Волжский Университет им. Татищева.

Лекции
по математической логике и
теории алгоритмов.

Составитель: доцент С.В. Каверин.

Тольятти 2002.

Содержание

Глава I. Алгебра логики.

§1.1. Определение булевой функции.

Булевой функцией $y=f(x_1,x_2,\ldots,x_n)$ от *n* переменных x_1, $x_2,\ldots,x_n$ называется любая функция, в которой аргументы и функция могут принимать значение либо 0 либо 1, т.е. булева функция это правило по которому произвольному набору нулей и единиц $(x_1,x_2,\ldots,x_n)$ ставится в соответствие значение 0 или 1.

Булевы функции называются также **функциями алгебры логики, двоичными функциями и переключательными функциями.**

Булеву функцию от ***n*** переменных можно задать таблицей истинности, в которой наборы значений аргументов расположены в порядке возрастания их номеров**:** сначала идет набор, представляющий собой двоичное разложение 0 (этот набор имеет номер 0); затем идет набор, являющийся двоичным разложением 1, потом 2, 3 и т.д. Последний набор состоит из ***n*** единиц и является двоичным разложением числа 2^n-1 (такой порядок расположения наборов назовем **лексикографическим порядком**). Учитывая, что отсчет начинается с 0, а значение булевой функции может быть либо 0 либо 1, заключаем, что существует всего 2^{2^n} различных булевых функций от ***n*** переменных. Таким образом, имеется, например, 16 булевых функций от двух переменных, 256 — от трех и т. д.

Пример 1.1.1. (голосование). Рассмотрим устройство, фиксирующее принятие некоторой резолюции "комитетом трех". Каждый член комитета при одобрении резолюции нажимает свою кнопку. Если большинство членов голосуют «за», то резолюция принимается. Это фиксируется регистрирующим прибором. Таким образом, устройство реализует функцию f(x,y,z*)*, таблица истинности которой имеет вид

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

Булева функция также однозначно задается перечислением всех наборов, на которых она принимает значение 0, либо перечислением всех наборов, на которых она принимает значение 1.Полученную в примере функцию *f* можно также задать следующей системой равенств: f(0,0,0) = f(0,0,1) = f(0,1,0) = f(1,0,0) =0.

Вектором значений булевой функции $y=f(x_1,x_2,\ldots,x_n)$ называется упорядоченный набор всех значений функции f, при котором значения упорядочены по лексикографическому порядку. Например, пусть функция трех переменных f задана вектором значений (0000 0010) и необходимо найти набор, на котором f принимает значение 1. Т.к. 1 стоит на 7 месте, а нумерация в лексикографическом порядке начинается с 0, то необходимо найти двоичное разложение 6. Таким образом, функция f принимает значение 1 на наборе (110).

§1.2. Элементарные булевы функции.

Среди булевых функций особо выделяются так называемые элементарные булевы функции, посредством которых можно описать любую булеву функцию от любого числа переменных.

1. Булева функция $f(x_1,x_2,\ldots,x_n)$ принимающая значение 1 на всех наборах нулей и единиц называется **константой 1**, или тождественной единицей. Обозначение**: 1**.

2. Булева функция $f(x_1,x_2,\ldots,x_n)$ принимающая значение 0 на всех наборах нулей и единиц называется **константой 0**, или тождественным нулем. Обозначение**: 0**.

3. Отрицанием называется булева функция одной переменной, которая определяется следующей таблицей истинности

x	0	1
f(x)	1	0

Обозначения**:** $\neg x$, $\overline{x}$. Запись $\neg x$ читается «не икс» или «отрицание икс».

4. Конъюнкцией называется булева функция двух переменных, которая определяется следующей таблицей истинности

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

Другие названия**:** логическое умножение (произведение); логическое «и».

Обозначения**:** x&y, x·y, x∧y, min(x,y).

Запись x&y может читаться так**:** «икс и игрек» или «икс умножить на игрек».

5. Дизъюнкцией называется булева функция двух переменных, которая определяется следующей таблицей истинности

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

Другие названия: логическое сложение (сумма); логическое «или».

Обозначения: $x \vee y$, $\max(x,y)$.

Запись $x \vee y$ может читаться так: «икс или игрек» или «сумма икс и игрек».

6. Импликацией называется булева функция двух переменных, которая определяется следующей таблицей истинности

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

Другое название: логическое следование.

Обозначения: $x \to y$, $x \Rightarrow y$, $x \supset y$.

Запись $x \to y$ может читаться так: «из икс следует игрек».

7. Эквивалентностью называется булева функция двух переменных, которая определяется следующей таблицей истинности

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

Обозначения: $x \sim y$, $x \leftrightarrow y$,.$x \equiv y$.

Запись $x \sim y$ может читаться так: «икс эквивалентно игрек» или «икс равносильно игрек».

8. Суммой по модулю_2 называется булева функция двух переменных, которая определяется следующей таблицей истинности

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

Другое название: антиэквивалентность.

Обозначения: $x \oplus y$, $x+y$.

Запись $x \oplus y$ может читаться так: «икс плюс игрек».

9. Штрих Шеффера это булева функция двух переменных, которая определяется следующей таблицей истинности

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

Другое название: отрицание конъюнкции, логическое «не-и».

Обозначение: $x|y$.

Запись $x|y$ может читаться так: «не икс или не игрек», «икс и игрек несовместны», «икс штрих Шеффера игрек».

10. Стрелка Пирса это булева функция двух переменных, которая определяется следующей таблицей истинности

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

Другое название**:** отрицание дизъюнкции, логическое «не-или», функция Вебба.

Обозначение**:** $x \downarrow y$**;** для функции Вебба - $x \circ y$.

Запись $x \downarrow y$ может читаться так**:** «ни икс и ни игрек», «икс стрелка Пирса игрек».

Замечание. Символы $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\sim$, $\oplus$, $|$, $\downarrow$ участвующие в обозначениях элементарных функций будем называть связками или операциями.

§1.3. Задание булевых функций посредством элементарных.

Если в логическую функцию вместо переменных подставить некоторые булевые функции, то в результате получается новая булевая функция, которая называется **суперпозицией** подставляемых функций (сложная функция). С помощью суперпозиции можно получать более сложные функции, которые могут зависить от любого числа переменных. Запись булевых функций через элементарные булевые функции будем называть **формулой** реализующей данную функцию.

Пример 1.3.1. Пусть задана элементарная булева функция $x \vee y$. Подставим вместо x функцию $x_1 \downarrow x_2$. Получаем функцию трех переменных $(x_1 \downarrow x_2) \vee y$. Если вместо переменной y подставить, например, $x_3 \oplus x_4$, то получаем новую функцию от четырех переменных: $(x_1 \downarrow x_2) \vee (x_3 \oplus x_4)$. Очевидно, что таким образом мы будем получать булевы функции, которые будут выражаться через элементарные булевы функции.

Забегая вперед, отметим, что **любая** булева функция может быть представлена как суперпозиция элементарных функций.

Для более компактной записи сложных функций введем следующие соглашения**:** 1) внешние скобки опускаются; 2) сначала производятся операции в скобках; 3) считается, что приоритет связок убывает в следующем порядке**:** $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\sim$. Для равносильных связок и оставшихся связок $\{\oplus, |, \downarrow\}$, следует пока расставлять скобки.

Примеры 1.3.2. В формуле $x \wedge y \vee z$ скобки расставляются следующим образом: $((x \wedge y) \vee z)$, т.к. операция $\wedge$ сильнее операции $\vee$ согласно нашему соглашению.

1. В формуле $x \vee y \sim z \rightarrow x$ скобки расставляются следующим образом: $((x \vee y) \sim (z \rightarrow x))$
2. В формуле $(x \oplus y) \sim z \rightarrow xy \vee \neg z$ скобки расставляются следующим образом: $((x \oplus y) \sim (z \rightarrow ((xy) \vee (\neg z))))$.

3. Формула $x \to y \to z$, следуя нашему соглашению, записана не корректно, т.к. расстановка скобок приводит к двум разным функциям: $((x \to y) \to z)$ и $(x \to (y \to z))$.

§1.4. Существенные и несущественные переменные.

Булева функция $y=f(x_1,x_2,\ldots,x_n)$ **существенно зависит** от переменной x_k, если существует такой набор значений $a_1,a_2,\ldots,a_{k-1}, a_{k+1},a_{k+2},\ldots,a_n$, что

$$f(a_1,a_2,\ldots,a_{k-1},\mathbf{0},a_{k+1},a_{k+2},\ldots,a_n) \neq f(a_1,a_2,\ldots,a_{k-1},\mathbf{1},a_{k+1},a_{k+2},\ldots,a_n).$$

В этом случае x_k называют существенной переменной, в противном случае x_k называют несущественной (фиктивной) переменной. Другими словами, переменная является несущественной, если ее изменение не изменяет значения функции.

Пример 1.4.1. Пусть булевы функции $f_1(x_1,x_2)$ и $f_2(x_1,x_2)$ заданы следующей таблицей истинности:

x_1	x_2	$f_1(x_1,x_2)$	$f_2(x_1,x_2)$
0	0	0	1
0	1	0	1
1	0	1	0
1	1	1	0

Для этих функций переменная x_1 — существенная, а переменная x_2—несущественная.

Очевидно, что булевы функции не изменяют свои значения путем введения (или удаления) несущественных переменных. Поэтому, в дальнейшем булевы функции рассматриваются с точностью до несущественных переменных (в примере можно записать: $f_1(x_1,x_2)=x_1$, $f_2(x_1,x_2)=\neg x_1$).

§1.5. Таблицы истинности. Эквивалентные функции.

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

Пример 1.5.1. $F1=x_1 \wedge x_2 \vee (x_1 \wedge \neg x_2 \vee \neg x_1 \wedge x_2)$

x_1	x_2	$x_1 \wedge \neg x_2$	$\neg x_1 \wedge x_2$	$x_1 \wedge \neg x_2 \vee \neg x_1 \wedge x_2$	$x_1 \wedge x_2$	.F1
0	0	0	0	0	0	0
0	1	0	1	1	0	1
1	0	1	0	1	0	1
1	1	0	0	0	1	1

Таким образом, формула F1 реализует дизъюнкцию.

Пример 1.5.2. $F2=x_1 \wedge x_2 \rightarrow x_1$

x_1	x_2	$x_1 \wedge x_2$	$F2=(x_1 \wedge x_2) \rightarrow$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	1

Таким образом, формула F2 реализует константу 1.

Пример 1.5.3. $F3=((x_1 \wedge x_2) \oplus x_1) \oplus x_2$.

x_1	x_2	$x_1 \wedge x_2$	$(x_1 \wedge x_2) \oplus x_1$	$(x_1 \wedge x_2) \oplus x_1) \oplus x_2$
0	0	0	0	0
0	1	0	0	1
1	0	0	1	1
1	1	1	0	1

Таким образом, формула F3 реализует дизъюнкцию.

Булевы функции f1 и f2 называются **эквивалентными**, если на всяком наборе $(a_1, a_2, \ldots, a_n)$ нулей и единиц значения функций совпадают. Обозначение эквивалентных функций следующее**:** f1=f2.

Согласно приведенным примерам 1-3, можно написать

- $x_1 \wedge x_2 \vee (x_1 \wedge \neg x_2 \vee \neg x_1 \wedge x_2) = x_1 \vee x_2$;
- $x_1 \wedge x_2 \rightarrow x_1 = 1$;
- $((x_1 \wedge x_2) \oplus x_1) \oplus x_2 = x_1 \vee x_2$.

§1.6. Основные эквивалентности.

Приводимые эквивалентности часто оказываются полезными при оперировании с булевыми функциями.

Ниже H, H1, H2,… означают некоторые булевы функции.

1. Закон двойного отрицания : $H = \overline{\overline{H}}$.
2. Идемпотентность
   $H \& H = H$, $H \vee H = H$.

3. Коммутативность:
   $H1*H2=H2*H1$, где символ $*$ означает одну из связок $\&$, $\vee$, $\oplus$, $\sim$, $|$, $\downarrow$.

4. Ассоциативность:
   $H1*(H2*H3)=(H1*H2)*H3$, где символ $*$ означает одну из связок $\&$, $\vee$, $\oplus$, $\sim$.

5. Дистрибутивность:
   $H1\&(H2\vee H3)=(H1\&H2)\vee(H1\&H3)$;
   $H1\vee(H2\&H3)=(H1\vee H2)\&(H1\vee H3)$;
   $H1\&(H2\oplus H3)=(H1\&H2)\oplus(H1\&H3)$.

6. Законы де Моргана:
   $\overline{H1\,\&\,H2}=\overline{H1}\vee\overline{H2}$, $\overline{H1\vee H2}=\overline{H1}\,\&\,\overline{H2}$.

7. Правила поглощения:
   $H1\vee(H2\&H3)=H1$, $H1\&(H2\vee H3)=H1$

8. Законы склеивания:
   $H1\,\&\,H2\vee H1\,\&\,\overline{H2}=H1$, $(H1\vee H2)\,\&\,(H1\vee\overline{H2})=H1$.

9. Законы инверсий: $H\vee\overline{H}=1$, $H\,\&\,\overline{H}=0$.

10. Правила операций с константами:
    $H\vee1=1$, $H\&1=H$, $H\vee0=H$, $H\&0=0$.

Для проверки истинности приведенных эквивалентностей достаточно построить соответствующие таблицы истинности.

Отметим, что свойство ассоциативности операции позволяет распространить эту операцию для любого количества переменных. Так, например, запись $x\vee y\vee z\vee w$ корректна, т.к. любая расстановка скобок приводит к одной и той же функции. Коммутативность операции позволяет менять местами переменные в формуле. Например, $x\wedge y\wedge z\wedge w=w\wedge y\wedge x\wedge z$.

§1.7. Функциональная полнота.

В типичной современной цифровой вычислительной машине цифрами являются 0 и 1. Следовательно, команды, которые выполняет процессор, суть булевы функции. Ниже будет показано, что любая булева функция реализуется через конъюнкцию, дизъюнкцию и отрицание. Следовательно, можно построить нужный процессор, имея в распоряжении элементы, реализующие конъюнкцию, дизъюнкцию и отрицание. Этот раздел посвящен ответу на вопрос: существуют ли (и если существуют, то какие) другие системы булевых функций, обладающих тем свойством, что с их помощью можно выразить все другие функции.

Введем в рассмотрение ряд классов функций.

1. Класс функций, сохраняющих константу 0, т.е. таких функций $y=f(x_1,x_2,\dots,x_n)$,что $f(0,0,\dots,0)=0$. Например, $x\vee y(\neg x)$.
2. Класс функций, сохраняющих константу 1, т.е. таких функций $y=f(x_1,x_2,\dots,x_n)$,что $f(1,1,\dots,1)=1$. Например, $x\vee\neg(yx)$.
3. Класс самодвойственных функций, т.е. таких функций $y=f(x_1,x_2,\dots,x_n)$, что $f(x_1,x_2,\dots,x_n)=\overline{f(\overline{x}_1,\overline{x}_2,\dots,\overline{x}_n)}$.
4. Класс линейных функций, т.е. таких функций $y=f(x_1,x_2,\dots,x_n)$, которые могут быть представлены в виде $f(x_1,x_2,\dots,x_n)=c_0\oplus c_1x_1\oplus c_2x_2\oplus\dots\oplus c_nx_n$, где c_0, c_1, c_2…коэффициенты, которые могут принимать значение 0 или 1.
5. Класс монотонных функций. На множестве наборов из нулей и единиц $B^n=\{(x_1,x_2,\dots,x_n):x_i\in\{0,1\},i=1,2,\dots,n\}$ введем частичный порядок следующим образом: $(a_1,,a_2,\dots,a_n)\le(b_1,b_2,\dots,b_n)$ тогда и только тогда когда $a_1\le b_1$, $a_2\le b_2,\dots,a_n\le b_n$. Функция $f(x_1, x_2,\dots,x_n)$ называется монотонной, если для любых двух элементов из B^n таких, что $(a_1,,a_2,\dots,a_n)\le(b_1,b_2,\dots,b_n)$ следует, что $f(a_1,,a_2,\dots,a_n)\le f(b_1,b_2,\dots,b_n)$.

Система S булевых функций, суперпозицией которых может быть представлена любая булева функция, называется **функционально полной**. Говорят, что функционально полная система S булевых функций образует **базис** в логическом пространстве. Базис S называется **минимальным**, если удаление из него любой функции превращает эту систему в неполную.

Критерий полноты (Теорема Поста). Система S булевых функций является полной тогда и только тогда, когда она включает хотя бы одну функцию: не сохраняющую константу 0, не сохраняющую константу 1, не самодвойственную, нелинейную и немонотонную.

В таблице 1.7 приведены свойства, которыми обладают элементарные булевы функции (символ * - отмечает свойство, которым обладает данная функция).

Используя теорему Поста и таблицу 1.7 можно строить базисы из элементарных функций по следующему правилу. Выбрав любую элементарную булеву функцию и дополнив ее при необходимости другими функциями так, чтобы все они вместе удовлетворяли теореме о функциональной полноте. Через функции этого базиса можно выразить **все** другие булевы функции.

Построим некоторые часто используемые в приложениях базисы.

Таблица 1.7

Название	Обозн.	Не сохранимость константы 0	Не сохранимость константы 1	Не Самодвойственность	Не Линейность	Не Монотон-ность
Конст.0	0		*	*		
Конст.1	1	*		*		
Отриц.	¬	*	*			*
Конъюн.	&			*	*	
Дизъюн.	∨			*	*	
Имплик.	→	*		*	*	*
Эквивал.	~	*		*		*
Сумма по мод_2	⊕		*	*		*
Штрих Шеффера	|	*	*	*	*	*
Стрелка Пирса	↓	*	*	*	*	*

1. Система функций S1={¬,∧,∨} образует базис. Для приведения булевой функции к виду содержащему лишь связки из базиса S1 могут быть полезны следующие эквивалентности:
$x \to y = \overline{x} \vee y$, $x \leftrightarrow y = (x \vee \overline{y})(\overline{x} \vee y)$,
$x \oplus y = \overline{x}y \vee x\overline{y}$, $x \mid y = \overline{x} \vee \overline{y}$,
$x \downarrow y = \overline{x} \,\&\, \overline{y}$.

2. Система S2={¬,&} образует базис. Произвольную функцию можно сначала привести к виду, содержащему связки из S1 а затем использовать соотношение $x \vee y = \overline{\overline{x} \cdot \overline{y}}$.

3. Система S3={¬,∨} образует базис. Произвольную функцию можно сначала привести к виду, содержащему связки из S1 а затем использовать соотношение $x \cdot y = \overline{\overline{x} \vee \overline{y}}$.

4. Система S4={1,&,⊕} образует базис. Произвольную функцию можно сначала привести к виду, содержащему связки из S1 а затем использовать соотношения $\overline{x} = 1 \oplus x$, $x \vee y = x \oplus y \oplus x \cdot y$.

5. Система S5={|} образует базис. Произвольную функцию можно сначала привести к виду, содержащему связки из S2 а затем использовать соотношения $x \cdot y = \overline{\overline{x}|\overline{y}}$, $\overline{x} = x|x$.

6. Система S6={↓} образует базис. Произвольную функцию можно сначала привести к виду, содержащему связки из S3 а затем использовать соотношения $x \vee y = \overline{\overline{x} \downarrow \overline{y}}$, $\overline{x} = x \downarrow x$.

7. Система S7={→,0} образует базис.

Пример 1.7.1. Записать функцию $x \leftrightarrow (y \oplus z)$ в базисе S1={¬,∧,∨}.

$$x \leftrightarrow (y \oplus z) = (x \vee \overline{y \oplus z}) \cdot (\overline{x} \vee (y \oplus z)) = (x \vee \overline{\overline{y} \cdot z \vee y \cdot \overline{z}}) \cdot (\overline{x} \vee \overline{y} \cdot z \vee y \cdot z)$$

Глава II. Булева алгебра.

Множество всех булевых в базисе **S1={¬, &, ∨}** образуют **булеву алгебру**. Таким образом в булевой алгебре все формулы записываются при помощи трех связок: ¬, &, ∨. Частично свойства этой алгебры мы рассмотрели в главе I (см., например, основные эквивалентности). В этой главе рассматриваются свойства, специфичные для булевой алгебры и поэтому везде в этой главе мы будем иметь дело только с тремя функциями: **¬, &, ∨.**

§2.1. Нормальные формы.

Нормальные формы являются синтаксически однозначным способом записи формулы, реализующей заданную функцию.

Если x - логическая переменная, а $\sigma \in \{0,1\}$ то выражение

$$x^{\sigma} = \begin{cases} x & \text{если } \sigma = 1 \\ \overline{x} & \text{если } \sigma = 0 \end{cases} \quad \text{или} \quad x^{\sigma} = \begin{cases} 1 & \text{если } x = \sigma \\ 0 & \text{если } x \neq \sigma \end{cases},$$

называется литерой. Литеры x и ¬x называются контрарными. **Конъюнктом** называется конъюнкция литер. **Дизъюнктом** называется дизъюнкция литер. Например, формулы $x \cdot y \cdot \overline{z}$ и $x \cdot y \cdot x \cdot \overline{x}$ являются конъюнктами, формулы $x \vee y \vee \overline{z}$ и $x \vee y \vee \overline{x}$ - дизъюнктами, а формула $\bar{z}$ является одновременно и конъюнктом и дизъюнктом.

Дизъюнктивной нормальной формой (ДНФ) называется дизъюнкция конечного числа конъюнктов**.**

Конъюнктивной нормальной формой (КНФ) называется конъюнкция конечного числа дизъюнктов**.**

Более просто: ДНФ - это сумма произведений, а КНФ - это произведение логических сумм

Примеры.

1. $x \cdot y \vee y \cdot z \vee x$ — это ДНФ (сумма произведений).

2. $(x \vee \bar{y} \vee \bar{z}) \cdot (x \vee y) \cdot z$ —это КНФ (произведение сумм).

3. $\bar{x} \vee y \vee z \vee \bar{w}$ — это ДНФ и КНФ (одновременно).

4. $\bar{x} \cdot y \cdot \bar{z} \cdot w$ — это ДНФ и КНФ (одновременно).

5. $(x \vee x \vee y) \cdot (y \vee z \vee x) \cdot z$ — это КНФ.

6. $x \cdot y \cdot \bar{z}$ и $x \cdot y \cdot x \cdot \bar{x}$ — это ДНФ.

7. $x \cdot (x \vee yz) \cdot \overline{x \cdot \bar{y} \cdot z}$ — это не нормальная форма (не ДНФ и не КНФ).

Пусть функция f записана в базисе S1. Данная функция приводится к нормальной форме следующим путем:

1) используем законы де Моргана, чтобы преобразовать формулу к виду, в котором знаки отрицания относятся только к отдельным переменным;
2) применяем правило снятия двойного отрицания: ¬¬x=x;
3) далее использовать законы дистрибутивности, причем необходимо использовать первый закон дистрибутивности для приведения к ДНФ

   H1&(H2∨H3)=(H1&H2)∨(H1&H3) ,

   и второй закон дистрибутивности для приведения к КНФ.

   H1∨(H2&H3)=(H1∨H2)&(H1∨H3).

Любая булева функция может иметь бесконечно много представлений в виде ДНФ и КНФ. Например, используя дополнительно законы инверсий и правила операций с константами можно добиться, чтобы в каждом отдельном конъюнкте или дизъюнкте любая переменная входила бы не более одного раза (либо сама либо ее отрицание).

Пример 2.1.1. Для приведения к ДНФ используем 1-ый закон дистрибутивности.

$x \cdot \bar{y} \cdot \overline{x \cdot y \cdot \bar{z}} \cdot (y \vee z) = x \cdot \bar{y} \cdot (\bar{x} \vee \bar{y} \vee \bar{\bar{z}}) \cdot (y \vee z) = (x \cdot \bar{y} \cdot \bar{x} \vee x \cdot \bar{y} \cdot \bar{y} \vee x \cdot \bar{y} \cdot z) \cdot (y \vee z) =$ -это КНФ

$= (0 \vee x \cdot \bar{y} \vee x \cdot \bar{y} \cdot z) \cdot (y \vee z) = (x \cdot \bar{y} \vee x \cdot \bar{y} \cdot z) \cdot (y \vee z) =$ -это другая КНФ

$= x \cdot \bar{y} \cdot y \vee x \cdot \bar{y} \cdot z \cdot y \vee x \cdot \bar{y} \cdot z \vee x \cdot \bar{y} \cdot z \cdot z = 0 \vee 0 \vee x \cdot \bar{y} \cdot z \vee x \cdot \bar{y} \cdot z =$

$= x \cdot \bar{y} \cdot z \vee x \cdot \bar{y} \cdot z$ -это ДНФ

Пример 2.1.2. Для приведения к КНФ необходимо использовать второй закон дистрибутивности.

$$x \vee y \cdot \overline{x \cdot y \vee \bar{z}} = x \vee y \cdot (\overline{x \cdot y} \cdot \bar{\bar{z}}) = x \vee y \cdot (\bar{x} \vee \bar{y}) \cdot z =$$

$$= x \vee y \cdot z \cdot (\bar{x} \vee \bar{y}) = (x \vee y \cdot z) \cdot (x \vee \bar{x} \vee \bar{y}) = (x \vee y) \cdot (x \vee z) \cdot (1 \vee \bar{y}) =$$

$= (x \vee y) \cdot (x \vee z)$ это КНФ

§2.2. Совершенные нормальные формы.

Если в каждом члене нормальной формы представлены все переменные (либо сами, либо их отрицания), причем в каждом отдельном конъюнкте или дизъюнкте любая переменная входит ровно один раз (либо сама либо ее отрицание), то эта форма называется **совершенной нормальной формой (СДНФ или СКНФ).**

Примеры: Пусть задана функция трех переменных f(x,y,z).

1. $\overline{x} \cdot y \cdot z \vee x \cdot \overline{y} \cdot \overline{z} \vee \overline{x} \cdot \overline{y} \cdot \overline{z}$ – это совершенная ДНФ.

2. $(x \vee y \vee z) \cdot (x \vee \overline{y} \vee z) \cdot (\overline{x} \vee \overline{y} \vee z)$ – это совершенная КНФ.

3. $(x \vee y) \cdot (x \vee z)$ – это не совершенная КНФ, т.к. например, в первую сумму не входит переменная z.

4. $x \cdot y \cdot z$ – это совершенная ДНФ.

Теорема 2.2.1.

1. Любая булева функция, не являющаяся тождественным нулем, имеет только одну СДНФ, с точностью до расположения членов.

2. Любая булева функция, не являющаяся тождественной 1, имеет только одну СКНФ, с точностью до расположения членов.

Доказательство теоремы проведем конструктивно, как решение следующей задачи: **по данной таблице истинности построить СДНФ.**

Решение рассмотрим на примере функции f(x,y,z) заданной таблично (таб.2.2) при n=3.

Пример 2.2.1. По данной таблице истинности (таб.2.2) построить СДНФ.

Решение.

Таблица 2.2

x	y	z	основные конъюнкции	f(x,y,z)
0	0	0	$\overline{x} \cdot \overline{y} \cdot \overline{z}$	0
0	0	1	$\overline{x} \cdot \overline{y} \cdot z$	1
0	1	0	$\overline{x} \cdot y \cdot \overline{z}$	1
0	1	1	$\overline{x} \cdot y \cdot z$	0
1	0	0	$x \cdot \overline{y} \cdot \overline{z}$	0
1	0	1	$x \cdot \overline{y} \cdot z$	1
1	1	0	$x \cdot y \cdot \overline{z}$	1
1	1	1	$x \cdot y \cdot z$	1

Основные конъюнкции (или **конституенты_1**), включенные в таблицу, соответствуют конкретному набору нулей и единиц, которые принимают переменные x,y,z. Строятся конституенты_**1** по

следующему правилу: переменная входит в произведение сама, если на данном наборе она принимает значение 1, в противном случае в произведение входит ее отрицание. Так, например, на наборе (0,0,1) переменные x,y принимают значение 0 и поэтому в произведение входят их отрицания, а переменная z принимает значение 1 и поэтому входит в произведение сама. Для данного набора (0,0,1) конституента_1 равна $\bar{x}\cdot\bar{y}\cdot z$.

Очевидно, что конъюнкция $\bar{x}\cdot\bar{y}\cdot\bar{z}$ равна 1 только на наборе (0,0,0), а $\bar{x}\cdot\bar{y}\cdot z$ равна 1 на наборе (0,0,1) и т.д. (см. по таблице). Далее, заметим, что дизъюнкция двух основных конъюнкций равна 1 ровно на двух наборах, которые соответствуют данным основным конъюнкциям. Например, функция $\bar{x}\cdot\bar{y}\cdot\bar{z}\vee\bar{x}\cdot\bar{y}\cdot z$ равна 1 только на двух наборах – (0,0,0) и (0,0,1), а на всех других наборах она равна 0. Аналогично, дизъюнкция трех основных конъюнкций равна 1 ровно на трех наборах, которые соответствуют данным основным конъюнкциям и т.д.

Т.о. получаем **правило: для построения СДНФ** следует выбрать строки, в которых функция равна 1, а затем взять дизъюнкцию соответствующих основных конъюнкций, получим искомую СДНФ. Так для нашего примера, имеем

$\bar{x}\cdot\bar{y}\cdot z\vee\bar{x}\cdot y\cdot\bar{z}\vee x\cdot\bar{y}\cdot z\vee x\cdot y\cdot\bar{z}\vee x\cdot y\cdot z$.

Задача. По данной таблице истинности построить СКНФ.

Конституента_0 для набора нулей и единиц (которые принимают переменные x,y,z) строится следующим образом: переменная входит в дизъюнкцию сама, если на данном наборе она принимает значение **0**, в противном случае в произведение входит ее отрицание.

Правило для построения СКНФ: следует выбрать строки, в которых функция равна **0**, а затем взять конъюнкцию соответствующих конституент_0. В результате получится искомая СКНФ. Так для нашего примера, имеем

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

Замечание. Для построения совершенной КНФ функции f, достаточно построить совершенную ДНФ для функции $\bar{f}$, а затем использовать $f=\bar{\bar{f}}$ и законы де Моргана.

Построим СКНФ для нашего примера на основании замечания.

1. Строим СДНФ для отрицания.

$\bar{x}\cdot\bar{y}\cdot\bar{z}\vee\bar{x}\cdot y\cdot z\vee x\cdot\bar{y}\cdot\bar{z}$

2. Используем законы де Моргана

$$f=\bar{\bar{f}}=\overline{\bar{x}\cdot\bar{y}\cdot\bar{z}\vee\bar{x}\cdot y\cdot z\vee x\cdot\bar{y}\cdot\bar{z}}=\overline{\bar{x}\cdot\bar{y}\cdot\bar{z}}\,\&\,\overline{\bar{x}\cdot y\cdot z}\,\&\,\overline{x\cdot\bar{y}\cdot\bar{z}}=$$

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

Описанный способ нахождения СДНФ (и СКНФ) по таблице истинности бывает часто более трудоемким, чем следующий алгоритм.

1. **Для нахождения СДНФ** данную формулу приводим сначала к ДНФ.
2. Если в некоторый конъюнкт К (т.е. К это произведение некоторого числа переменных или их отрицаний) не входит скажем переменная y, то этот конъюнкт заменяем на эквивалентную формулу $K\&(y \vee \bar{y})$ и, применяя закон дистрибутивности, приводим полученную формулу к ДНФ; если недостающих переменных несколько, то для каждой из них к конъюнкту добавляем соответствующую формулу вида $(y \vee \bar{y})$.
3. Если в полученной ДНФ имеется несколько одинаковых конституент единицы, то оставляем только одну из них. В результате получается СДНФ.

Замечание: Для построения СКНФ в дизъюнкт не содержащий скажем переменную **y** добавляем формулу вида $y \cdot \bar{y}$, т.е. этот дизъюнкт заменяем на эквивалентную формулу $D \vee y \cdot \bar{y}$ и, применяя 2-й закон дистрибутивности.

Пример 2.2.2. Построить СДНФ для функции f при помощи эквивалентных преобразований.

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

ОТСТУПЛЕНИЕ.

Вычисление СДНФ имеет не только теоретическое, но и большое практическое значение. Например, во многих современных программах с графическим интерфейсом для составления сложных логических условий используется наглядный бланк в виде таблицы: в клетках записываются условия, причем клетки одного столбца считаются соединенными конъюнкцией, а столбцы — дизъюнкцией, то есть образуют ДНФ (или наоборот, в таком случае получается КНФ). В частности, так устроен графический интерфейс QBE (Query-by-Example), применяемый для формулировки логических условий при запросе к СУБД.

Алгоритм 2.2.1. Построение СДНФ

Вход: вектор X: array [1..n] of string - идентификаторов переменных,
матрица V: array [1..2^n, 1..n] of 0..1 всех различных наборов значений переменных,
вектор F: array [1..2^n] of 0..1 соответствующих значений функции.

Выход: последовательность символов, образующих запись формулы СДНФ для заданной функции.

```
f:= false { признак присутствия левого операнда дизъюнкции }
for i from 1 to 2^n do
```

```
if F[i] = 1 then
    if f then
        yield '∨' { добавление в формулу знака дизъюнкции;
                    оператор yield m выводит на печать
  символ m}
      else
        f:=true
    end if
    g:=false { признак присутствия левого операнда конъюнкции }
    for j from 1 to n do
      if g then
          yield '∧' {добавление в формулу знака конъюнкции }
       else
          g: =true
      end if
      if V[i,j} =0 then
         yield '¬' { добавление в формулу знака отрицания }
      end if
      yield X[j] { добавление в формулу идентификатора переменной
                                }
    end for
  end if
end for
```

§2.3. Минимизация ДНФ методом Квайна.

Каждая формула имеет конечное число вхождений переменных. Под вхождением переменной понимается место, которое переменная занимает в формуле. Задача заключается в том, чтобы для данной булевой функции f найти ДНФ, представляющую эту функцию и имеющую наименьшее число вхождений переменных.

Если x - логическая переменная, а $\sigma \in \{0,1\}$ то выражение

$$x^{\sigma} = \begin{cases} x & \text{если } \sigma = 1 \\ \overline{x} & \text{если } \sigma = 0 \cdot \end{cases}$$

называется **литерой**. **Конъюнктом** называется конъюнкция литер. Например, формулы $x \cdot y \cdot \overline{z}$ и $x \cdot y \cdot x \cdot \overline{x}$ являются конъюнктами**.** Элементарным произведением называется конъюнкт, в который любая переменная входит не более одного раза (либо сама либо ее отрицание).

Формула f1 называется **импликантой** формулы f, если f1 — элементарное произведение и $f1 \wedge f = f1$, т. е. для соответствующих формулам функций справедливо неравенство $f1 \le f$. Импликанта f1 формулы f называется **простой**, если после отбрасывания любой литеры из f1 не получается формула, являющаяся импликантой формулы f.

Пример 2.3.1. Найдем все импликанты и простые импликанты для формулы f=x→y. Всего имеется 8 элементарных произведений с переменными *x* и *y*. Ниже, для наглядности, приведены их таблицы истинности:

x	y	x→y	$x \cdot y$	$\overline{x} \cdot y$	$x \cdot \overline{y}$	$x \cdot y$	$\overline{x}$	$\overline{y}$	x	y
0	0	1	1	0	0	0	1	1	0	0
0	1	1	0	1	0	0	1	0	0	1
1	0	**0**	0	0	**1**	0	0	**1**	**1**	0
1	1	1	0	0	0	1	0	0	1	1

Из таблиц истинности заключаем, что формулы $\overline{x} \cdot \overline{y}$, $\overline{x} \cdot y$, $x \cdot y$, $\overline{x}$,y— все импликанты формулы x→y, а из этих импликант простыми являются формулы $\overline{x}$ и y (формула $\overline{x} \cdot \overline{y}$, например, не является простой импликантой, поскольку, отбрасывая литеру $\overline{y}$, получаем импликанту $\overline{x}$).

Сокращенной ДНФ называется дизъюнкция всех простых импликант данной формулы (функции)**.**

Теорема 2.3.1. Любая булева функция, не являющаяся константой 0, представима в виде сокращенной ДНФ.

В примере 2.3.1 функция, соответствующая формуле x→y представима формулой $\overline{x} \vee y$ которая является ее сокращенной ДНФ.

Сокращенная ДНФ может содержать лишние импликанты, удаление которых не меняет таблицы истинности. Если из сокращенной ДНФ удалить все лишние импликанты, то получается ДНФ, называемая **тупиковой.**

Заметим, что представление функции в виде тупиковой ДНФ в общем случае неоднозначно. Выбор из всех тупиковых форм, формы с наименьшим числом вхождений переменных дает **минимальную ДНФ {МДНФ).**

Рассмотрим метод **Квайна,** для нахождения МДНФ, представляющей данную булеву функцию. Определим следующие три операции:

1. операция полного склеивания*:*

$$f \cdot x \vee f \cdot \overline{x} = f \cdot (x \vee \overline{x}) = f \text{ ;}$$

2. операция неполного склеивания:

$$f \cdot x \vee f \cdot \overline{x} = f \cdot (x \vee \overline{x}) \vee f \cdot x \vee f \cdot \overline{x} = f \vee f \cdot x \vee f \cdot \overline{x} \text{ ;}$$

3. операция элементарного поглощения

$$f \cdot x^{\sigma} \vee f = f \text{ , } \quad \sigma \in \{0,1\} \text{ .}$$

Теорема 2.3.2 (теорема Квайна). Если исходя из СДНФ функции произвести все возможные операции неполного склеивания, а затем

элементарного поглощения, то в результате получится сокращенная ДНФ, т. е. дизъюнкция всех простых импликант.

Пример 2.3.2. Пусть функция f(x,y,z) задана совершенной ДНФ $f = \overline{x} \cdot y \cdot \overline{z} \vee \overline{x} \cdot y \cdot z \vee x \cdot \overline{y} \cdot z \vee x \cdot y \cdot \overline{z} \vee x \cdot y \cdot z$. Тогда, производя в два этапа все возможные операции неполного склеивания, а затем элементарного поглощения, имеем

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

Таким образом, сокращенной ДНФ функции f является формула $y \vee x \cdot z$.

На практике при выполнении операций неполного склеивания на каждом этапе можно не писать члены, участвующие в этих операциях, а писать только результаты всевозможных полных склеиваний и конъюнкты, не участвующие ни в одном склеивании.

Пример 2.3.3. Пусть функция f{x,y,z) задана совершенной ДНФ

$$f = \overline{x} \cdot \overline{y} \cdot \overline{z} \vee \overline{x} \cdot \overline{y} \cdot z \vee x \cdot \overline{y} \cdot z \vee x \cdot y \cdot z .$$

Тогда, производя операции склеивания, а затем элементарного поглощения, имеем

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

Для получения минимальной ДНФ из сокращенной ДНФ используется матрица Квайна, которая строится следующим образом. В заголовках столбцов таблицы записываются конституенты единицы совершенной ДНФ, а в заголовках строк — простые импликанты из полученной сокращенной ДНФ. В таблице звездочками отмечаются те пересечения строк и столбцов, для которых конъюнкт, стоящий в заголовке строки, входит в конституенту единицы, являющейся заголовком столбца.

Для примера 2.3.3. матрица Квайна имеет вид

импликанты	$\bar{x}\cdot\bar{y}\cdot\bar{z}$	$\bar{x}\cdot\bar{y}\cdot z$	$x\cdot\bar{y}\cdot z$	$x\cdot y\cdot z$
$\bar{x}\cdot\bar{y}$	*	*		
$\bar{y}\cdot z$		*	*	
$x\cdot z$			*	*

В тупиковую ДНФ выбирается минимальное число простых импликант, дизъюнкция которых сохраняет все конституенты единицы, т. е. каждый столбец матрицы Квайна содержит звездочку, стоящую на пересечении со строкой, соответствующей одной из выбранных импликант. В качестве минимальной ДНФ выбирается тупиковая, которая имеет наименьшее число вхождений переменных.

В примере 2.3.3 по матрице Квайна находим, что минимальная ДНФ заданной функции есть $\bar{x}\cdot\bar{y}\vee x\cdot z$.

Замечание. Для построения минимальной КНФ функции f, достаточно построить минимальную ДНФ для функции $\bar{f}$, а затем использовать $f=\bar{\bar{f}}$ и законы де Моргана.

§ 2.4. Карты Карно.

Другой способ получения простых импликант формул с малым числом переменных (и, значит, нахождения минимальной ДНФ) основан на использовании так называемых карт Карно.

Карта Карно - это специального вида таблица, которая позволяет упростить процесс поиска минимальных форм и успешно применяется, когда число переменных не превосходит шести. Карты Карно для функций, зависящих от n переменных, представляет собой прямоугольник, разделенный на 2^n клеток. Каждой клетке диаграммы ставится в соответствие двоичный n-мерный набор. Значения заданной функции f из таблицы истинности вносятся в нужные квадраты, однако если клетке соответствует 0, то обычно она остается пустой.

В таб.2.4.1. показан пример разметки карты Карно для функции, зависящей от трех переменных. Нижние четыре клетки карты соответствуют двоичным наборам, в которых переменная **x** принимает значение 1, четыре верхние клетки соответствуют наборам, в которых переменная **x** принимает значение 0. Четырем клеткам составляющим правую половину карты, соответствуют наборы, в которых переменная y; принимает значение 1 и т.д. В таб.2.4.2. приведена разметка карты Карно для n=4 переменных.

таблица 2.4.1.

y z / x	0 0	0 1	1 1	1 0
0	000	001	011	010
1	100	101	011	010

таблица 2.4.2.

Z w / x y	0 0	0 1	1 1	1 0
00	0000	0001	0011	0010
01	0100	0101	0011	0010
11	1100	1101	1111	1110
10	1000	1001	1011	1010

Для построения минимальной ДНФ производится **процедура склеивания "1".** Склеивающимся значениям "1" соответствуют соседние клетки, т.е. клетки отличающиеся лишь значением одной переменной (на графическом изображении разделенных вертикальной или горизонтальной линией с учетом соседства противоположных крайних клеток).

Процесс склеивания "1" сводится к объединению в группы единичных клеток карты Карно, при этом необходимо выполнять следующие правила;

1. Количество клеток, входящих в одну группу, должно выражаться числом кратным 2, т.е. 2^m где m=0,1,2,...
2. Каждая клетка, входящая в группу из 2^m клеток, должна иметь m соседних в группе.
3. Каждая клетка должна входить хотя бы в одну группу.
4. В каждую группу должно входить максимальное число клеток, т.е. ни одна группа не должна содержаться в другой группе.
5. Число групп должно быть минимальным.

Считывание функции f по группе склеивания производится следующим образом: переменные, которые сохраняют одинаковые значения в клетках группы склеивания, входят в конъюнкцию, причем значениям 1 соответствуют сами переменные, а значениям 0 их отрицания.

Приведем шаблоны, которые помогают строить покрытия 1 (переменные считаем теми же, но их писать не будем). Для упрощения записи мы не будем отмечать переменные, хотя сохраним их обозначения как и в таблицах 2.4.1,2.4.2.

A) n=3

F=z

		1	1
		1	1

f=¬x

1	1	1	1

F=¬y

1			1
1			1

F=¬z&¬y

1			
1			

f=¬x&y

	1	1	

F=¬y&x

1			1

B) n=4

F=w

		1	1
		1	1
		1	1
		1	1

f=¬y

1	1	1	1
1	1	1	1

F=¬z

1			1
1			1
1			1
1			1

F=¬x&z

		1	1
		1	1

f=y&w

	1	1	
	1	1	

F=¬x&¬y

1	1	1	1

F=¬y&¬w

1			1
1			1

f=¬y&¬z

1	1		
1	1		

F=¬z&¬x

1			1
1			1

F=y&z&w

		1	
		1	

f=¬y&¬z&¬w

1			
1			

F=x&y&¬z

1			1

Пример 2.4.1. Построить МДНФ.

Z w x y	0 0	0 1	1 1	1 0
00	1	1		1
01	1	1	1	
11		1	1	
10				1

Сначала смотрим, есть ли покрытия_**1** из 16 клеток покрывающих хотя бы одну непокрытую 1. Таких покрытий нет. Переходим к покрытиям из 8 клеток. Смотрим, есть ли покрытия 1 из 8 клеток покрывающих хотя бы одну непокрытую 1. Таких покрытий нет. Переходим к покрытиям из 4 клеток. Смотрим, есть ли покрытия 1 из 4 клеток покрывающих хотя бы одну непокрытую 1. Таких покрытий два. Переходим к покрытиям из 2 клеток. Такое покрытие одно. Таким образом, все 1 стали покрытыми. Далее, смотрим можно ли убрать некоторые покрытия, так чтобы все единицы остались покрытыми. В конце выписываем МДНФ: $f=\bar{x}\cdot\bar{z}\vee y\cdot w\vee\bar{y}\cdot z\cdot\bar{w}$.

Замечание. Для построения минимальной КНФ функции f, достаточно построить минимальную ДНФ для функции $\bar{f}$, а затем использовать $f=\bar{\bar{f}}$ и законы де Моргана.

Глава III. Алгебра Жегалкина.

Множество булевых функций, заданный в базисе Жегалкина $S4=\{\oplus,\&,1\}$ называется алгеброй Жегалкина.

Основные свойства.

1. коммутативность
   $H1\oplus H2=H2\oplus H1$, $H1\&H2=H2\&H1$;
2. ассоциативность
   $H1\oplus(H2\oplus H3)=(H1\oplus H2)\oplus H3$, $H1\&(H2\&H3)=(H1\&H2)\&H3$;
3. дистрибутивность
   $H1\&(H2\oplus H3)=(H1\&H2)\oplus(H1\&H3)$;
4. свойства констант
   $H\&1=H$, $H\&0=0$, $H\oplus 0=H$;
5. $H\oplus H=0$, $H\&H=H$.

Утверждение 3.1.1. Через операции алгебры Жегалкина можно выразить все другие булевы функции:

$\neg x=1\oplus x$, $x\vee y=x\oplus y\oplus xy$, $x\sim y=1\oplus x\oplus y$,
$x\to y=1\oplus x\oplus xy$, $x\downarrow y=1\oplus x\oplus y\oplus xy$, $x|y=1\oplus xy$.

Определение. Полиномом Жегалкина (полиномом по модулю 2) от n переменных $x_1,x_2,\ldots,x_n$ называется выражение вида

$$c_0\oplus c_1x_1\oplus c_2x_2\oplus\ldots\oplus c_nx_n\oplus c_{12}x_1x_2\oplus\ldots\oplus c_{12\ldots n}x_1x_2\ldots x_n,$$

где постоянные $c_к$ могут принимать значения 0 ли 1.

Если полином Жегалкина не содержит произведений отдельных переменных, то он называется линейным (линей ная функция).

Например, $f=x\oplus yz\oplus xyz$ и $f1=1\oplus x\oplus y\oplus z$–полиномы, причем второй является линейной функцией.

Теорема 3.1.1. Каждая булева функция представляется в виде полинома Жегалкина единственным образом.

Приведем основные методы построения полиномов Жегалкина от заданной функции.

1. Метод неопределенных коэффициентов. Пусть $P(x_1,x_2,\ldots,x_n)$ - искомый полином Жегалкина, реализующий заданную функцию $f(x_1,x_2,\ldots,x_n)$. Запишем его в виде

$$P= c_0\oplus c_1x_1\oplus c_2x_2\oplus\ldots\oplus c_nx_n\oplus c_{12}x_1x_2\oplus\ldots\oplus c_{12\ldots n}x_1x_2\ldots x_n.$$

Найдем коэффициенты $c_к$. Для этого последовательно придадим переменным $x_1,x_2,\ldots,x_n$ значения из каждой строки таблицы истинности. В итоге получим систему из 2^n уравнений с 2^n неизвестными, имеющую единственное решение. Решив ее, находим коэффициенты полинома $P(x_1,x_2,\ldots,x_n)$.

2. Метод, основанный на преобразовании формул над множеством связок {¬,&}. Строят некоторую формулу Ф над множеством связок{¬,&}, реализующую данную функцию $f(x_1,x_2,\ldots,x_n)$. Затем заменяют всюду подформулы вида $\overline{A}$ на $A\oplus1$, раскрывают скобки, пользуясь дистрибутивным законом (см. свойство 3), а затем применяют свойства 4 и 5.

Пример 3.1.1. Построить полином Жегалкина функции $f(x,y)=x\to y$.

Решение. 1. (метод неопределенных коэффициентов). Запишем искомый полином в виде

$$P(x,y)= c_0\oplus c_1x\oplus c_2y\oplus c_{12}xy$$

Пользуясь таблицей истинности

x	0	0	1	1
y	0	1	0	1
x→y	1	1	0	1

получаем, что

$f(0,0)=P(0,0)= c_0=1$, $f(0,1)=P(0,1)= c_0\oplus c_2=1$,

$f(1,0)=P(1,0)= c_0\oplus c_1=0$, $f(1,1)=P(1,1)= c_0\oplus c_1\oplus c_2\oplus c_{12}=1$

Откуда последовательно находим, $c_0=1$, $c_1=1$, $c_2=0$, $c_{12}=1$. Следовательно

$x\to y=1\oplus x\oplus xy$ (сравните с утверждением 3.1).

2.(Метод преобразования формул.) Имеем

$$x \to y = \overline{x} \vee y = \overline{x \cdot \overline{y}} = (x \cdot (y \oplus 1)) \oplus 1 = 1 \oplus x \oplus x \cdot y .$$

Заметим, что преимущество алгебры Жегалкина (по сравнению с другими алгебрами) состоит в арифметизации логики, что позволяет выполнять преобразования булевых функций довольно просто. Ее недостатком по сравнению с булевой алгеброй является громоздкость формул.

Глава IV. Высказывания. Предикаты.

§4.1. Высказывания.

При построении алгебры логики мы использовали функциональный подход. Однако, можно было бы построить эту алгебру конструктивно. Сначала определить объекты изучения (высказывания), ввести операции над этими объектами и изучить их свойства. Дадим формальные определения.

Высказыванием назовем повествовательное предложение относительно которого можно однозначно сказать истинно оно (значение И или 1) или ложно (значение Л или 0) в конкретный момент времени. Например, «5-простое число», «нажата клавиша «Esc»» и т.д. При помощи связок «не», «и», «или», «если,… то», «если и только если» (им отвечают операции «¬», «&», «∨», «→», «~» соответственно) можно построить более сложные высказывания (предложения). Так строится алгебра высказываний.

Для упрощения записи сложных высказываний вводится старшинство связок: «¬», «&», «∨», «→», «~», что помогает опустить лишние скобки.

Простые высказывания назовем пропозициональными переменными.

Введем понятие формулы.

1. Пропозициональные переменные являются формулами.
2. Если A, B формулы, то выражения ¬A, A∧B, A∨B, A→B, A~B являются формулами.
3. Формулами являются только выражения построенные в соответствии с пп.1 и 2.

Формула, принимающая значение И при всех значениях пропозициональных переменных называется **тавтологией (или общезначимой),** а формула, принимающая значение Л при всех значениях пропозициональных переменных называется **противоречием (или невыполнимой)**

Описание свойств алгебры высказываний аналогично описанию соответствующих функций в булевой алгебре и мы их опускаем.

§4.2. Предикаты. Логические операции над предикатами.

Предметом изучения в этой главе будут предикаты — отображения произвольных множеств во множество высказываний. Фактически, мы совершаем переход на новый уровень абстракции, переход такого типа, какой был совершен в школе — от арифметики вещественных чисел к алгебре числовых функций.

Определение 2.1 Пусть $x_1,x_2,\ldots,x_n$ — символы переменных произвольной природы. Эти переменные будем называть предметными. Пусть наборы переменных $(x_1,x_2,\ldots,x_n)$ принадлежат множеству M=(M1,M2,...Mn), которое будем называть предметной областью (т.е. $x_i \in M_i$, где M_i называются областью определения переменной x_i). Предикатом местности n (n-местным предикатом), определенным на предметной области M, называют логическую функцию принимающую либо значение И либо значение Л.

Пример 4.2.1. D(x1,x2) = «Натуральное число x1 делится (без остатка) на натуральное число x2.» — двуместный предикат, определенный на множестве пар натуральных чисел M=N×N. Очевидно, D(4,2) = И, D(3,5) = 0.

Пример 4.2.2. Q(x) ==«$x^2<-1$, $x \in R$» — одноместный предикат, определенный на множестве действительных чисел M=R. Ясно, что Q(-1) = Л, Q(5) = Л, и вообще предикат Q(x) — тождественно ложен, т. е. Q(x) = Л для всех $x \in R$.

Пример 4.2.3. B(x,y,z) =«$x^2+y^2<z$; $x,y,z \in R$.» — трехместный предикат, определенный на R^3. B(1,1,-2)=Л, B(1,1,2)=И.

Предикат P, определенный на M, называется тождественно истинным, если он принимает значение И при любых значениях предметных переменных; Предикат P называется тождественно ложным, если он принимает значение Л при любых значениях предметных переменных. Предикат Q из примера 4.2.2. является тождественно ложным.

Поскольку предикаты — это функции со значениями во множестве высказываний, где введены логические операции, то эти операции естественно определяются и для предикатов. Пусть P и Q — предикаты, определенные на M. Тогда

1. $\neg P(x_1,x_2,\ldots,x_n) = \overline{P(x_1,x_2,\ldots,x_n)}$
2. $(P \wedge Q)(x_1,x_2,\ldots,x_n) = P(x_1,x_2,\ldots,x_n) \wedge Q(x_1,x_2,\ldots,x_n)$
3. $(P \vee Q)(x_1,x_2,\ldots,x_n) = P(x_1,x_2,\ldots,x_n) \vee Q(x_1,x_2,\ldots,x_n)$
4. $(P \to Q)(x_1,x_2,\ldots,x_n) = P(x_1,x_2,\ldots,x_n) \to Q(x_1,x_2,\ldots,x_n)$

Предикаты P и Q, определенные на M, называются равносильными (пишут P=Q), если $P(x_1,x_2,\ldots,x_n)=Q(x_1,x_2,\ldots,x_n)$ для любого набора $(x_1,x_2,\ldots,x_n)$ предметных переменных из M.

Теорема 4.2.1 Множество n-местных предикатов, определенных на M, образует булеву алгебру предикатов. Т.о., для них справедливы основные эквивалентности (см. §1.6).

§4.3. Кванторы, их свойства.

Пусть $P(x_1,x_2,\ldots,x_n)$ — n-местный предикат, определенный на M. Зафиксируем $x_i=a$. Определим (n-1)-местный предикат $Q(x_1,x_2,\ldots,x_{k-1}, x_{k+1},x_n)$ следующим образом: $Q(x_1,x_2,\ldots,x_{k-1},x_{k+1},x_n)=P(x_1,x_2,\ldots,x_{k-1},a,x_{k+1},x_n)$. Говорят, что предикат $Q(x_1,x_2,\ldots,x_{k-1}, x_{k+1},x_n)$ получен из предиката $P(x_1,x_2,\ldots,x_n)$ фиксацией значения i-й переменной: $x_i=a$.

Определение 4.3.1. Пусть P(x) — одноместный предикат. Поставим ему в соответствие высказывание, обозначаемое ∀xP(x) (читается «для любого x P(x)»}, которое истинно тогда и только тогда, когда P(x) — тождественно истинный предикат. О высказывании ∀xP(x) говорят, что оно получено из предиката P навешиванием квантора всеобщности по переменной x.

Определение 4.3.2. Пусть P(x) — одноместный предикат. Поставим ему в соответствие высказывание, обозначаемое ∃xP(x) (читается «существует x P(x)»), которое ложно тогда и только тогда, когда P(x) — тождественно ложный предикат. О высказывании ∃xP(x) говорят, что оно получено из предиката P навешиванием квантора существования по переменной x.

Замечание 1. Обозначения ∀ и ∃ для кванторов — это перевернутые латинские буквы A и E соответственно, которые являются первыми буквами в английских словах **All** — все, **Exist** — существовать.

Замечание 2. Высказывания можно считать предикатами, не содержащими переменных, т. е. 0-местными предикатами (или предикатами любой местности).

Пусть $P(x_1,x_2,\ldots,x_n)$ — n-местный предикат, определенный на M. Зафиксируем в нем значения переменных $x_1,x_2,\ldots,x_{k-1},x_{k+1},x_n$. На полученный одноместный предикат $Q(x_к)$ навесим квантор всеобщности (существования), получим высказывание. Тем самым фиксированному набору значений переменных $x_1,x_2,\ldots,x_{k-1},x_{k+1},x_n$ с помощью квантора всеобщности (существования) поставлено в соответствие высказывание. Говорят, что этот (n-1)-местный предикат переменных $x_1,x_2,\ldots,x_{k-1},x_{k+1},x_n$ получен из исходного предиката $P(x_1,x_2,\ldots,x_n)$ навешиванием квантора всеобщности (существования) по k-й переменной. Этот предикат обозначают: $\forall x_к P(x_1,x_2,\ldots,x_n)$ ($\exists x_к P(x_1,x_2,\ldots,x_n)$). Об к-й переменной (которой уже нет) говорят, что она связана квантором всеобщности (существования).

Пример 4.3.1. Пусть D(x1,x2) = «Натуральное число x1 делится (без остатка) на натуральное число x2.» — двухместный предикат. Навесим последовательно на его переменные кванторы. Ясно, что

1) $\forall x1\forall x2D(x1,x2)=0$ 2) $\forall x2\forall x1D(x1,x2)=0$ 3) $\exists x1\exists x2D(x1,x2)=1$
4) $\exists x2\exists x1D(x1,x2)=1$ 5) $\forall x1\exists x2D(x1,x2)=1$ 6) $\exists x2\forall x1D(x1,x2)=1$
7) $\exists x1\forall x2D(x1,x2)=0$ 8) $\forall x1\exists x2D(x1,x2)=1$.

Таким образом (сравнением 7 и 8 в последнем примере) мы доказали теорему:

Обычно, связки и кванторы упорядочиваются по приоритету следующим образом ¬, ∀, ∃, &, ∨, →, ~.

Теорема 4.3.1. Разноименные кванторы, вообще говоря, не коммутируют.

Теорема 4.3.2. (основные равносильности, содержащие кванторы) Имеют место следующие равносильности:

1. Законы де Моргана

$$\overline{\forall x P(x)} = \exists x \overline{P(x)}, \qquad \overline{\exists x P(x)} = \forall x \overline{P(x)}$$

2. Коммутативность

$$\forall x\forall y P(x,y) = \forall y\forall x P(x,y), \qquad \exists x\exists y P(x,y) = \exists y\exists x P(x,y)$$

3. Дистрибутивность

$$\forall x(P(x) \& Q(x)) = \forall x P(x) \& Q(x), \qquad \exists x(P(x) \vee Q(x)) = \exists x P(x) \vee Q(x)$$

4. Ограничения действия кванторов

$$\forall x(P(x) \vee Q(y)) = \forall x P(x) \vee \forall x Q(y), \qquad \exists x(P(x) \& Q(y) = \exists x P(x) \& \exists x Q(y)$$

5. Для любого двухместного предиката

$$\exists y\forall x P(x,y) \to \forall x\exists y P(x,y) = 1$$

Глава V. Формальные теории.

§5.1. Определение формальной теории.

Формальная теория (или **исчисление) Ψ**— это:

1. множество ***A*** символов, образующих **алфавит***;*

1. множество ***F*** слов в алфавите ***A, F⊂A***, которые называются **формулами***;*

3. подмножество ***B*** формул, ***B⊂F,*** которые называются **аксиомами;**

4. множество отношений ***R*** на множестве формул, которые называются **правилами вывода.**

Множество символов ***A*** может быть конечным или бесконечным. Обычно для образования символов используют конечное множество букв, к которым, если нужно, приписываются в качестве индексов натуральные числа.

Множество формул ***F*** обычно задается индуктивным определением, например, с помощью формальной грамматики. Как правило, это множество бесконечно. Множества ***A*** и ***F*** в совокупности определяют **язык**, или **сигнатуру**, формальной теории.

Множество аксиом ***B*** может быть конечным или бесконечным. Если множество аксиом бесконечно, то, как правило, оно задается с помощью конечного множества схем аксиом и правил порождения конкретных аксиом из схемы аксиом.

Множество правил вывода ***R***, как правило, конечно.

Итак, исчисление **Ψ** есть четверка ***(A, F, B, R)***.

Выводом в исчислении **Ψ** называется последовательность формул $F_1,F_2,\ldots,F_n$ такая, что для любого k ($1 \le k \le n$) формула F_k есть либо аксиома исчисления Ψ, либо непосредственное следствие каких-либо предыдущих формул, полученное по правилу вывода.

Формула G называется теоремой исчисления Ψ (выводимой в Ψ или доказуемой в Ψ), если существует вывод $F_1,F_2,\ldots,F_n,G$ который называется выводом формулы G или доказательством теоремы G. Записывается это следующим образом:

$$F_1,F_2,\ldots,F_n+ G.$$

Исчисление **Ψ** называется **непротиворечивым**, если не все его формулы доказуемы. Можно дать другое определение непротиворечивости: Исчисление Ψ называется непротиворечивым, если в нем не являются выводимыми одновременно формулы F и ¬F (отрицание F).

Исчисление **Ψ** называется **полным** (или адекватным), если каждому истинному высказыванию M соответствует теорема теории **Ψ**.

Формальная теория **Ψ** называется **разрешимой**, если существует алгоритм, который для любой формулы теории определяет, является ли эта формула теоремой теории **Ψ** или нет.

§5.2. Исчисление высказываний.

Используя понятие формального исчисления, определим исчисление высказываний (ИВ).

Алфавит ИВ состоит из

1. **букв** A, B, Q, R, P и других, возможно с индексами (которые называются пропозициональными переменными},
2. **логических символов** (связок) ¬, &, ∨, →,
3. вспомогательных **символов** (,).

Множество формул ИВ определяется индуктивно:

1. все пропозициональные переменные являются формулами ИВ;
2. если A, B —.формулы ИВ**,** то¬A, A∧B, A∨B, A→B — формулы ИВ**;**
3. выражение является формулой ИВ тогда и только тогда, когда это может быть установлено с помощью пунктов "1" и "2".

Таким образом, любая формула'ИВ строится из пропозициональных переменных с помощью связок ¬, ∧, ∨, →.

В дальнейшем при записи формул будем опускать некоторые скобки, используя те же соглашения, что и в предыдущей главе.

Аксиомами ИВ являются следующие формулы (для любых формул A,B,C)

1. A→(B→A);
2. (A→B)→((A→(B→C))→(A→C));
3. (A∧B)→A;
4. (A∧B)→B;
5. (A→B)→((A→C)→(A→(B∧C)));
6. A→(A∨B);
7. A→(B∨A);
8. (A→C)→((B→C)→((A∨B)→C));
9. (A→B)→((A→¬B)→¬A);
10. ¬¬A→A.

Указанные формулы называются схемами аксиом ИВ. При подстановке конкретных формул в какую-либо схему получается частный случай схемы аксиом.

Правилом вывода в ИВ является правило заключения {modus ponens):

если A и A→B — выводимые формулы, то B — также выводимая формула.

Символически это записывается так: $\frac{A,\ A \to B}{B}$.

Например, если высказывания A∧B и A∧B→(A→C) выводимы, то высказывание A→C также выводимо согласно правилу заключения.

Говорят, что формула G выводима из формул $F_1,F_2,\ldots,F_n$ (обозначается $F_1,F_2,\ldots,F_n$+G), если существует последовательность формул $F_1,F_2,\ldots,F_k$,G, в которой любая формула является либо аксиомой, либо принадлежит списку формул $F_1,F_2,\ldots,F_n$ (называемых гипотезами), либо получается из предыдущих формул по правилу вывода. Выводимость формулы G из ∅ (обозначается +G) равносильно тому, что G — теорема ИВ.

Пример 5.2.1. Покажем, что формула A→A выводима в ИВ. Для этого построим вывод данной формулы:

1) в аксиоме 2 заменим B на A→A, C — на A. Получаем аксиому
(A→(A→A))→((A→((A→A)→A))→(A→A));
2) в аксиоме 1 заменим B на A. Получаем
A→(A→A);
3) из 1 и 2 по modus ponens заключаем
(A→((A→A)→A))→(A→A);
4) в аксиоме 1 заменяем B на A→A. Получаем

$A \to ((A \to A) \to A)$;

5) из пп. 3 и 4 по правилу вывода справедливо $+ A \to A$.

Теорема 5.2.1.

1. Если $F_1,F_2,\ldots,F_n,A,B$ — формулы ИВ, $\Gamma=\{F_1,F_2,\ldots,F_n\}$, $\Gamma+A$, то $\Gamma,B+A$. (можно увеличивать число гипотез).
2. Тогда и только тогда $F_1,F_2,\ldots,F_n+A$, когда $F_1 \wedge F_2 \wedge \ldots \wedge F_n+A$ (сведение множества гипотез к одной гипотезе).

§5.3. Теорема о дедукции. Полнота ИВ.

Теорема 5.3.1. (теорема о дедукции)

Если $\Gamma,B+A$, то $\Gamma+B \to A$, где Γ — набор некоторых формул $\Gamma=\{F_1,F_2,\ldots,F_n\}$.

Следствие 5.3.1. Тогда и только тогда $F_1,F_2,\ldots,F_n+A$, когда $+ F_1 \to (F_2 \to \ldots \to (F_{n-1} \to (F_n \to A))\ldots)$

Доказательство. Пусть $F_1,F_2,\ldots,F_n+A$. Тогда, применяя теорему о дедукции, имеем $F_1,F_2,\ldots,F_{n-1}+F_n \to A$. Аналогично $F_1,F_2,\ldots,F_{n-2}+F_{n-1} \to (F_n \to A)$, и т. д. Продолжая процесс необходимое число раз, получаем

$$+ F_1 \to (F_2 \to \ldots \to (F_{n-1} \to (F_n \to A))\ldots)$$

Для доказательства достаточности предположим, что $+B$, где $B=F_1 \to (F_2 \to \ldots \to (F_{n-1} \to (F_n \to A))\ldots)$. Воспользуемся теоремой 5.2.1., п.1: F_1+B. По правилу заключения получаем $F_1+ (F_2 \to \ldots \to (F_{n-1} \to (F_n \to A))\ldots)$, Далее через n шагов имеем $F_1,F_2,\ldots,F_n+A$.

Таким образом, благодаря следствию 5.3.1., проверка выводимости формулы A из формул $F_1,F_2,\ldots,F_n$, сводится к проверке доказуемости формулы

$$F_1 \to (F_2 \to \ldots \to (F_{n-1} \to (F_n \to A))\ldots).$$

Напомним, что формула A называется тождественно истинной (или тавтологией), если значение формулы A равно единице при любых наборах значений пропозициональных переменных. Следующая теорема сводит проверку доказуемости формулы к проверке ее тождественной истинности.

Теорема 5.3.2. (о полноте). Формула A доказуема тогда и только тогда, когда A тождественно истинна (тавтология): $+A \Leftrightarrow$ A – тавтология.

Таким образом, для того чтобы установить, доказуема ли формула, достаточно составить ее таблицу истинности. Как известно, существует эффективный алгоритм построения таблицы истинности, и, значит, ИВ **разрешимо.**

Пример 5.3.1. Докажем, что $P+P$. По теореме о дедукции это равносильно тому, что $+(P \to P)$. В свою очередь, по теореме о полноте,

достаточно доказать, что (P→P) тавтология. Составляя таблицу истинности для формулы (P→P), убеждаемся, что (P→P) тождественно истинна и, следовательно, доказуема.

Теорема 5.3.3. (о непротиворечивости). Исчисление ИВ непротиворечиво.

Доказательство. По теореме о полноте любая формула, не являющаяся тождественно истинной, не доказуема в ИВ. Например, такой формулой является формула A∧(¬A).

Множество формул Г называется **противоречивым**, если Г+A∧(¬A). Если Г — противоречивое множество формул, будем обозначать этот факт через Г+.

Утверждение 5.3.1. Формула A выводима из множества формул Г тогда и только тогда, когда множество Г∪{¬A} — противоречиво.

§5.4. Автоматическое доказательство теорем.

Автоматическое доказательство теорем — это краеугольный камень логического программирования, искусственного интеллекта и других современных направлений в программировании. Вообще говоря, может не существовать алгоритма, с помощью которого для произвольной формулы A через конечное число шагов можно определить, является ли A выводимой в исчислении Ψ или нет. Однако, для некоторых простых формальных теорий (например исчисление высказываний) и некоторых простых классов формул (например прикладное исчисление предикатов с одним одноместным предикатом) алгоритмы автоматического доказательства теорем известны. Ниже, на примере исчисления высказываний, излагаются основы метода резолюций — классического и в то же время популярного метода автоматического доказательства теорем.

§5.5. Метод резолюций в ИВ.

Напомним, что если x - логическая переменная, а $\sigma\in\{0,1\}$ то выражение

$$x^{\sigma}=\begin{cases} x & \text{если } \sigma=1 \\ \overline{x} & \text{если } \sigma=0 \end{cases} \qquad \text{или} \qquad x^{\sigma}=\begin{cases} 1 & \text{если } x=\sigma \\ 0 & \text{если } x\neq\sigma \end{cases},$$

называется **литерой**. Литеры x и ¬x называются **контрарными. Конъюнктом** называется конъюнкция литер. **Дизъюнктом** называется дизъюнкция литер.

Пусть $D_1=B_1\vee A$, $D_2=B_2\vee\overline{A}$ — дизъюнкты. Дизъюнкт $B_1\vee B_2$ называется **резольвентой** дизъюнктов D_1 и D_2 по литере A и обозначается через $res_A(D_1,D_2)$. Резольвентой дизъюнктов D_1 и D_2

называется резольвента по некоторой литере и обозначается через $res(D_1,D_2)$. Очевидно, что $res(A,\neg A)=0$. Действительно, т.к. $A=A\vee 0$ и $\neg A=\neg A\vee 0$, то $res(A,\neg A)=0\vee 0=0$. Если дизъюнкты D_1 и D_2 не содержат контрарных литер, то резольвент у них не существует.

Пример 5.5.1. Если

$D_1=A\vee B\vee C$, $D_2=\overline{A}\vee\overline{B}\vee Q$, то

$res_A(D_1,D_2)=B\vee C\vee\overline{B}\vee Q$, $res_B(D_1,D_2)=A\vee C\vee\overline{A}\vee Q$,

$res_C(D_1,D_2)$ не существует.

Утверждение 5.5.1. Если $res(D_1,D_2)$ существует, то $D_1,D_2+ res(D_1,D_2)$.

Пусть $S=(D_1,D_2,\ldots,D_n)$ – множество дизъюнктов. Последовательность формул $F_1,F_2,\ldots,F_n$ называется резолютивным выводом из S, если для каждой формулы F_k выполняется одно из условий:

1. $F_k\in S$;
2. существуют j, k <i такие, что $F_i=res(F_j,F_k)$.

Теорема 5.5.1. (о полноте метода резолюций). Множество дизъюнктов S противоречиво в том и только в том случае, когда существует, резолютивный вывод из S, заканчивающийся 0.

Отметим, что метод резолюций можно использовать для проверки выводимости формулы F из данного множества формул $F_1,F_2,\ldots,F_n$. Действительно, условие $F_1,F_2,\ldots,F_n+F$ равносильно условию $F_1,F_2,\ldots,F_n,\neg F+$ (множество формул противоречиво), что в свою очередь равносильно условию Q+, где $Q=F_1\wedge F_2\wedge\ldots\wedge F_n\wedge(\neg F)$. Приведем формулу Q к КНФ: $Q=D_1\wedge D_2\wedge\ldots\wedge D_m$, тогда $Q+ \Leftrightarrow D_1\wedge D_2\wedge\ldots\wedge D_m+ \Leftrightarrow D_1,D_2,\ldots,D_m+$. Таким образом, задача проверки выводимости $F_1,F_2,\ldots,F_n+F$ сводится к проверке противоречивости множества дизъюнктов $S=\{D_1,D_2,\ldots,D_m\}$, что равносильно существованию резолютивного вывода 0 из S.

Пример 5.5.2. Проверить методом резолюций соотношение

$$A\to(B\to C),\ CD\to E,\ \neg F\to D\&(\neg)E\ +\ A\to(B\to F).$$

Согласно утверждению 5.3.1. надо проверить на противоречивость множество формул

$$S=\{A\to(B\to C),\ CD\to E,\ \neg F\to D\&(\neg)E,\ \neg(A\to(B\to F))\}.$$

Приведем все формулы из. S к КНФ:

$$S=\{\overline{A}\vee\overline{B}\vee C,\overline{C\cdot D}\vee E,F\vee D\cdot\overline{E},\overline{\overline{A}\vee\overline{B}\vee F}\}=$$

$$=\{\overline{A}\vee\overline{B}\vee C,\overline{C}\vee\overline{D}\vee E,(F\vee D)\cdot(F\vee\overline{E}),A\cdot B\cdot\overline{F}\}.$$

Таким образом, получаем множество дизъюнктов

$$S=\{\overline{A}\vee\overline{B}\vee C,\overline{C}\vee\overline{D}\vee E,F\vee D,F\vee\overline{E},A,B,\overline{F}\}$$

Построим резолютивный вывод из S, заканчивающийся 0:

1. $\mathrm{res}_A\{\overline{A} \vee \overline{B} \vee C, A\} = \overline{B} \vee C$;
2. $\mathrm{res}_B\{\overline{B} \vee C, B\} = C$;
3. $\mathrm{res}_D\{\overline{C} \vee \overline{D} \vee E, F \vee D\} = \overline{C} \vee E \vee F$;
4. $\mathrm{res}_E\{\overline{C} \vee E \vee F, F \vee \overline{E}\} = \overline{C} \vee F$;
5. $\mathrm{res}_C\{C, \overline{C} \vee F\} = F$;
6. $\mathrm{res}\{F, \overline{F}\} = 0$.

Итак, заключаем, что формула $A \to (B \to F)$ выводима из множества формул $A \to (B \to C)$, $CD \to E$, $\neg F \to D \& (\neg) E$.

Отметим, что метод резолюций достаточен для обнаружения возможной выполнимости данного множества дизъюнктов S. Для этого включим в множество S все дизъюнкты, получающиеся при резолютивных выводах из S. Из теоремы о полноте метода резолюций вытекает

Следствие 5.5.1. Если множество дизъюнктов S содержит резольвенты всех своих элементов, то S выполнимо тогда и только тогда, когда $0 \notin S$.

Глава VI. Элементы теории алгоритмов.

§6.1. Определение алгоритма.

Характерной чертой современности является широкое использование компьютеров при решении проблем (задач) в различных областях человеческой деятельности. Однако, предварительно задача должна быть решена алгоритмически, т.е. должно быть задано формальное предписание, следуя которому можно получить итоговый результат для решения всех задач определенного типа (это интуитивное, не строгое понятие алгоритма). Например, алгоритм нахождения наибольшего общего делителя двух натуральных чисел **a, b**, выглядит следующим образом :

1) разложить число **a** на простые множители;
2) повторить п. 1 для **b** *и* перейти к п. 3;
3) составить произведение общих простых множителей из разложений **a** и **b** с показателями, равными наименьшим из показателей вхождения в разложения.

Проанализировав этот пример, отметим важнейшие черты (свойства) алгоритма:

1. Массовость — применимость алгоритма не к одной задаче, а к классу задач.

2. Дискретность — четкая разбивка на отдельные этапы (шаги) алгоритма.

3. Детерминированность — такая организация этапов выполнения, при которой всегда ясно как осуществить переход от одного этапа к другому.

4. Конечность — для получения результата при применении алгоритма к решению конкретной задачи выполняется конечная последовательность шагов алгоритма:

Отметим, что если наличие алгоритма само по себе служит доказательством существования алгоритма, то для доказательства его отсутствия необходимо иметь строгое математическое определение алгоритма.

Попытки формализовать понятие алгоритма привели к созданию **машины Тьюринга**, как некоего воображаемого устройства, реализующего алгоритм. Ещё одним шагом в определении понятия алгоритма стало появление **рекурсивных функций,** как функций, формализующих понятие алгоритма и реализующих интуитивное понятие вычислимости. Вскоре было установлено, что множество рекурсивных функций совпадает с множеством функций, вычислимых на машинах Тьюринга. Появлявшиеся затем новые понятия, претендующие на объяснение понятия алгоритма, оказывались эквивалентными функциям, вычислимым на машинах Тьюринга, а значит, и рекурсивным функциям. Итогом развернувшейся дискуссии о том, что такое алгоритм, стало утверждение, называемое сейчас тезисом Чёрча.

Тезис Чёрча. Понятие алгоритма, или вычислимости некоторым механическим устройством, совпадает с понятием вычислимости на машинах Тьюринга (а значит, с понятием рекурсивной функции). Другими словами, алгоритм – это процесс, который может быть представлен функциональной схемой и реализован некоторой машиной Тьюринга.

Это утверждение нельзя считать математической теоремой. Это есть некоторый естественнонаучный тезис, принятый большинством исследователей.

§6.2. Машина Тьюринга.

Машина Тьюринга представляет собой (абстрактное) устройство, состоящее из ленты, управляющего устройства и считывающей головки.

Лента разбита на ячейки. Во всякой ячейке в точности один символ из **внешнего алфавита** $A=\{a_0,a_1,\ldots,a_n\}$. Некоторый символ (будем обозначать его Λ) алфавита A называется пустым, а любая ячейка, содержащая в данный момент пустой символ, называется пустой ячейкой (в этот момент). Лента предполагается потенциально неограниченной в обе стороны.

Управляющее устройство в каждый момент времени находится в некотором состоянии q_j, принадлежащем множеству $\mathbf{Q=\{q_0, q_1,..., q_m\}}$ (m=l). Множество Q называется **внутренним алфавитом**. Одно из таких состояний (обычно $\mathbf{q_0}$) называется заключительным, а некоторое другое (обычно $\mathbf{q_1}$) -начальным.

Считывающая головка перемещается вдоль ленты так, что в каждый момент времени она обозревает ровно одну ячейку ленты. Головка может считывать содержимое обозреваемой ячейки и записывать в нее вместо обозреваемого символа некоторый новый символ из внешнего алфавита **A** (возможно тот же самый).

В процессе работы управляющее устройство в зависимости от состояния, в котором оно находится и символа, обозреваемого головкой, изменяет свое внутреннее состояние **q**. Затем выдает головке приказ напечатать в обозреваемой ячейке определенный символ из внешнего алфавита **A,** а потом приказывает головке либо остаться на месте, либо сдвинуться на одну ячейку вправо, либо сдвинуться на одну ячейку влево. Попав в заключительное состояние, машина прекращает работу.

Конфигурацией на ленте (или машинным словом) называется совокупность, образованная:

1) последовательностью $\mathbf{a_{i(1)}, a_{i(2)},..., a_{i(s)}}$ символов из внешнего алфавита **A**, записанных в ячейках ленты, где $\mathbf{a_{i(1)}}$.-символ записанный в первой ячейке слева и т.д. (любая такая последовательность называется **словом),**

2) состоянием внутренней памяти q_r;

3) номером **k** воспринимаемой ячейки.

Конфигурацию машины будем записывать так:

$$a_{i(1)}, a_{i(2)},..., a_{i(r-1)} \frac{a_{i(r)}}{q_r} a_{i(r+1)}, a_{i(r+2)},..., a_{i(s)}$$

здесь **r**-воспринимаемая ячейка, выделяется в виде дроби.

Если машина, находясь во внутреннем состоянии $\mathbf{q_i}$, воспринимает ячейку с символом $\mathbf{a_u}$, записывает к следующему моменту времени в эту ячейку символ $\mathbf{a_r}$, переходит во внутреннее состояние $\mathbf{q_s}$ и сдвигается по ленте, то говорят, что машина выполняет команду $\mathbf{q_i a_u \rightarrow q_s a_r S}$, где символ S может принять одно из значений: -1 – сдвиг головки влево; +1 - сдвиг головки вправо; 0 – головка остается на месте. Список всех команд (пятерок), определяющих работу машины Тьюринга, называется **программой** этой машины. Программу машины часто задают в виде таблицы. Так для описанной выше ситуации в таблице на пересечении строки $\mathbf{a_u}$ и столбца $\mathbf{q_i}$ будет стоять $\mathbf{q_s a_r S}$ (см. таб.6.2.1)

Таб.6.2.1.

	q_0	…	q_i	…	q_m
…					
a_u			$q_s a_r S$		
…					

Если в программе машины для пары (q_i, a_u) пятерка отсутствует, то в таблице на пересечении строки a_u, и столбца q_i ставится прочерк.

Итак, **машина Тьюринга есть, по определению**, набор **M=(A,Q,П)**, где **A** — внешний алфавит, **Q** — алфавит внутренних состояний, **П** — программа.

Если машина, начав работу с некоторым словом P, записанным на ленте, придет в заключительное состояние, то она называется **применимой к этому слову**. Результатом ее работы считается слово, записанное на ленте в заключительном состоянии. В противном случае, машина называется не применимой к слову P.

Пример 6.2.1. Построим машину Тьюринга, складывающую натуральные числа, записанные в унарной системе счисления (т.е. записанные при помощи одного символа **|**. Например 5=|||||.).

Решение. Рассмотрим алфавит $A = \{|, +, \wedge\}$.
Машина определяется следующей программой:

	q_1	q_2	q_3
|	$\|\, q_1 + 1$	$\wedge\, q_3 - 1$	$\|\, q_3 - 1$
+	$\|\, q_1 + 1$		
$\wedge$	$\wedge\, q_2 - 1$		$\wedge\, q_0 + 1$

Выпишем последовательно возникающие при работе этой машины конфигурации на исходном слове ||+|||, т.е. 2+3. Здесь при записи конфигурации будем использовать следующее соглашение: состояние, в котором находится машина, записывается в круглых скобках справа за обозреваемой буквой.

0) $\cdots\wedge|(q_1)|+|||\wedge\ldots$
1) $\cdots\wedge||(q_1)+|||\wedge\ldots$
2) $\cdots\wedge||+(q_1)|||\wedge\ldots$
3) $\cdots\wedge||||(q_1)||\wedge\ldots$
4) $\cdots\wedge|||||(q_1)|\wedge\ldots$
5) $\cdots\wedge||||||(q_1)\wedge\ldots$
6) $\cdots\wedge||||||\wedge(q_1)\ldots$
7) $\cdots\wedge||||||(q_2)\wedge\ldots$
8) $\cdots\wedge||||||(q_3)\wedge\ldots$
9) $\cdots\wedge||||(q_3)|\wedge\ldots$
10) $\cdots\wedge|||(q_3)||\wedge\ldots$
11) $\cdots\wedge||(q_3)|||\wedge\ldots$
12) $\cdots\wedge|(q_3)||||\wedge\ldots$
13) $\cdots\wedge(q_3)|||||\wedge\ldots$
14) $\cdots\wedge|(q_0)||||\wedge\ldots$

◄

Пример 6.2.2. Построить машину Тьюринга, удваивающую натуральные числа, записанные в унарной системе счисления.

Решение. Искомую машину построим в алфавите A={|, α, ∧}. Программа такой машины может выглядеть так:

	q_1	q_2	q_3
|	$\alpha\, q_1 + 1$	$\| q_2 - 1$	$\| q_3 + 1$
α		$\| q_3 + 1$	
∧	$\wedge\, q_2 - 1$	$\wedge\, q_0 + 1$	$\| q_2 - 1$

Применим полученную машину к слову ||.

0) $\cdots\wedge|(q_1)|+\ldots$ 1) $\cdots\wedge\alpha|(q_1)\wedge\ldots$ 2) $\cdots\wedge\alpha\alpha\wedge(q_1)\ldots$

3) $\cdots\wedge\alpha\alpha(q_2)\wedge\ldots$ 4) $\cdots\wedge\alpha|\wedge(q_3)\ldots$ 5) $\cdots\wedge\alpha|(q_2)|\wedge\ldots$

6) $\cdots\wedge\alpha|(q_2)|\wedge\ldots$ 7) $\cdots\wedge\alpha(q_2)||\wedge\ldots$ 8) $\cdots\wedge|(q_3)||\wedge\ldots$

9) $\cdots\wedge|(q_2)|||\wedge\ldots$ 13) $\cdots\wedge|(q_2)|||\wedge\ldots$ 14) $\cdots\wedge(q_2)||||\wedge\ldots$

15) $\cdots\wedge|(q_0)|||\wedge\ldots$ ◀

Введение новой буквы α и замена исходных | на α позволяет различить исходные | и новые (приписанные) |. Состояние $\mathbf{q_1}$ обеспечивает замену | на α, состояние $\mathbf{q_2}$ обеспечивает поиск α, предназначенных для замены на |, и останов машины в случае, когда α не обнаружено, $\mathbf{q_3}$ обеспечивает дописывание | в случае, когда произошла замена α на |.

§6.3. Рекурсивные функции

Договоримся, что в этом параграфе

1. множество N натуральных чисел содержит 0 (нуль), т.е. N={0,1,2,3,...};
2. рассматриваемые функции $f=f(x_1,x_2,\ldots,x_n)$ определены только тогда, когда переменные принимают значения из N, т.е. $x_i\in N$;
3. область значений функций $D\subseteq N$;
4. рассматриваемые функции $f=f(x_1,x_2,\ldots,x_n)$ могут быть частично определенные, т.е. определенные не для всех наборов натуральных чисел.

Введём в рассмотрение **простейшие функции**

o(x)=0, s(x)=x+1, $I_n^m(x_1,...,x_n)=x_m$

Эти функции могут быть вычислены с помощью соответствующего механического устройства (например, на машине Тьюринга). Определим операторы, которые по одной или нескольким заданным функциям строят новые функции.

Оператор суперпозиции. Пусть даны функция $f(x_1,x_2,\ldots,x_k)$ от k переменных и k функций $f_1(x_1,x_2,\ldots,x_n),\ldots,f_k(x_1,x_2,\ldots,x_n)$ от n переменных. Суперпозицией функций $f,f_1,\ldots,f_k$ называется функция

$\varphi(x_1,x_2,\ldots,x_n)= f(f_1(x_1,x_2,\ldots,x_n),\ldots,f_k(x_1,x_2,\ldots,x_n))$

Мы говорим, что функция φ получается применением оператора суперпозиции S^{k+1} к функциям $f,f_1,\ldots,f_k$, и пишем $\varphi=S^{k+1}(f,f_1,\ldots,f_k)$ Например, $S^2(s,o)=s(o(x))$, т.е. функция, тождественно равная 1, а $S^2(s,s)=s(s(x))$ – это функция $y(x)=x+2$.

Оператор примитивной рекурсии. Пусть даны функции $g(x_1,x_2,\ldots,x_n)$ и $h(x_1,x_2,\ldots,x_{n+2})$. Построим функцию $f(x_1,x_2,\ldots,x_{n+1})$ Пусть зафиксированы значения $x_1,x_2,\ldots,x_n$. Тогда полагаем:

$f(x_1,x_2,\ldots,x_n,0)= g(x_1,x_2,\ldots,x_n)$

$f_1(x_1,x_2,\ldots,x_n,y+1)= h(x_1,x_2,\ldots,x_n,y, f(x_1,x_2,\ldots,x_n,y))$

Эти равенства определяют функцию $f(x_1,x_2,\ldots,x_{n+1})$ однозначно. Функция f называется функцией, полученной с помощью оператора R примитивной рекурсии. Используется запись f=R(g,h).

Индуктивное определение функции (продемонстрированное в операторе примитивной рекурсии) в математике не редкость. Например, индуктивно определяются 1) степень с натуральным показателем: $a^0=1$, $a^{n+1}=a^n\cdot a$; 2) факториал: 0!=1, (n+1)!= n!·(n+1) и т.д.

Определение. Функции, которые могут быть получены из простейших $o(x)=0$, $s(x)=x+1$, $I_n^m(x_1,...,x_n)=x_m$ применением конечного числа раз операторов суперпозиции и примитивной рекурсии, называются **примитивно рекурсивными.**

Проверим, что функция $u(x,y)=x+y$ является примитивно рекурсивной. Действительно, мы имеем: $u(x,0)=0$, $u(x,y+1)=x+y+1=u(x,y)+1$. Это есть схема примитивной рекурсии, так как $x=I_1^1(x)$, а $u(x,y)+1=s(u(x,y))=S^2(s,u)$. Здесь $g(x)=I_1^1(x)$, а $h(x,y,u)=s(u)=S^2(s,I_3^3)$.

Аналогично доказывается, что функции $m(x,y)=x\cdot y$, $d(x,y)=x^y$ (считаем по определению $0^0=1$), $fact(x)=x!$ и многие другие являются примитивно рекурсивными.

Отметим; что примитивно рекурсивные функции всюду определены (т.е. определены для всех значений их аргументов). Действительно, простейшие функции **o, s,** I_n^m являются всюду определёнными, а применение операторов суперпозиции и

примитивной рекурсии ко всюду определённым функциям даёт также всюду определённые функции. Значит, такая функция, как

$$f(x,y)=\begin{cases} x-y, & \text{если } x \geq y \\ \text{не существует, если } x < y \end{cases}$$

примитивно рекурсивной быть не может. Рассматривать функцию f(x,y)=x-y здесь мы не имеем права, так как значения функций должны быть натуральными числами (поэтому не отрицательными). Однако, можно рассмотреть функцию

$$x \div y=\begin{cases} x-y & \text{если } x \geq y \\ 0 & \text{если } x < y. \end{cases}$$

Проверим, что она примитивно рекурсивна. Докажем вначале, что функция $\varphi(x)=x\div 1$ примитивно рекурсивна. Действительно, $\varphi(0)=0$. $\varphi(y+1)=(y+1)\div 1=y$, что служит схемой примитивной рекурсии для функции $x\div 1$. Наконец, $x\div 0=x$, $x\div(y+1)=(x\div y)\div 1=\varphi(x\div y)$ – схема примитивной рекурсии для $x\div y$.

Существенно более широким классом функций, чем примитивно рекурсивные функции, является класс рекурсивных функций (определение см. ниже). В литературе эти функции называют также **частично рекурсивными.** Для их определения введём ещё один оператор.

Оператор минимизации. Пусть дана функция $f(x_1,x_2,\dots,x_n,x_{n+1})$. Зафиксируем какие-либо значения $x_1,x_2,\dots,x_n$ первых n переменных и будем вычислять $f(x_1,x_2,\dots,x_n,0)$, $f(x_1,x_2,\dots,x_n,1)$, $f(x_1,x_2,\dots,x_n,2)$ и т.д. Если y – наименьшее натуральное число, для которого $f(x_1,x_2,\dots,x_n,y)=x_{n+1}$ (т.е. значения $f(x_1,x_2,\dots,x_n,0)$, $f(x_1,x_2,\dots,x_n,1),\dots,f(x_1,x_2,\dots,x_n,y-1)$ все существуют и не равны x_{n+1}), то полагаем $g(x_1,x_2,\dots,x_n,x_{n+1})=y$. Таким образом,

$$g(x_1,x_2,\dots,x_n,x_{n+1})=\min(y|f(x_1,x_2,\dots,x_n,y)=x_{n+1}).$$

Если такого **y** нет, то считаем, что $f(x_1,x_2,\dots,x_n,x_{n+1})$ не определено. Итак, возможны три случая:

1. $f(x_1,x_2,\dots,x_n,0)$, $f(x_1,x_2,\dots,x_n,1),\dots,f(x_1,x_2,\dots,x_n,y-1)$ существуют и не равны x_{n+1}, а $f(x_1,x_2,\dots,x_n,y)=x_{n+1}$;
2. $f(x_1,x_2,\dots,x_n,0)$, $f(x_1,x_2,\dots,x_n,1),\dots,f(x_1,x_2,\dots,x_n,y-1)$ существуют и не равны x_{n+1}, а $f(x_1,x_2,\dots,x_n,y)$ не существует;
3. $f(x_1,x_2,\dots,x_n,i)$ существуют при всех $i\in N$ и отличны от x_{n+1}.

Если имеет место 1-й случай, то $g(x_1,x_2,\dots,x_n,x_{n+1})=y$, а если 2-й или 3-й, то $g(x_1,x_2,\dots,x_n,x_{n+1})$ не определено. Про функцию g полученную таким образом, говорят, что она получена из f применением оператора минимизации **M**. Мы пишем g= Mf.

Оператор минимизации – это очевидное обобщение оператора взятия обратной функции. Обобщение довольно глубокое, так как от

функции f не требуется, чтобы она была взаимно однозначной (по переменной x_{n+1})

Определение. Функции, которые могут быть получены из простейших **o(x)=0, s(x)=x+1,** $\mathbf{I_n^m(x_1,..., x_n) = x_m}$ применением конечного числа раз операторов суперпозиции, примитивной рекурсии и минимизации, называются **рекурсивными.**

Класс рекурсивных функций шире класса примитивно рекурсивных функций хотя бы по тому, что он содержит не только всюду определённые функции. Докажем, например, что функция

$$f(x,y)=\begin{cases} x-y, & \text{если } x \ge y \\ \text{не существует, если } x < y \end{cases}$$

является рекурсивной. Действительно, f(x,y)=min(z| y+z=x), а ранее было установлено, что функция u(x,y)=x+y примитивно рекурсивна.

Рекурсивные функции отражают наше интуитивное представление о функциях, вычислимых некоторым механическим устройством. В частности, они вычислимы на машинах Тьюринга (см. предыдущий параграф). Наоборот, всякая функция, вычислимая на машине Тьюринга является рекурсивной. Мы не будем проверять этот факт, так как это потребовало бы слишком много времени и места. Полное доказательство можно найти, например, в книге А.И.Мальцева "Алгоритмы и рекурсивные функции".

Отметим, что не всякая функция натуральных аргументов является рекурсивной, даже не всякая функция одного аргумента. Существование нерекурсивных функций и является "математической причиной" наличия алгоритмически неразрешимых задач.

§6.4. Алгоритмически неразрешимые задачи.

В разных разделах математики встречаются алгоритмически неразрешимые задачи, т.е. задачи, для которых нет алгоритма решения, причём нет не потому что его пока не придумали, а потому что он невозможен в принципе. Разумеется, алгоритм надо понимать в смысле машин Тьюринга и рекурсивных функций. Сформулируем одну из этих задач

Проблема остановки машины Тьюринга. Машина Тьюринга – это объект, определяемый конечным числом параметров. Все частичные отображения одного конечного множества в другое могут быть эффективным образом перенумерованы. Поэтому каждой машине Тьюринга можно присвоить номер (натуральное число). Пусть T(n) машина Тьюринга с номером n. Некоторые машины, начинающие работать на пустой ленте, в конце концов останавливаются, а некоторые работают бесконечно долго. Возникает задача: по натуральному числу n определить, остановится или нет машина

Тьюринга T(n) запущенная на пустой ленте. Эта задача алгоритмически неразрешима. То есть не существует автоматической процедуры, для каждого n решающей, останавливается или нет машина T(n). Это не исключает того, что для какой-либо конкретной машины мы установим, останавливается она или нет. Не существует метода, решающего это сразу для всех машин.

§6.5. Алгоритмы и их сложности.

Если дана задача, как найти для ее решения эффективный алгоритм? А если алгоритм найден, как сравнить его с другими алгоритмами, решающими ту же задачу? Как оценить его качество? Вопросы такого рода интересуют и программистов, и тех, кто занимается теоретическим исследованием вычислений.

Для оценки алгоритмов существует много критериев. Чаще всего нас будет интересовать порядок роста необходимых для решения задачи времени и емкости памяти при увеличении входных данных. Нам хотелось бы связать с каждой конкретной задачей некоторое число, называемое ее размером, которое выражало бы меру количества входных данных. Например, размером задачи умножения матриц может быть наибольший размер матриц−сомножителей.

Время, затрачиваемое алгоритмом, как функция размера задачи, называется **временной сложностью** этого алгоритма. Поведение этой сложности в пределе при увеличении размера задачи называется **асимптотической временной сложностью**. Аналогично можно определить **емкостную сложность** и асимптотическую емкостную сложность.

Именно асимптотическая сложность алгоритма определяет в итоге размер задач, которые можно решить этим алгоритмом. Если алгоритм обрабатывает входы размера n за время $c \cdot n^2$, где c — некоторая постоянная, то говорят, что временная сложность этого алгоритма есть $O(n^2)$ (читается "порядка эн квадрат").

Можно было бы подумать, что колоссальный рост скорости вычислений, вызванный появлением нынешнего поколения цифровых вычислительных машин, уменьшит значение эффективных алгоритмов. Однако происходит противоположное. Так как вычислительные машины работают все быстрее и быстрее, и мы можем решать большие по размеру задачи, именно сложность алгоритма определяет то увеличение размера задачи, которое можно достичь с увеличением скорости машины.

Допустим, у нас есть пять алгоритмов A1,A2,…,A5 со следующими временными сложностями:

Алгоритм	Временная сложность
A1	n

A2	$n \cdot \log(n)$
A3	n^2
A4	n^3
A5	2^n

Здесь временная сложность — это число единиц времени, требуемого для обработки входа размера n. Пусть единицей времени будет одна миллисекунда (1сек=1000 милисекунд). Тогда алгоритм A1 может обработать за одну секунду вход размера 1000, в то время как A5 — вход размера не более 9. В таб. 6.5.1. приведены размеры задач, которые можно решить за одну секунду, одну минуту и один час каждым из этих пяти алгоритмов.

Таблица 6.5.3.

Алгоритм	Временная сложность	Максимальный размер задачи		
		1 сек	1 мин	1 час
A1	n	1000	$6?10^4$	$3{,}6 \cdot 10^6$
A2	$n \cdot \log(n)$	140	4893	$2{,}0 \cdot 10^4$
A3	n^2	31	244	1897
A4	n^3	10	39	153
A5	2^n	9	15	21

Предположим, что следующее поколение вычислительных машин будет в 10 раз быстрее нынешнего. В таб.6.5.2. показано, как возрастут размеры задач, которые мы сможем решить благодаря этому увеличению скорости. Заметим, что для алгоритма A5 десятикратное увеличение скорости увеличивает размер задачи, которую можно решить, только на три единицы (см. последнюю строку в таб.6.5.2.), тогда как для алгоритма A3 размер задачи более чем утраивается.

Таблица 6.5.4.

Алгоритм	Временная сложность	Максимальный размер задачи	
A1	n	S1	10 S1.
A2	$n \cdot \log(n)$	S2	Примерно 10 S2 для больших S2.

A3	n^2	S3	3,1б5 S3
A4	n^3	S4	2,155 S4.
A5	2^n	S5	S5+3,3

Вместо эффекта увеличения скорости рассмотрим теперь эффект применения более действенного алгоритма. Вернемся к таб.6.5.1. Если в качестве основы для сравнения взять 1 мин, то, заменяя алгоритм A4 алгоритмом A3, можно решить задачу, в 6 раз большую, а заменяя A4 на A2, можно решить задачу, большую в 125 раз. Эти результаты производят гораздо большее впечатление, чем двукратное улучшение, достигаемое за счет десятикратного увеличения скорости. Если в качестве основы для сравнения взять 1 ч, то различие оказывается еще значительнее. Отсюда мы заключаем, что асимптотическая сложность алгоритма служит важной мерой качественности алгоритма, причем такой мерой, которая обещает стать еще важнее при последующем увеличении скорости вычислений.

Несмотря на то что основное внимание здесь уделяется порядку роста величин, надо понимать, что большой порядок роста сложности алгоритма может иметь меньшую мультипликативную постоянную (постоянная **c** в определении O(f(x))), чем малый порядок роста сложности другого алгоритма. В таком случае алгоритм с быстро растущей сложностью может оказаться предпочтительнее для задач с малым размером — возможно, даже для всех задач, которые нас интересуют. Например, предположим, что временные сложности алгоритмов A1, A2, A3, A4, A5 в действительности равны соответственно $1000n$, $100n \cdot \log(n)$, $10n^2$, n^3 и 2^n. Тогда A5 будет наилучшим для задач размера $2 \le n \le 9$, A2 — для задач размера $10 \le n \le 58$, A1 — при $59 \le n \le 1024$, а *A1*— при $n > 1024$.-

ЛИТЕРАТУРА.

1. Ф.А.Новиков. Дискретная математика для программистов./ Санкт-Петербург: Питер, 2001г., 304С.
2. С.В.Судоплатов, Е.В.Овчинникова. Элементы Дискретной математики./ М., ИНФРА-М, Новосибирск, Изд-во НГТУ, 2002г., 208С.
3. Я.М.Ерусалимский. Дискретная математика/ М., «Вузовская книга», 2001г.,279С.
4. А.Ахо, Дж.Хопкрофт, Дж.Ульман. Построение и анализ вычислительных алгоритмов. / М., Мир, 1979г., 536С.
5. В.Н.Нефедов, В.А.Осипова Курс дискретной математики./ М., Изд-во МАИ, 1992г., 264С.