)(F(x))$  – формулы, в которых переменная  $x$  связанная, а все остальные предметные переменные, входящие в формулу  $F$  свободно или связано, остаются и в новых формулах соответственно такими же.

5. Никаких других формул логики предикатов, кроме полученных по пунктам 1 – 4 этого определения, нет.

Формулы, определённые в пунктах 1 и 2, называют элементарными или атомарными, остальные формулы – составными.

Формулы логики предикатов являются формальными объектами, построенными по определенным правилам. Смысл они приобретают лишь при замене предметных переменных значениями из выбранного множества (предметной области), а предикатных букв – конкретными предикатами (отношениями) на этом множестве. Такая замена называется интерпретацией.

Интерпретацией формулы  $F$  логики предикатов называется всякая система  $\{D, \omega\}$ , где  $D$  – область интерпретации,  $\omega$  – соответствие, которое каждой предметной переменной формулы  $F$  сопоставляет элемент множества  $D$ , а каждой

$n$ -местной предикатной переменной  $P(x_1, x_2, \dots, x_n)$  –  
 $n$ -местный предикат на множестве  $D$ .

**Пример.** Пусть  $F = (\forall x)(\exists y)(P(x, y))$  – формула логики предикатов. Построим её интерпретацию на множестве  $D = R \times R$  ( $R$  – множество действительных чисел).

Пусть  $\omega$  сопоставляет двухместной предикатной переменной  $P(-, -)$  следующий двухместный предикат на множестве  $D = R \times R$ :  $P(x, y): x > y$

Тогда при данной интерпретации формула  $F = (\forall x)(\exists y)(P(x, y))$  превращается в следующее выражение на выбранной области интерпретации:  
 $F = (\forall x)(\exists y)(x > y), x, y \in R$ .

Данное выражение есть истинное высказывание, утверждающее, что для любого действительного числа существует меньшее него действительное число (множество действительных чисел не ограничено снизу).

Из примера видно, что для одной и той же формулы логики предикатов можно построить бесконечное число интерпретаций, выбирая каждый раз новую область интерпретации и на неё – новые предикаты, соответствующие входящим в формулу предикатным буквам. Соответственно, при некоторых интерпретациях формула может обращаться в истинное высказывание, при других – в ложное.

Если же в исходной формуле не все предметные переменные были связаны кванторами, то при интерпретации она превратится в предикат на выбранной области интерпретации. И только после замены свободных предметных переменных конкретными значениями – в высказывание.

Формула логики предикатов называется *выполнимой (опровержимой)* на множестве  $D$ , если при некоторой подстановке вместо предикатных переменных конкретных предикатов, заданных на этом множестве, она превращается в выполнимый (опровержимый) предикат.

Формула логики предикатов называется *тождественно-истинной (тождественно-ложной)* на множестве  $D$ , если при любой подстановке вместо предикатных переменных любых конкретных предикатов, заданных на этом множестве, она превращается в тождественно-истинный (тождественно-ложный) предикат.

Формула логики предикатов называется *общезначимой или тавтологией (противоречием)*, если при любой подстановке вместо предикатных переменных любых конкретных предикатов, заданных на любых множествах, она превращается в тождественно-истинный (тождественно-ложный) предикат.

Например, формула  $F = (\forall x)(\exists y)(P(x, y))$  из приведенного выше примера является выполнимой. Формула  $F = \neg(\forall x)(P(x)) \leftrightarrow (\exists x)(\neg P(x))$  будет тавтологией. Формула  $\neg P(x) \wedge (\forall y)(P(y))$  – противоречием.

### 3. Булевы функции

#### 3.1. Булевы функции от одного и двух аргументов

*Булевой функцией от одного аргумента* называется отображение  $f$  множества  $\{0, 1\}$  в себя:

$$f: \{0, 1\} \rightarrow \{0, 1\}.$$

Таким образом, аргумент булевой функции может принимать одно из значений: 0 или 1, и сама функция принимает значения в этом же множестве. Нетрудно подсчитать, что всего существует четыре различных булевых функций от одного аргумента, которые представлены в таблице.

Аргумент	Булевы функции			
	$f_0(x)$	$f_1(x)$	$f_2(x)$	$f_3(x)$
0	0	0	1	1
1	0	1	0	1

Функция  $f_0(x) = 0$ , поэтому она так и называется – *тождественный ноль*. Функция  $f_1(x) = x$  называется *тождественной функцией*.  $f_2(x)$  принимает значение, противоположное значению своего аргумента, и поэтому называется *отрицанием*:  $f_2(x) = x'$ . Функция  $f_3(x) = 1$  называется *тождественной единицей*.

*Булевой функцией от двух аргументов* называется отображение  $g$  декартова квадрата множества  $\{0, 1\}$  в это же множество:

$$g: \{0, 1\} \times \{0, 1\} \rightarrow \{0, 1\}.$$

*Булевой функцией от  $n$  аргументов* называется отображение  $f$  декартового произведения множества  $\{0, 1\}$   $n$  раз на себя в это же множество:

$$f: \underbrace{\{0,1\} \times \{0,1\} \times \dots \times \{0,1\}}_{n \text{ раз}} \rightarrow \{0,1\}.$$

Можно доказать, что различных булевых функций от  $n$  аргументов существует ровно  $2^{2^n}$ . Поэтому булевых функций от 2 аргументов будет 16. Перечислим все эти функции.

Аргументы		Булевы функции															
x	y	0	·	→	x	←	y	+	∨	↓	↔	y'	←	x'	→		1
		$g_0$	$g_1$	$g_2$	$g_3$	$g_4$	$g_5$	$g_6$	$g_7$	$g_8$	$g_9$	$g_{10}$	$g_{11}$	$g_{12}$	$g_{13}$	$g_{14}$	$g_{15}$
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Функции в таблице пронумерованы так, что номер функции, записанный в двоичной системе счисления, совпадает со столбцом ее значений. Охарактеризуем каждую из функций.

Нулевая и последняя функции есть *тождественный ноль* и *тождественная единица* соответственно. Третья и пятая тождественно равны своему первому и второму аргументу соответственно, а десятая и двенадцатая - *отрицанию* первого и второго аргумента:

$$g_3(x, y) = x, g_5 = y, g_{10}(x, y) = y', g_{12}(x, y) = x'.$$

Первая функция есть *конъюнкция*, седьмая – *дизъюнкция*, девятая – *эквивалентность*, тринадцатая – *импликация* в том смысле как они определены и для высказываний:

$$g_1(x, y) = x \cdot y, g_7(x, y) = x \vee y, g_9(x, y) = x \leftrightarrow y, g_{13}(x, y) = x \rightarrow y.$$

Одиннадцатая функция носит название *антиимпликации*, так у нее  $y$  есть посылка, а  $x$  – заключение, а вторая и четвертая функции есть *отрицание импликации* и *антиимпликации* соответственно:

$$g_{11}(x, y) = x \leftarrow y = y \rightarrow x, g_2(x, y) = (x \rightarrow y)', g_4(x, y) = (x \leftarrow y)'.$$

Шестая функция называется *сложением по модулю 2* или *суммой Жегалкина* и является отрицанием эквивалентности, восьмая – *стрелкой Пирса* и является отрицанием дизъюнкции, четырнадцатая – *штрихом Шеффера* и является отрицанием конъюнкции:

$$g_6(x, y) = x + y = (x \leftrightarrow y)', g_8(x, y) = x \downarrow y = (x \vee y)',$$

$$g_{14}(x, y) = x | y = (x \cdot y)'.$$

Две булевы функции называются *равными*, если они принимают одинаковые значения на любых одинаковых наборах значений переменных.

*Суперпозицией* булевых функций  $g_1(y_1^1, y_1^2, \dots, y_1^{m_1}), \dots, g_n(y_n^1, y_n^2, \dots, y_n^{m_n})$  в булеву функцию  $f(x_1, x_2, \dots, x_n)$  называется новая булева функция  $F$ , получающаяся из функции  $f(x_1, x_2, \dots, x_n)$  подстановкой вместо (всех или некоторых) аргументов  $x_1, x_2, \dots, x_n$  функций  $g_1, g_2, \dots, g_n$  соответственно.

Например, построим суперпозицию функций  $g_1(x, y) = x \cdot y$  и  $g_8(x, y) = x \downarrow y$  в функцию  $f(x, y) = (x \vee y) \rightarrow x$ :

$$F(x, y) = f(g_1, g_8) = (x \cdot y \vee x \downarrow y) \rightarrow x \cdot y.$$

Для булевых функций выполняются те же законы, что и для соответствующих логических операций над высказываниями: коммутативность, ассоциативность конъюнкции и дизъюнкции, законы поглощения, де Моргана, двойного отрицания и т.д., с помощью которых можно сложные булевы функции приводить к более простому виду.

**Пример.** Упростить выражение для булевой функции:

- $f(x, y, z) = (x' \cdot y' \cdot z \vee x' \cdot y \cdot z) \vee (x \cdot y' \cdot z \vee x \cdot y \cdot z) \vee x \cdot y \cdot z'$  [по законам ассоциативности и коммутативности].

- $f(x, y, z) = (y' \vee y) \cdot x' \cdot z \vee (y' \vee y) \cdot x \cdot z \vee x \cdot y \cdot z'$  [по законам коммутативности и дистрибутивности].

- $f(x, y, z) = (1 \cdot x' \cdot z) \vee (1 \cdot x \cdot z) \vee x \cdot y \cdot z'$  [по закону исключенного третьего].

- $f(x, y, z) = x' \cdot z \vee x \cdot z \vee x \cdot y \cdot z'$  [по свойству операции конъюнкции  $x \cdot 1 = x$ ].

- $f(x, y, z) = (x' \vee x) \cdot z \vee x \cdot y \cdot z'$  [по закону дистрибутивности].

- $f(x, y, z) = 1 \cdot z \vee x \cdot y \cdot z'$  [по закону исключенного третьего].

- $f(x, y, z) = z \vee x \cdot y \cdot z'$  [по свойству операции конъюнкции  $x \cdot 1 = x$ ].

- $f(x, y, z) = (z \vee x \cdot y) \cdot (z \vee z')$  [по закону дистрибутивности].

- $f(x, y, z) = (z \vee x \cdot y) \cdot 1$  [по закону исключенного третьего].

- $f(x, y, z) = z \vee x \cdot y$  [по свойству операции конъюнкции  $x \cdot 1 = x$ ].

Кроме законов коммутативности, ассоциативности и дистрибутивности, здесь использовался закон исключенного третьего:  $x \vee x' = 1$  и свойство операции конъюнкция:  $x \cdot 1 = x$ .

### 3.2. Релейно-контактные схемы

Под *релейно-контактной схемой (РКС)* понимается устройство из проводников и двухпозиционных контактов. Контакты РКС делятся на *замыкающие* и *размыкающие*. Каждый контакт подключен к *реле* (переключателю), причем к одному реле может быть подключено несколько контактов. Когда реле срабатывает, т.е. через него идет ток, то все подключенные к нему замыкающие контакты замыкаются, а размыкающие – размыкаются. При отключении реле все происходит в обратном порядке.

Каждому реле ставится в соответствие своя булева переменная –  $x_1$ , или  $x_2, \dots$ , или  $x_n$ . Каждая переменная, соответствующая некоторому реле, принимает значение 1, когда реле срабатывает и значение 0 при отключении реле. Все замыкающие контакты, подключенные к реле  $x$ , обозначаются также  $x$ , а размыкающие –  $x'$ . Это означает, что при срабатывании реле все замыкающие контакты принимают значение 1, а размыкающие – значение 0. При отключении реле все происходит наоборот.

Тем самым всей РКС ставится в соответствие булева переменная  $y$ , которая зависит от булевых переменных  $x_1, x_2, \dots, x_n$ , обозначающих реле и участвующих в данной схеме. Если при данном наборе состояний реле  $x_1, x_2, \dots, x_n$  вся РКС проводит ток, то переменная  $y$  принимает значение 1, если схема ток не проводит, то переменная  $y$  принимает значение 0.

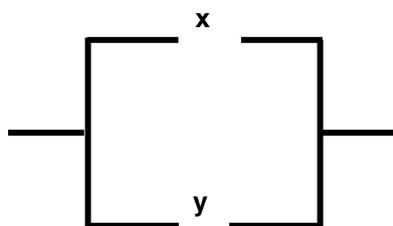
Поскольку каждый набор состояний реле  $x_1, x_2, \dots, x_n$  представляет набор длины  $n$  из нулей и единиц, то каждая РКС задает правило, согласно которому каждому такому набору сопоставляется также либо 0, либо 1. Таким образом, каждая РКС определяет некоторую булеву функцию  $y$  от  $n$  аргументов, которая принимает значение 1 на тех и только тех наборах значений аргументов  $x_1, x_2, \dots, x_n$ , соответствующих тем состояниям реле  $x_1, x_2, \dots, x_n$ , при которых данная схема проводит ток. Такая булева функция  $y = f(x_1, x_2, \dots, x_n)$  называется *функцией проводимости* данной РКС.

Рассмотрим простейшие РКС и найдем их функции проводимости.



Очевидно, что эта схема проводит ток тогда и только тогда, когда оба контакта замкнуты, то есть только тогда, когда обе переменные  $x$  и  $y$  принимают значение 1. Булева функция двух аргументов, удовлетворяющая этому условию, есть конъюнкция, поэтому функция проводимости данной РКС:  $f(x, y) = x \cdot y$ .

Говорят, что *последовательное соединение двух контактов* реализует конъюнкцию соответствующих этим контактам булевых переменных.



Вторая схема проводит ток тогда и только тогда, когда хотя бы один из контактов замкнут, то есть когда хотя бы одна из булевых переменных  $x$  или  $y$  принимает

значение 1. Булева функция двух аргументов, удовлетворяющая этому условию, есть дизъюнкция, поэтому функция проводимости данной РКС:  $f(x, y) = x \vee y$ .

Говорят, что *параллельное соединение двух контактов* реализует дизъюнкцию соответствующих этим контактам булевых переменных.

Для булевых функций может быть доказана следующая теорема:

**Теорема.** Любую булеву функцию можно представить в виде суперпозиции конъюнкции, дизъюнкции и отрицания.

Согласно этой теореме, всякая булева функция может быть реализована в виде РКС.

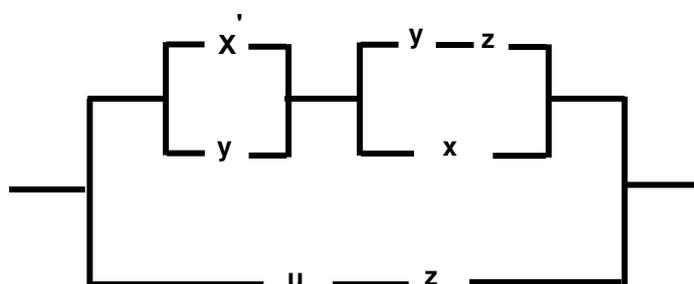
**Пример.** Построить релейно-контактную схему, для которой функция  $f(x, y, z, u) = [(x' \vee y) \cdot (y \cdot z \vee x)] \vee u \cdot z$  являлась бы функцией проводимости.

Функция может быть реализована как параллельное соединение двух веток: первая выражена квадратной скобкой, вторая – произведением  $u \cdot z$ . Квадратная скобка реализуется последовательным соединением двух круглых скобок. Первая круглая скобка есть параллельное соединение двух контактов –  $x'$  и  $y$ , вторая – параллельное соединение контакта  $x$  и последовательного соединения контактов  $y$  и  $z$ .

На рисунке показана реализация булевой функции

$$f(x, y, z, u) = [(x' \vee y) \cdot (y \cdot z \vee x)] \vee u \cdot z$$

в виде релейно-контактной схемы.



**Рис. 4.** Релейно-контактная схема функции  $f(x, y, z, u)$

Согласно теореме о том, что всякую булеву функцию можно представить как суперпозицию конъюнкции, дизъюнкции и отрицания, для каждой булевой функции существует равносильная ей функция, содержащая только эти три операции. Очевидно, что такое представление неоднозначно. Однако среди всех таких представлений существуют так называемые *нормальные представления*, играющие особую роль.

*Конъюнктивным (дизъюнктивным) одночленом от аргументов  $x_1, x_2, \dots, x_n$*  называется булева функция, представляющая конъюнкцию (дизъюнкцию) самих этих аргументов или их отрицаний. Причем союз «или» употребляется не в исключаящем смысле, например,  $f = x \vee y \vee x' \vee z$  – дизъюнктивный одночлен от аргументов  $x, y, z$ .

*Дизъюнктивной нормальной формой* называется дизъюнкция конъюнктивных одночленов.

*Конъюнктивной нормальной формой* называется конъюнкция дизъюнктивных одночленов.

Например,  $f(x, y, z) = (x' \vee y \vee z) \cdot (x \vee y \vee x' \vee z) \cdot (x \vee y \vee z')$  – конъюнктивная нормальная форма булевой функции от аргументов  $x, y, z$ .

Очевидно, что всякая булева функция обладает как конъюнктивной, так и дизъюнктивной нормальной формой, причем этих форм существует неограниченно много. Среди множества всех таких форм существует одна, как конъюнктивная, так и дизъюнктивная, которая для данной функции единственна. Это *совершенная конъюнктивная (дизъюнктивная) нормальная форма*.

Одночлен (конъюнктивный или дизъюнктивный) от аргументов  $x_1, x_2, \dots, x_n$  называется *совершенным*, если в него от каждой пары букв  $x_i$  и  $x_i'$  входит только одна буква.

Нормальная форма от аргументов  $x_1, x_2, \dots, x_n$  называется *совершенной*, если в нее входят лишь совершенные одночлены от этих аргументов.

Например,  $f(x, y, z) = (x' \vee y \vee z) \cdot (x \vee y \vee z)$  – совершенная конъюнктивная нормальная форма булевой функции от аргументов  $x, y, z$ .

**Теорема.** Для каждой булевой функции от  $n$  аргументов, тождественно не равной 0, существует и притом единственная СДНФ.

**Теорема.** Для каждой булевой функции от  $n$  аргументов, тождественно не равной 1, существует и притом единственная СКНФ.

Существуют различные способы приведения булевой функции к СДНФ и СКНФ: с помощью равносильных преобразований и с помощью таблицы истинности.

**Пример.** Для функции  $f(x, y, z) = [(x' \vee y) \cdot (y \cdot z \vee x)] \vee x \cdot z$  построить СКНФ и СДНФ двумя способами.

Рассмотрим первый способ (с помощью равносильных преобразований).

### Получим СДНФ

1.  $f(x, y, z) = (x' \cdot (y \cdot z \vee x) \vee y \cdot (y \cdot z \vee x)) \vee x \cdot z$  [по закону дистрибутивности].

2.  $f(x, y, z) = x' \cdot y \cdot z \vee x' \cdot x \vee y \cdot y \cdot z \vee y \cdot x \vee x \cdot z$  [по закону дистрибутивности].

3.  $f(x, y, z) = x' \cdot y \cdot z \vee 0 \vee y \cdot z \vee y \cdot x \vee x \cdot z$  [по свойству  $x \cdot x' = 0$  и закону идемпотентности].

4.  $f(x, y, z) = x' \cdot y \cdot z \vee y \cdot z \vee y \cdot x \vee x \cdot z$  [по свойству  $x \vee 0 = x$ ].

Первый конъюнктивный одночлен является совершенным, а остальные три – нет, так как в каждом из них нет какого-либо аргумента (или его отрицания). Приведем эти одночлены к совершенной форме.

1.  $y \cdot z = y \cdot z \cdot 1$  [по свойству  $x \cdot 1 = x$ ].

2.  $y \cdot z = y \cdot z \cdot 1 = y \cdot z \cdot (x \vee x')$  [по закону исключенного третьего  $x \vee x' = 1$ ].

3.  $y \cdot z = y \cdot z \cdot 1 = y \cdot z \cdot (x \vee x') = y \cdot z \cdot x \vee y \cdot z \cdot x'$  [по дистрибутивному закону].

Получили два совершенных конъюнктивных одночлена  $y \cdot z \cdot x, y \cdot z \cdot x'$

Аналогично:  $y \cdot x = y \cdot x \cdot 1 = y \cdot x \cdot (z \vee z') = y \cdot x \cdot z \vee y \cdot x \cdot z'$ ;

$x \cdot z = x \cdot z \cdot 1 = x \cdot z \cdot (y \vee y') = x \cdot z \cdot y \vee x \cdot z \cdot y'$ .

Тогда для функции  $f$  получим (с учетом коммутативного закона):

$f(x, y, z) = x \cdot y \cdot z \cdot \vee x' \cdot y \cdot z \vee x \cdot y \cdot z \vee x \cdot y \cdot z' \vee x \cdot y \cdot z \vee x \cdot y' \cdot z$ .

Применяем закон идемпотентности к конъюнкции  $x \cdot y \cdot z$  и получаем СДНФ:  
 $f(x, y, z) = x \cdot y \cdot z \cdot \vee x' \cdot y \cdot z \vee x \cdot y \cdot z' \vee x \cdot y' \cdot z$

### Получим СКНФ

1.  $f(x, y, z) = (x' \vee y \vee x \cdot z) \cdot (y \cdot z \vee x \vee x \cdot z)$  [по дистрибутивному закону].

2.  $f(x, y, z) = ((x' \vee x \cdot z) \vee y) \cdot (y \cdot z \vee (x \vee x \cdot z))$  [по ассоциативному закону и по коммутативному закону].

3.  $f(x, y, z) = ((x' \vee x) \cdot (x' \vee z) \vee y) \cdot (y \cdot z \vee (x \vee x \cdot z))$  [по дистрибутивному закону].

4.  $f(x, y, z) = ((x' \vee x) \cdot (x' \vee z) \vee y) \cdot (y \cdot z \vee x)$  [по закону поглощения].

5.  $f(x, y, z) = ((x' \vee z) \vee y) \cdot (y \cdot z \vee x)$  [по закону исключенного третьего].

6.  $f(x, y, z) = (x' \vee z \vee y) \cdot (y \vee x) \cdot (z \vee x)$  [по ассоциативному закону и дистрибутивному закону].

Первый конъюнктивный одночлен является совершенным, а остальные два – нет, так как в каждом из них нет какого-либо аргумента (или его отрицания). Приведем эти одночлены к совершенной форме.

1.  $y \vee x = x \vee y$  [по коммутативному закону].

2.  $y \vee x = x \vee y = x \vee y \vee 0$  [по свойству  $x \vee 0 = x$ ].

3.  $y \vee x = x \vee y = x \vee y \vee 0 = x \vee y \vee (z \cdot z')$  [по свойству  $x \cdot x' = 0$ ].

4.  $y \vee x = x \vee y = x \vee y \vee 0 = x \vee y \vee (z \cdot z') = (x \vee y \vee z) \cdot (x \vee y \vee z')$  [по дистрибутивному закону].

Аналогично  $x \vee z = x \vee z \vee 0 = x \vee z \vee (y \cdot y') = (x \vee z \vee y) \cdot (x \vee z \vee y')$ .

Тогда для функции  $f$  получим СКНФ (с учетом закона поглощения и коммутативного закона):

$f(x, y, z) = (x' \vee y \vee z) \cdot (x \vee y \vee z) \cdot (x \vee y \vee z') \cdot (x \vee y' \vee z)$

Рассмотрим второй способ (с помощью таблицы истинности)

Строим для функции  $f(x, y, z) = [(x' \vee y) \cdot (y \cdot z \vee x)] \vee x \cdot z$  таблицу истинности.

Введем обозначения:  $A = x' \vee y$ ,  $B = y \cdot z$ ,  $C = y \cdot z \vee x$ ,  $D = (x' \vee y) \cdot (y \cdot z \vee x)$ ,  
 $E = x \cdot z$

x	y	z	x'	A	B	C	D	E	$f(x, y, z)$
0	0	0	1	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	1	0	1	1	0	0	0	0	0
0	1	1	1	1	1	1	1	0	1
1	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	1	0	1	1
1	1	0	0	1	0	1	1	0	1
1	1	1	0	1	1	1	1	1	1

Чтобы для функции  $f$  построить СДНФ, нужно:

– выбрать все те наборы значений ее аргументов, на которых она принимает значение 1;

– для каждого такого набора выписать совершенный конъюнктивный одночлен, принимающий значение 1 на этом наборе и только на нем;

– полученные совершенные конъюнктивные одночлены соединить дизъюнкцией.

Чтобы для функции  $f$  построить СКНФ, нужно:

– выбрать все те наборы значений ее аргументов, на которых она принимает значение 0;

– для каждого такого набора выписать совершенный дизъюнктивный одночлен, принимающий значение 0 на этом наборе и только на нем;

– полученные совершенные дизъюнктивные одночлены соединить конъюнкцией.

Применим эти правила к заданной функции.

### **СДНФ.**

Значение 1 функция  $f(x, y, z) = [(x' \vee y) \cdot (y \cdot z \vee x)] \vee x \cdot z$  принимает на наборах значений аргументов  $x, y, z$ , расположенных в четвертой, шестой, седьмой и восьмой строках таблицы истинности. Для каждого из этих наборов строим совершенный конъюнктивный одночлен, принимающий значение 1 на этом наборе и только на нем и соединяем их дизъюнкцией.

Так как в четвертой строке таблицы истинности  $x$  принимает значение 0, а  $y$  и  $z$  – 1, то чтобы конъюнкция трех аргументов была истина на этом наборе и только на нем,  $x$  должен войти в нее со знаком отрицания, а  $y$  и  $z$  – без отрицания. Аналогично рассуждаем для остальных наборов значений аргументов.

$$\text{СДНФ: } f(x, y, z) = x \cdot y \cdot z \vee x' \cdot y \cdot z \vee x \cdot y \cdot z' \vee x \cdot y' \cdot z$$

Аналогично строим СКНФ.

$$\text{СКНФ: } f(x, y, z) = (x' \vee y \vee z) \cdot (x \vee y \vee z) \cdot (x \vee y \vee z') \cdot (x \vee y' \vee z)$$

Сравнивая с выражениями, полученными первым способом, видим, что они одинаковы с точностью до порядка следования совершенных одночленов. Используя СДНФ и СКНФ можно построить релейно-контактную схему с заданными условиями работы.

#### 4. Машина Тьюринга

В 1937 году английский инженер и математик А. Тьюринг сформулировал понятие алгоритма в виде абстрактной математической машины, которая по имени своего создателя получила название *машины Тьюринга (МТ)*.

Машину Тьюринга можно представить себе в виде некоторого автоматического устройства, через считывающую головку которого проходит бесконечная в обе стороны лента, разбитая на ячейки. В каждый момент времени устройство, находясь в определенном состоянии, обзревает содержимое только одной ячейки, в которой записано не больше одного символа некоторого заданного алфавита. Рабочий шаг машины заключается в том, что устройство стирает символ, записанный в обзреваемой ячейке, заменяет его на новый символ и перемещается на соседнюю слева или справа ячейку в новом состоянии. Шаг осуществляется в соответствии с *командой*. Набор команд задает *программу* данной МТ.

Чтобы задать программу МТ, необходимо задать:

1. Внешний алфавит  $A = \{a_0, a_1, \dots, a_n\}$ , символы которого записываются в ячейки ленты. Символ  $a_0$  называют «пустым»; он означает, что в ячейке ничего не записано и вводится для удобства рассуждений.

2. Алфавит внутренних состояний  $Q = \{q_0, q_1, \dots, q_m\}$ , в каждом из которых может находиться данная МТ. Среди всех внутренних состояний машины выделяют два:  $q_0$  - состояние остановки, попав в которое, машина прекращает работу, и  $q_1$  - начальное состояние, находясь в котором, машина начинает работу.

3. Набор команд, каждая из которых имеет вид:

$$q_i a_j \rightarrow q_k a_l K, \text{ где } \begin{cases} 1 \leq i \leq m, & 0 \leq j \leq n, \\ 0 \leq k \leq m, & 0 \leq l \leq n, \\ K \in \{Л, П, С\}. \end{cases}$$

Буквы Л, П, С, записанные в правой части команды означают, что после замены символа  $a_j$  на  $a_l$ , машина передвигается на одну ячейку влево (Л), вправо (П) или остается на месте (С), переходя из состояния  $q_i$  в состояние  $q_k$ .

Так как работа МТ полностью определяется ее состоянием в данный момент -  $q_i$  и содержимым обзреваемой в этот момент ячейки -  $a_j$ , то для каждой пары символов  $(q_i, a_j)$ , программа МТ должна содержать одну и только одну команду, начинающуюся этими символами. Очевидно, ни одна команда не может начинаться с символа  $q_0$ , поскольку он означает состояние остановки. Таким образом, программа МТ с внешним алфавитом  $A = \{a_0, a_1, \dots, a_n\}$  и алфавитом внутренних состояний  $Q = \{q_0, q_1, \dots, q_m\}$  должна содержать  $m \cdot (n + 1)$  команд. Программа МТ чаще всего записывается в виде таблицы, в строках которой записаны символы внешнего алфавита, в столбцах – символы алфавита внутренних состояний, а на пересечении строки с символом  $q_i$  и столбца с символом  $a_j$  записана правая часть команды  $q_k a_l K$ :

Q \ A	$a_0$	$a_1$	...	$a_j$	...	$a_n$
$q_1$						
$\vdots$						

$q_i$				$q_k a_l K$		
$\vdots$						
$q_m$						

Словом в алфавите  $A$  или  $Q$  или  $A \cup Q$  называется любая конечная последовательность букв соответствующего алфавита.

$k$ -той конфигурацией называется слово в алфавите  $A$ , записанное на ленте к началу  $k$ -го шага работы МТ, с указанием того, какая ячейка обозревается на этом шаге и в каком состоянии находится машина.

Конфигурация называется *заключительной*, если состояние, в котором находится МТ – заклочительное, т.е. состояние остановки  $q_0$ .

Непустое слово  $\alpha$  в алфавите  $A \setminus \{a_0\}$  *воспринимается МТ в стандартном положении*, если оно записано в последовательных ячейках ленты, все другие ячейки пусты, и МТ обозревает крайнюю справа ячейку из тех, в которых записано слово  $\alpha$ . Если при этом МТ находится в состоянии  $q_1$ , то стандартное положение называется *начальным*, если в состоянии  $q_0$  - то *заключительным*.

Слово  $\alpha$  *перерабатывается МТ в слово  $\beta$* , если от слова  $\alpha$ , воспринимаемого машиной в начальном состоянии, после выполнения конечного числа команд МТ приходит к слову  $\beta$ , воспринимаемому в положении остановки.

Задача переработки слов МТ с заданной программой называется *задачей применения МТ к словам*.

**ПРИМЕР.** Применить машину Тьюринга с данной программой:

$Q$	$A$	$a_0$	1	*
$q_1$		$q_1 a_0 П$	$q_3 a_0 Л$	$q_0 a_0$
$q_2$		$q_2 a_0 Л$	$q_4 a_0 П$	$q_4 a_0 П$
$q_3$		$q_2 a_0 П$	$q_3 1Л$	$q_3^* Л$
$q_4$		$q_1 a_0 Л$	$q_4 1П$	$q_4^* П$

к слову  $\alpha = 111^*1$ , воспринимаемому в начальном стандартном положении.

**РЕШЕНИЕ:**

Так как слово  $\alpha = 111^*1$  воспринимается МТ в начальном стандартном положении, то в начальный момент времени машина обозревает крайнюю правую ячейку, видит в ней символ 1, находясь в начальном состоянии  $q_1$ :

$$1 \ 1 \ 1^* \ 1$$

$\underbrace{\hspace{1.5cm}}_{q_1}$

Находим в программе МТ команду, которая начинается сочетанием символов  $q_1 1$ . Для этого выбираем в таблице строку с символом  $q_1$  и столбец с символом 1. На пересечении этих строки и столбца записана вторая часть команды:  $q_3 a_0 Л$ , выполняя которую машина стирает 1, записывает вместо нее пустой символ  $a_0$  и переходит в соседнюю слева ячейку в состоянии  $q_3$ :

$$1 \ 1 \ 1 \ \underbrace{*}_{q_3} a_0 .$$

В ячейке слева машина обозревает символ \* в состоянии  $q_3$ . На пересечении строки  $q_3$  и столбца \* записана команда  $q_3^*Л$ , по которой машина оставляет \* без изменения и в том же состоянии  $q_3$  передвигается еще на одну ячейку влево:

$$1 \ 1 \ \underbrace{1}_{q_3} * a_0 .$$

Теперь машина обозревает 1 в состоянии  $q_3$ . Следующий шаг она делает согласно команде:  $q_3 1 \rightarrow q_3 1Л$ :

$$\underbrace{1 \ 1 \ 1}_{q_3} * a_0 .$$

Очевидно, что эта команда будет повторяться до тех пор, пока машина будет «видеть» 1, находясь в состоянии  $q_3$ :

$$\underbrace{1 \ 1 \ 1}_{q_3} * a_0 \quad \text{и} \quad \underbrace{a_0 \ 1 \ 1 \ 1}_{q_3} * a_0 .$$

«Пересчитав» все единицы, считывающее устройство дойдет до пустой ячейки, в которой по договоренности записан пустой символ  $a_0$ . Обозревая  $a_0$  в состоянии  $q_3$ ,

машина должна выполнить следующую команду:  $q_3 a_0 \rightarrow q_2 a_0 П$ , согласно которой машина перейдет к ячейке с 1 в новом состоянии  $q_2$ :

$$a_0 \ \underbrace{1 \ 1 \ 1}_{q_2} * a_0 .$$

Далее по команде  $q_2 1 \rightarrow q_4 a_0 П$  машина стирает 1 и переходит уже на соседнюю справа ячейку в новом состоянии  $q_4$ :

$$a_0 \ a_0 \ \underbrace{1 \ 1}_{q_4} * a_0 .$$

По команде  $q_4 1 \rightarrow q_4 1П$  машина, оставив без изменений следующую справа 1 и оставаясь в состоянии  $q_4$ , дойдет в этом состоянии до ячейки с символом \*:

$$a_0 \ \underbrace{1 \ 1}_{q_4} * a_0 \quad \text{и} \quad a_0 \ 1 \ \underbrace{1}_{q_4} * a_0 .$$

Далее по команде  $q_4^* \rightarrow q_4^*П$ , машина, оставив \* без изменения, передвинется еще на одну ячейку вправо, где нет никакого символа, то есть, по договоренности, мы считаем, что там записан пустой символ  $a_0$ :

$$a_0 \ 1 \ 1 * \underbrace{a_0}_{q_4} a_0 .$$

По команде  $q_4 a_0 \rightarrow q_1 a_0 Л$ , «проверив», что единиц справа на ленте больше нет, машина, ничего не вписав в пустую ячейку, возвращается назад, влево, перейдя в состояние  $q_1$ :

$$a_0 \ 1 \ \underbrace{1}_{q_1} * a_0 .$$

Наконец, по команде  $q_1^* \rightarrow q_0 a_0$  машина «стирает» символ \* и останавливается.

Таким образом, слово  $\alpha = 111^*1$  перерабатывается данной МТ в слово  $\beta = 11$ .

Можно сказать, что данная МТ вычитает из числа единиц, записанных слева от \*, число единиц, записанных справа от нее.

### Конструирование машин Тьюринга

Задача написания программы машины Тьюринга, которая решала бы некоторую проблему, получила название *задачи конструирования машины Тьюринга*. Для конструирования машины Тьюринга необходимо задать внешний алфавит из  $n$  символов, алфавит внутренних состояний из  $m$  символов и программу, содержащую  $m(n + 1)$  команду. Формат допустимых команд описан в задании 1. Задача конструирования является более сложной, чем задача применения готовой машины Тьюринга к словам. Здесь не может существовать единого алгоритма, описывающего процесс конструирования программы. Выбор символов внешнего алфавита и состояний внутреннего алфавита обуславливается данными конкретной задачи, для решения которой строится программа.

**ПРИМЕР.** Сконструировать машину Тьюринга, которая вычисляла бы функцию  $f(x) = 2x$  для чисел в десятичной системе счисления.

### РЕШЕНИЕ:

В условии сказано, что числа представлены в десятичной системе счисления, поэтому, кроме обязательного пустого символа  $a_0$  во внешний алфавит необходимо включить цифры от 0 до 9:

$$A = \{a_0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \quad .$$

В алфавит внутренних состояний обязательно входят начальное состояние  $q_1$  и состояние остановки  $q_0$ . Иногда ими можно и ограничиться, а иногда необходимы новые состояния. В нашем случае, чтобы определить это, проведем ряд рассуждений. При удвоении натурального числа, записанного в десятичной системе счисления, реализуется один из двух случаев.

1. Если все цифры исходного числа не превосходят 4, то каждая из цифр заменяется цифрой, в два раза большей – получаем удвоенное число. Для осуществления этой операции достаточно состояния  $q_1$  и передвижения считывающего устройства влево на соседнюю ячейку.

2. Если же одна или несколько цифр превышают 4, то потребуется новое состояние  $q_2$ . Оно должно обеспечить не только замену последней цифры, но и изменение предыдущих цифр числа.

Получаем следующий набор команд:

1.  $q_1 0 \rightarrow q_1 0Л$ ;
2.  $q_1 1 \rightarrow q_1 2Л$ ;
3.  $q_1 2 \rightarrow q_1 4Л$ ;
4.  $q_1 3 \rightarrow q_1 6Л$ ;
5.  $q_1 4 \rightarrow q_1 8Л$ .

В первом случае, после удвоения каждой цифры числа, считывающее устройство доходит до пустой ячейки в состоянии  $q_1$ . Так как число удвоено, то машина должна остановиться. Получаем пятую команду:

$$6. q_1 a_0 \rightarrow q_0 a_0.$$

Во втором случае, когда последняя (или вообще, какая-либо из цифр) числа есть одна из цифр от 5 до 9, получаем следующие команды:

7.  $q_1 5 \rightarrow q_2 0Л$ ;
8.  $q_1 6 \rightarrow q_2 2Л$ ;
9.  $q_1 7 \rightarrow q_2 4Л$ ;

$$10. q_1 8 \rightarrow q_2 6Л;$$

$$11. q_1 9 \rightarrow q_2 8Л.$$

Состояние  $q_1$  в правой части этих команд использовать нельзя, так как в этом случае машина, перейдя влево, удвоит предыдущую цифру и мы получим неверный результат. Например, число 15 при использовании только состояния  $q_1$  перешло бы в число 20, а не в 30, как должно быть.

Теперь осталось задать команды для случая, когда машина в состоянии  $q_2$  обозревает ячейку с одной из цифр или с пустым символом. Получаем последние 11 команд, которые завершают программу.

$$12. q_2 0 \rightarrow q_1 1Л; \quad 17. q_2 5 \rightarrow q_2 1Л;$$

$$13. q_2 1 \rightarrow q_1 3Л; \quad 18. q_2 6 \rightarrow q_2 3Л;$$

$$14. q_2 2 \rightarrow q_1 5Л; \quad 19. q_2 7 \rightarrow q_2 5Л;$$

$$15. q_2 3 \rightarrow q_1 7Л; \quad 20. q_2 8 \rightarrow q_2 7Л;$$

$$16. q_2 4 \rightarrow q_1 9Л; \quad 21. q_2 9 \rightarrow q_2 9Л;$$

$$22. q_2 a_0 \rightarrow q_0 1.$$

Команды с 12 по 16 обеспечивают правильное изменение цифр числа от 0 до 4, если эти цифры не являются последними в десятичной записи числа; для цифр от 5 до 9 ту же задачу решают команды с 17 по 21. Последняя команда программы срабатывает в том случае, когда машина «добирается» до пустой ячейки в состоянии  $q_2$ .

Таким образом, алфавит внутренних состояний сконструированной машины содержит три состояния:

$$Q = \{q_0, q_1, q_2\}.$$

Проверим работу программы на числах **99108** и **1043**.

$$\begin{array}{ccccccc} 99108 & \xrightarrow{10} & 99106 & \xrightarrow{12} & 99116 & \xrightarrow{2} & 99216 & \xrightarrow{11} \\ \underbrace{\phantom{99108}}_{q_1} & & \underbrace{\phantom{99106}}_{q_2} & & \underbrace{\phantom{99116}}_{q_1} & & \underbrace{\phantom{99216}}_{q_1} & \\ \xrightarrow{11} & a_0 \underbrace{\phantom{98216}}_{q_2} & \xrightarrow{21} & a_0 \underbrace{\phantom{98216}}_{q_2} & \xrightarrow{22} & \underbrace{\phantom{198216}}_{q_0} & & \end{array}$$

Действительно, **99108•2 = 198216**.

$$\begin{array}{ccccccc} 1043 & \xrightarrow{4} & 1046 & \xrightarrow{5} & 1086 & \xrightarrow{1} & 1086 & \xrightarrow{2} \\ \underbrace{\phantom{1043}}_{q_1} & & \underbrace{\phantom{1046}}_{q_1} & & \underbrace{\phantom{1086}}_{q_1} & & \underbrace{\phantom{1086}}_{q_1} & \\ \xrightarrow{2} & a_0 \underbrace{\phantom{2086}}_{q_1} & \xrightarrow{21} & a_0 \underbrace{\phantom{2086}}_{q_0} & & & & \end{array}$$

Действительно, **1043•2 = 2086**.

### Машина произвольного доступа

Очередная попытка сформулировать понятие алгоритма в виде абстрактной математической машины была предпринята в 70-е годы 20 века. Эта машина получила название *машины произвольного доступа (МПД)*.

МПД состоит из бесконечного числа регистров  $R_1, R_2, \dots, R_n, \dots$ , в каждом из которых может быть записано натуральное число или 0. Будем обозначать число, записанное в регистре  $R_n$  через  $r_n$ .

Состоянием машины или конфигурацией будем называть последовательность чисел  $r_1, r_2, \dots, r_n, \dots$ . Работа МПД заключается в изменении ее конфигураций путем выполнения команд в порядке их написания. Машина имеет следующие типы команд.

1. **Команды обнуления.** Для всякого натурального числа  $n$  имеется команда  $Z(n)$ , действие которой заключается в замене содержимого регистра  $R_n$  на число 0. Содержимое других регистров при этом не меняется.

Обозначение:  $Z(n) - r_n := 0$ .

2. **Команды прибавления единицы.** Для всякого натурального числа  $n$  имеется команда  $S(n)$ , действие которой заключается в увеличении содержимого регистра  $R_n$  на 1. Содержимое других регистров при этом не меняется.

Обозначение:  $S(n) - r_n := r_n + 1$ .

3. **Команды переадресации.** Для всяких натуральных чисел  $n$  и  $m$  имеется команда  $T(m, n)$ , действие которой заключается в замене содержимого регистра  $R_n$  числом  $r_m$ , хранящимся в регистре  $R_m$ . Содержимое других регистров при этом не меняется.

Обозначение:  $T(m, n) - r_n := r_m$  или  $r_n \rightarrow R_m$ .

4. **Команды условного перехода.** Для всяких натуральных  $n, m$  и  $q$  имеется команда  $J(m, n, q)$ , действие которой заключается в следующем:

- сравнивается содержимое регистров  $R_n$  и  $R_m$ , затем:
- если  $r_n = r_m$ , то МПД переходит к выполнению команды с номером  $q$ ;
- если  $r_n \neq r_m$ , то МПД переходит к выполнению команды, следующей по списку.

Пусть  $P$  – программа МПД,  $K_0$  – начальная конфигурация. Применение программы  $P$  к начальной конфигурации  $P(K_0)$  называется *вычислением*. Будем обозначать команды программы  $P$  через  $I_1, I_2, \dots, I_S$ . В качестве начальных конфигураций будем рассматривать только такие, в которых имеется лишь *конечное число* ненулевых элементов и вместо  $(a_1, a_2, \dots, a_n, 0, \dots)$  будем писать  $(a_1, a_2, \dots, a_n)$ .

Опишем класс функций, вычисляемых на МПД. Множество натуральных чисел с нулем обозначим  $N_0$ . Будем рассматривать частичные функции

$$f : N_0 \times N_0 \times \dots \times N_0 \rightarrow N_0$$

Будем говорить, что программа  $P$  вычисляет функцию  $f$  с результатом  $b$  ( $b \in N$ ), если она применима к начальной конфигурации  $(a_1, a_2, \dots, a_n)$ ,  $f$  определена на этой конфигурации и  $f(a_1, a_2, \dots, a_n) = b$ .

#### **ПРИМЕР.**

Записать программу и построить блок-схему МПД, которая вычисляла бы функцию  $f(x, y) = x + y$ .

#### **РЕШЕНИЕ:**

В качестве начальной конфигурации возьмем  $(x, y, 0)$ . Тогда содержимое первого регистра до начала работы программы будет равно  $x$ , второго -  $y$ , а третьего - 0:

$$R_1 : r_1 := x, R_2 : r_2 := y, R_3 : r_3 := 0.$$

Функция  $f(x, y) = x + y$  может быть вычислена следующей программой, состоящей из пяти команд:

$I_1 = J(3, 2, 5)$  – команда условного перехода, которая сравнивает содержимое третьего и второго регистров и, если числа, записанные в них равны, то осуществляется переход к пятой команде  $I_5$ , а если не равны, то к следующей по списку команде  $I_2$ .

$I_2 = S(1) - r_1 := r_1 + 1$  – команда прибавления единицы к содержимому первого регистра.

$I_3 = S(3) - r_3 := r_3 + 1$  - прибавления единицы к содержимому третьего регистра.

$I_4 = J(1, 1, 1)$  – команда, обеспечивающая цикл. Так как содержимое первого регистра всегда равно самому себе, то эта команда возвращает машину к выполнению первой команды до тех пор, пока число в третьем регистре не станет равно числу во втором регистре. Так прибавление по единице идет последовательно к содержимому третьего, а затем первого регистров, то когда в третьем регистре получим число  $y$ , то в первом, соответственно, получим  $x + y = f(x, y)$ .

$I_5$  - **останов.** -  $r_1 := x + y = f(x, y)$ .

### Нумерация машин Тьюринга

Под нумерацией алгоритмов понимают эффективные кодирования натуральными числами множества всех алгоритмов для каждой из рассматриваемых моделей алгоритмов. Данные результаты относятся к числу фундаментальных, так как они используются, в частности, для установления невычислимости ряда конкретных функций.

Будем считать, что символы внешних алфавитов и алфавитов внутренних состояний машин Тьюринга берутся из двух множеств символов:

$$\{a_0, a_1, \dots, a_i, \dots\} \text{ и } \{q_0, q_1, \dots, q_j, \dots\}.$$

При этом будем считать, что  $a_0$  принадлежит всем внешним алфавитам машин Тьюринга и означает пустой символ, а символы  $q_0$  и  $q_1$  принадлежат всем алфавитам внутренних состояний и означают начальное состояние и состояние остановки соответственно. Назовем следующий набор символов *стандартным алфавитом*:

$$A = \{L, P, C, a_0, a_1, \dots, a_i, \dots, q_0, q_1, \dots, q_j, \dots\}.$$

Каждому символу стандартного алфавита поставим в соответствие двоичный набор – код – согласно таблице 1.

**Таблица 1**

	Символ	Код	Число нулей в коде
Символы сдвига	П	10	1
	Л	100	2
	С	1000	3
Символы внешнего алфавита	$a_0$	10000	4
	$a_1$	1000000	6
	...	...	...
	$a_i$	100...00	$2i + 4$
	...	...	...
Символы алфавита состояний	$q_0$	100000	5
	$q_1$	10000000	7
	...	...	...
	$q_j$	100...000	$2i + 5$
	...	...	...

Если команда машины Тьюринга  $I$  имеет формат:

$$I =: qa \rightarrow q'a'X, \quad X \in \{L, P, C\},$$

то ей ставится двоичный набор по правилу:

$$\text{Код}(I) = \text{Код}(q)\text{Код}(a)\text{Код}(q')\text{Код}(a')\text{Код}(X) \quad (1)$$

Пусть  $\Theta$  - произвольная машина Тьюринга. Упорядочим все ее команды в соответствии с лексикографическим порядком левых частей команд:

$$q_1a_0, q_1a_1, \dots, q_1a_n, q_2a_0, q_2a_1, \dots, q_2a_n, \dots, q_ma_0, q_ma_1, \dots, q_ma_n.$$

Всего программа машины Тьюринга с внешним алфавитом из  $n$  символов и алфавитом внутренних состояний из  $m$  символов содержит  $m(n+1)$  команд, которые по указанному выше правилу оказались лексикографически упорядочены следующим образом:  $I_1, I_2, \dots, I_{m(n+1)}$ . Данной последовательности команд поставим в соответствие двоичный набор вида:

$$\text{Код}(\Theta) = \text{Код}(I_1) \text{Код}(I_2) \dots \text{Код}(I_{m(n+1)}) \quad (2).$$

Из указанной процедуры следует, что машина  $\Theta$  переводит конфигурацию вида  $q_1a_1^x$  в конфигурацию вида  $q_0a_1^x$ , поэтому, представляя натуральное число  $n$  в виде  $a_1^{n+1}$ , получаем, что  $\Theta$  вычисляет функцию

$$f(x) = x.$$

Очевидно, что такое кодирование является алгоритмической процедурой. Зная код машины, можно однозначно восстановить ее программу. Для этого нужно выделить все подслова, начинающиеся единицей с каким-либо числом нулей, до следующей единицы. Пятерка таких подслов образует команду.

*Номером* машины Тьюринга называется натуральное число, которое получается при переводе в десятичную систему счисления двоичный код этой машины. Очевидно, что так как все коды начинаются с единицы, то разным кодам соответствуют разные натуральные числа. Тогда все мыслимые машины Тьюринга можно упорядочить по возрастанию их номеров:

$$T_0, T_1, \dots, T_n, \dots \quad (3).$$

Указанное упорядочение является эффективным в том смысле, что существует алгоритм, который по номеру  $n$  машины  $T_n$  выдает ее код, и, обратно, существует алгоритм, который по заданному коду выдает номер машины.

### Нумерация МПД – программ

Определим нумерацию команд, которые может выполнять машина произвольного доступа. Для этого зададим некоторую функцию  $\alpha$ , вычисляющую номера команд МПД.

$$\begin{aligned} \alpha(Z(n)) &= 4(n-1); \\ \alpha(S(n)) &= 4(n-1)-1; \\ \alpha(T(m, n)) &= 4p(m-1, n-1)+2; \\ \alpha(J(m, n, q)) &= 4\pi(m, n, q)+3, \end{aligned} \quad (4)$$

где  $p(x, y) = 2^x(2y+1)-1$  и  $\pi(x, y, z) = p(p(x-1, y-1), z-1)$ .

Можно показать, что функция  $\alpha$  и обратная к ней эффективно вычислимы. Определим теперь номер программы произвольной МПД.

Пусть программа имеет вид  $P = I_1 \dots I_s$ . Номер программы  $\gamma(P)$  вычисляется следующим образом:

$$\gamma(P) = \tau(\alpha(I_1), \dots, \alpha(I_s)) \quad (5),$$

где  $\tau(x_1, \dots, x_s) = 2^{x_1} + 2^{x_1+x_2+1} + 2^{x_1+x_2+x_3+2} + \dots + 2^{x_1+\dots+x_s+s-1} - 1$  (6).

Присвоим каждой программе МПД ее номер  $\gamma(P)$  и упорядочим номера по возрастанию. Указанное упорядочение также является эффективным, так как по каждой программе позволяет найти ее номер и обратно.

**ПРИМЕР.**

1. Занумеровать машину Тьюринга с программой  $\Theta$ :

$$\begin{aligned} \Theta: \quad & q_1 a_0 \rightarrow q_2 a_1 L \\ & q_1 a_1 \rightarrow q_2 a_0 C \\ & q_2 a_0 \rightarrow q_1 a_1 \Pi \\ & q_2 a_1 \rightarrow q_0 a_1 C. \end{aligned}$$

2. Занумеровать программу МПД:

$$\begin{aligned} P: \quad & I_1 = S(2) \\ & I_2 = Z(3) \\ & I_3 = T(1, 3) \\ & I_4 = J(1, 1, 1). \end{aligned}$$

**РЕШЕНИЕ:**

1. В программе  $\Theta$  4 команды. Сначала найдем коды этих команд согласно таблице 1 правилу (1).

Первая команда  $q_1 a_0 \rightarrow q_2 a_1 L$  имеет код  $10^7 10^4 10^9 10^6 10^2$ ; вторая команда  $q_1 a_1 \rightarrow q_2 a_0 C$  - код  $10^7 10^6 10^9 10^4 10^3$ ; третья команда  $q_2 a_0 \rightarrow q_1 a_1 \Pi$  - код  $10^9 10^4 10^7 10^6 10^1$ , и, наконец, последняя команда -  $q_2 a_1 \rightarrow q_0 a_1 C$  - код  $10^9 10^6 10^5 10^6 10^3$ . Тогда код машины будет иметь вид:

$$10^7 10^4 10^9 10^6 10^2 10^7 10^6 10^9 10^4 10^3 10^9 10^4 10^7 10^6 10^1 10^9 10^6 10^5 10^6 10^3.$$

2. Согласно формулам (4), рассчитаем значение  $\alpha$ -функции от каждой команды программы P.

$$\alpha(I_1) = \alpha(S(2)) = 4(2 - 1) - 1 = 4 - 1 = \underline{3};$$

$$\alpha(I_2) = \alpha(Z(3)) = 4(3 - 1) = 4 \cdot 2 = \underline{8};$$

$$\alpha(I_3) = \alpha(T(1, 3)) = 4p(1 - 1, 3 - 1) + 2 = 4p(0, 2) + 2,$$

так как  $p(0, 2) = 2^0(2 \cdot 2 + 1) - 1 = 4$ , то  $\alpha(I_3) = \alpha(T(1, 3)) = 4 \cdot 4 + 2 = \underline{18}$ .

$$\alpha(I_4) = \alpha(J(1, 1, 1)) = 4\pi(1, 1, 1) + 3.$$

Т. к.  $\pi(1, 1, 1) = p(p(1 - 1, 1 - 1), 1 - 1) = p(p(0, 0), 0) = p([2^0(2 \cdot 0 + 1) - 1], 0) = p(0, 0) = 2^0(2 \cdot 0 + 1) - 1 = 0$ , то:

$$\alpha(I_4) = \alpha(J(1, 1, 1)) = 4 \cdot 0 + 3 = \underline{3}.$$

Согласно формуле (5), номер программы P равен:

$$\gamma(P) = \tau(\alpha(I_1), \alpha(I_2), \alpha(I_3), \alpha(I_4)) = \tau(3, 8, 18, 3).$$

Значение функции  $\tau$  рассчитаем по формуле (6):

$$\tau(3, 8, 18, 3) = 2^3 + 2^{3+8+1} + 2^{3+8+18+2} + 2^{3+8+18+3+3} - 1 = 2^3 + 2^{12} + 2^{31} + 2^{35} - 1.$$

Следовательно,  $\gamma(P) = 2^3 + 2^{12} + 2^{31} + 2^{35} - 1 = 36507226119$ .

## Частично-рекурсивные функции

Класс частично рекурсивных функций также был введен в качестве еще одного уточнения понятия алгоритма. Данный класс определяется путем указания конкретных исходных функций, называемых *базисными* и фиксированного множества операций получения новых функций из заданных.

Обозначим через  $\mathbf{N}_0$  множество натуральных чисел с нулем. Тогда в качестве базисных функций берутся следующие функции.

1). Нуль-функция  $0(x)$ :

$$(\forall x \in \mathbf{N}_0) \quad 0(x) = 0 \quad (1).$$

2). Функция следования  $s(x)$ :

$$(\forall x \in \mathbf{N}_0) \quad s(x) = x + 1 \quad (2).$$

3). Функция выбора аргумента  $I_m^n(x_1, x_2, \dots, x_n)$ :

$$(\forall x \in \mathbf{N}_0) \quad I_m^n(x_1, x_2, \dots, x_n) = x_m, \quad m \in \{1, 2, \dots, n\} \quad (3).$$

Допустимыми операциями над функциями являются операции *суперпозиции* или *подстановки*, *примитивной рекурсии* и *минимизации*.

Операция суперпозиции.

Пусть заданы одна  $n$ -местная функция  $g(x_1, x_2, \dots, x_n)$  и  $n$   $m$ -местных функций, зависящих от одних и тех же переменных  $f_1(x_1, x_2, \dots, x_m)$ ,  $f_2(x_1, x_2, \dots, x_m)$ , ...,  $f_n(x_1, x_2, \dots, x_m)$ . (Этого можно добиться введением фиктивных переменных).

*Суперпозицией* функций  $g$  и  $f_1, f_2, \dots, f_n$  называется функция

$$h(x_1, x_2, \dots, x_m) = g(f_1(x_1, x_2, \dots, x_m), f_2(x_1, x_2, \dots, x_m)) \quad (4),$$

Очевидно, что функция  $h$  определена тогда и только тогда, когда определены все функции  $f_1, f_2, \dots, f_n$ . Операцию суперпозиции обозначают:

$$h = S(g, f_1, f_2, \dots, f_n) \quad (5).$$

Операция примитивной рекурсии.

Пусть заданы  $n$ -местная функция  $g(x_1, x_2, \dots, x_n)$  и  $(n + 2)$ -местная функция  $h(x_1, x_2, \dots, x_n, y, z)$ . Определим  $(n + 1)$ -местную функцию  $f$  индуктивно с помощью соотношений:

$$\begin{cases} f(x_1, x_2, \dots, x_n, 0) = g(x_1, x_2, \dots, x_n); \\ f(x_1, x_2, \dots, x_n, y+1) = h(x_1, x_2, \dots, x_n, y, f(x_1, x_2, \dots, x_n, y)) \end{cases} \quad (6).$$

Про функцию  $f$  говорят, что она получена рекурсией из функций  $g$  и  $h$  и обозначают:

$$f = R(g, h).$$

Операция минимизации.

Пусть задана  $n$ -местная функция  $g(x_1, x_2, \dots, x_{n-1}, y)$ . Зафиксируем набор  $x_1, x_2, \dots, x_{n-1}, x_n$  и рассмотрим уравнение относительно  $y$ :

$$g(x_1, x_2, \dots, x_{n-1}, y) = x_n \quad (7).$$

Будем решать данное уравнение, последовательно вычисляя

$$g(x_1, x_2, \dots, x_{n-1}, 0), g(x_1, x_2, \dots, x_{n-1}, 1), g(x_1, x_2, \dots, x_{n-1}, 2) \text{ и т.д.}$$

и сравнивая с  $x_n$ . Наименьшее  $y$ , для которого выполнено (7), обозначим

$$\mu_y = (g(x_1, x_2, \dots, x_{n-1}, y) = x_n) \quad (8).$$

Про функцию  $f$  говорят, что она получена минимизацией из функции  $g$  и обозначают:

$$f = \mu_y(g).$$

Функция называется *частично рекурсивной*, если она может быть получена из базисных функций применением конечного числа раз операций суперпозиции, примитивной рекурсии и минимизации.

**ПРИМЕР.**

Выразить функцию  $f(x, y) = x + y$  через базисные: нуль-функцию -  $0(x) = 0$ , функцию следования -  $s(x) = x + 1$  и функцию выбора аргументов -  $I_m^n(x_1, x_2, \dots, x_n) = x_m$ .

**РЕШЕНИЕ:**

Для любых чисел  $x$  и  $y$  выполняются соотношения:

$$x + 0 = x = I_1^2(x, y),$$

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

Пусть  $I_1^2(x, y) = g(x)$ ,  $s(z) = z + 1 = (x + y) + 1 = h(x, y, z)$ , т.е. заданы одноместная функция  $g(x)$  и трехместная функция  $h(x, y, z)$ .

Тогда функцию  $f(x, y) = x + y$  можно рассматривать как двуместную функцию, удовлетворяющую соотношениям:

$$f(x, 0) = x + 0 = I_1^2(x, y) = g(x),$$

$$f(x, y + 1) = s(z) = h(x, y, f(x, 1)).$$

Следовательно, функция  $f(x, y) = x + y$  получена с помощью операции примитивной рекурсии из функций  $g(x)$  и  $h(x, y, z)$ , или, с учетом вышесказанного, из функций  $I_1^2(x, y)$  - функция выбора аргумента и  $s(z)$  - функция следования.

**Тематика рефератов**

1. Алгоритмы вокруг нас.
2. Основатели теории алгоритмов – Клини, Черч, Пост, Тьюринг.
3. Тезис Черча.
4. Проблема вычислимости математической логике.
5. Нормальные алгоритмы Маркова..
6. Методы разработки алгоритмов.
7. Средства и языки описания (представления) алгоритмов.
8. История формирования «понятия алгоритмов».
9. Известнейшие алгоритмы в истории математики.
10. Нормальные алгоритмы Маркова.
11. Принцип нормализации Маркова.
12. Эквивалентность различных теорий алгоритмов.
13. Алгоритмические проблемы.
14. Теорема Гёделя о неполноте формальной арифметики.