# Позиционная алгебра логики. Почему P=NP.

**Тарасов В.А.**

**(статья в стадии разработки)**

Прежде чем начать изложение основ позиционной алгебры, я сделаю небольшое отступление в теории сложности. Эта теория была разработана двумя лауреатами премии Тьюринга: С.Куком и Р.Карпом [1,2].

Основная проблема теории сложности занимается следующим вопросом. Можно ли исключить перебор при решении дискретных задач? Иначе говоря, речь идёт о принципиальной возможности найти нужное решение не перебирая всех или почти всех вариантов в задаче. Эта проблема имеет не только чисто математическое, но и глубокое познавательное значение. Прикладная сторона этой проблемы такова: в переборных задачах, как правило, имеется конечное множество вариантов, среди которых нужно найти решение. Например, двоичных векторов размерности n имеется $2^n$ и, перебрав это экспоненциальное множество векторов, мы можем найти те вектора, которые удовлетворяют заданному свойству. Но с ростом n число векторов быстро растёт (экспоненциально), и задача становится "труднорешаемой", то есть практически неразрешимой. Стало общепринятым считать переборную задачу решаемой эффективно, если имеется алгоритм, решающий ее за время, ограниченное полиномом от размерности задачи.

Таким образом, в указанной выше проблеме главными объектами теории являются: класс NP всех переборных задач и класс P переборных задач, решаемых за полиномиальное время.

В 1971 году С.Кук показал в своей основополагающей работе [1], что проблема выполнимости полна в классе NP относительно полиномиальной редукции, или короче, NP полна. Грубо говоря, NP-полные проблемы имеют максимальную трудность среди всех проблем перебора. Отсюда следует, что если проблема выполнимости легка, то любая проблема перебора легка (то есть принадлежит к классу P). Из более точной формулировки теоремы Кука вытекает даже более сильное утверждение, а именно такое: быстрый алгоритм для решения проблемы выполнимости вполне механическим способом приводил бы к быстрой разрешающей процедуре для любой эффективно заданной проблемы перебора. Такой алгоритм служил бы отмычкой к проблемам перебора из всех областей математики.

В 1972 году Р.Карп значительно расширил список NP-полных проблем [2]. К настоящему моменту большинство естественно возникающих проблем перебора классифицированы либо как P, либо как NP-полные.

До настоящего времени вопрос о том, действительно ли NP-полные задачи труднорешаемы, считался одним из основных открытых вопросов современной математики и теоретической кибернетики. Вопреки готовности большинства специалистов считать, что все NP-полные задачи труднорешаемы, прогресс как в доказательстве., так и в опровержении этого далеко идущего предположения был весьма незначителен. Однако, несмотря на отсутствие доказательства того, что из NP-полноты следует труднорешаемость, NP-полнота задачи означает, что для её решения полиномиальным алгоритмом требуется по крайней мере крупное открытие [3].

Это открытие было сделано Мироном Ивановичем Тельпизом [4] в конце 2000 года. В основу этого открытия положен принцип позиционности, который был использован для создания позиционной алгебры логики.

Основное положение принципа позиционности формулируется следующим образом.

Позиционное представление с основанием (или по основанию) b определяется правилом:

$(\ldots a_k \ldots a_3 a_2 a_1 a_0, a_{-1} a_{-2} \ldots a_{-m} \ldots)_b = \ldots + a_k * b^k + \ldots + a_3 * b^3 + a_2 * b^2 + a_1 * b^1 + a_0 + a_{-2} * b^{-2} + \ldots + a_{-m} b^{-m} + \ldots, (1)$

где каждый символ $a_i$, получает значение, определяемое: 1) его начертанием, 2) его положением в записи числа. Наша традиционная десятичная система счисления — это, разумеется, тот частный случай, когда b равно десяти, и когда значения $a_i$ выбираются из десятичных цифр 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. В общем случае в качестве b берут любое целое число, большее единицы и числа $a_i$ — это целые числа из интервала: $0 \le a_i \le b$ . Так, например, получаются стандартные двоичная (b=2) или десятичная (b=10) системы счисления. Выбор основания является принципиально произвольным.

Я приведу небольшой пример, из которого станет ясно, зачем понадобился принцип позиционности. Известны два способа представления чисел: символьный и позиционный. О позиционном я уже сказал. Примером символьного – может служить римская система счисления. В ней значение цифр в числе определяется исключительно их начертанием и не зависит от их места расположения. Если мы рассмотрим задачу умножения двух чисел в позиционной и символьной системах счисления, например XXI*VII и 21*7, то получим интересный результат. Сложность задачи в римской системе счисления будет экспоненциальная, а в позиционной – квадратичная. Этот факт означает, что сложность задачи зависит от языка, на котором она сформулирована.

После осмысливания этого факта сразу возникает вопрос: «Можно ли распространить принцип позиционности на задачи математической логики?». Оказывается, что можно. Это сделал М.И.Тельпиз. Работы в этом направлении были начаты более 25-ти лет назад. В настоящий момент создана стройная теория: позиционная алгебра логики (ПАЛ). Написан ряд статей, программ и, что самое важное, написана монография.

## Основы позиционной алгебры логики

### Простые позиционные операторы

Позиционная операция – это такая операция, смысл которой зависит от положения символа операции в операторе. Оператор представляет из себя цепочку из символов операций и является словом в позиционном языке представления ФАЛ

Позиционный оператор – это набор позиционных операций, применяемых к некоторому набору аргументов. Введение совокупности операций в виде позиционного оператора позволяет изучать такие операторы независимо от наборов аргументов, а это упрощает исследование свойств ФАЛ.

Обобщим бинарные операции ФАЛ с помощью позиционной n-арной операции, которую будем называть юнкцией и для которой введем знак операции: символ <:> - знак юнкции. Эта операция вводится на двоичном наборе $X_n$. Набор $X_n$ определяется следующим образом: вектор $(x_1, x_2, \ldots, x_n)$ координаты которого принимают значения из множества $E \subset (0,1)$, называется двоичным вектором или набором и обозначается через $X_n$. Число n называется длиной вектора. На наборе $X_n$ будем рассматривать n+1 позицию, нулевая позиция предшествует первому элементу, i-я позиция – это промежуток между i-м и (i+1)-м элементами, n-я позиция следует за последним элементом. В любой из этих позиций может быть указан или опущен знак юнкции. Таким образом мы опреде-

лим набор $S_{n+1}$, состоящий из знаков операций. Если в какой либо позиции указан знак юнкции, то на это место в наборе будем записывать 1, если опущен – 0. Счёт элементов в наборе $S_{n+1}$ начинается с нуля.

По определению, если на наборе $X_n$ знак юнкции указан в позициях $k_1$, $k_2, ..., k_s$, то этим описано высказывание: набор $X_n$ содержит $k_1$ или $k_2$, … , или $k_s$ единиц. Если это высказывание истинно, то значение юнкции на этом наборе равно 1, если ложно, то значение юнкции равно 0. Если ни в одной позиции набора $X_n$ знака юнкции нет, то по определению будем считать, что задана юнкция, значение которой равно 0. Унарную операцию отрицания можно выразить через юнкцию так: $\bar{x} = (:x)$. Попутно отметим, что $x = (x:), (:x:) = 1, (x) = 0$. Самые внешние скобки, когда это не может привести к недоразумению, будем опускать, т.е. (:a)=:a или (:(:ab))=:(:ab). Разумеется, если нет ни одного знака операции, то скобки обязательны.

Некоторые бинарные операции алгебры логики выражаются через операции юнкции так:

$$a \wedge b = ab: \quad a \vee b = a:b:$$

$$a \oplus b = a:b \quad a \downarrow b =: ab$$

$$a \Leftrightarrow b =: ab: \quad a \perp b =: a:b$$

Если на некотором наборе введена юнкция, то это означает, что в некоторых позициях (промежутках между элементами набора) расставлены знаки юнкции. Рассмотрим наборы, состоящие только из знаков операций, указывая знаком <_> (пробел) на промежуток, где нет знака <:>, такой набор из знаков <_>, <:> и будет позиционным оператором.

Введем более строгое понятие простого позиционного оператора: оператор $S_j^n$ – это такой оператор, в котором содержится n+1 символов таких, что если заменить <:> на 1, а пробелы на 0, то получившееся в результате двоичное n-разрядное число и будет j (при этом надо учесть, что счет символов идет слева на право, начиная с нулевой позиции). Действие простого позиционного оператора на набор данных формально определяется следующим образом:

$$S_j^n(X_n) = s_i \quad где \quad i = \sum_{r=1}^{n} x_r \quad S_j^n = (s_0, s_1, ..., s_n)$$

Любая из операций $(\wedge, \vee, \oplus, \perp, \Leftrightarrow, \downarrow)$ может быть представлена соответствующим позиционным оператором следующим образом:

$$\wedge = (001) = S_4^2 \qquad \vee = (011) = S_6^2$$

$$\oplus = (010) = S_2^2 \qquad \perp = (110) = S_3^2$$

$$\Leftrightarrow = (101) = S_5^2 \qquad \downarrow = (100) = S_1^2$$

Из определения n-арной юнкции и простого позиционного оператора видно, что простой позиционный оператор позволяет представить любую симметрическую ФАЛ.

Если мы используем один и тот же алфавит для представления данных и операторов, то мы можем выполнять не только операторы над данными, но и операторы над операторами. Это одно из фундаментальных отличий ПАЛ от ФАЛ. В теори функций алгебры логики невозможно выполнить одну операцию над другой. В ФАЛ такие действия совершенно бессмысленны, они не определены.

Оператор над операторами выполняется следующим образом.

Прежде всего, отметим, что размерность оператора применяемого к операторам должна быть согласована. Это значит, что должно выполняться условие $S_j^n(S_{1,...,}S_n)$. Иначе говоря, число позиций в строке аргументов должно быть равно длине применяемого к ним оператора, а эта длина равна n+1, так как счёт позиций начинается с 0.

Операторы, являющиеся аргументами образуют матрицу следующего вида

$$
\begin{array}{l}
\quad S_i^k \quad S_j^l \quad \bullet \quad S_r^m \\
\text{———————} \\
\quad s_0 \quad s_0 \quad \bullet \quad s_0 \\
\quad s_1 \quad \bullet \quad \bullet \quad s_1 \\
(\,S^k{}_i,\, S^l{}_j,..,\, S^m{}_r) = \quad \bullet \quad s_l \quad \bullet \quad \bullet \qquad (0.1) \\
\quad s_k \quad \bullet \quad \bullet \quad \bullet \\
\quad \bullet \quad \bullet \quad \bullet \quad s_m \\
\text{———————} \\
\quad a \quad b \quad \bullet \quad c
\end{array}
$$

где в верхней строке указаны операторы, а под ними соответствующие им наборы. Сами операторы в эту матрицу не входят.

На следующем шаге суммируем вертикально по разрядам двоичные числа, представляющие операторы, их сумма показана в формуле под матрицей, символы a, b,…, c.

Каждый разряд результирующего оператора выбирается из набора применяемого оператора (двоичного числа). Номером разряда является значение поразрядной суммы полученной на предыдущем шаге.

При этом проверяется второе правило. Если полученная поразрядная сумма или номер разряда результирующего оператора больше его длинны, то эта операция не определена. Иначе говоря, она выходит из области определения.

Результат этой операции будет выглядеть так:

$$S_j^n(S_1, S_2, ..., S_n) = S_k^n \quad где \quad k = a \cdot 2^0 + b \cdot 2^1 + ... + c \cdot 2^n$$

Продемонстрируем на примерах выполнение операторов $S^2{}_6$, $S^2{}_1$, $S^2{}_3$ над операторами $S^2{}_5$ и $S^2{}_4$.

Вычислим поразрядную сумму:

$$
\begin{array}{l}
S_5^2 = 101 \\
S_4^2 = 001 \\
\text{—————} \\
\sum = 102
\end{array}
$$

Теперь, имея поразрядную сумму $\Sigma$, по правилу применения простого оператора к набору данных вычисляем значения разрядов результирующих операторов. Учитывая, что все вычисления проводятся в каждом вертикальном разрядном срезе. Результирующими операторами будут:

$S^2{}_6(\,S^2{}_5,\, S^2{}_4) = (101) = S^2{}_5$

$S^2{}_1(\,S^2{}_5,\, S^2{}_4) = (010) = S^2{}_2$

$S^2{}_3(\,S^2{}_5,\, S^2{}_4) = (110) = S^2{}_3$

Более существенный эффект можно получить, переходя от бинарных операций к операциям большей арности. Например, легко проверить, что $S^3_3(S^4_9, S^4_{12}, S^4_{28}) = S^4_{19}$, $S^3_5(S^4_9, S^4_{12}, S^4_{28}) = S^4_6$, $S^3_{12}(S^4_9, S^4_{12}, S^4_{28}) = S^4_{12}$.

Однако тот факт, что с помощью простого позиционного оператора могут быть представлены только симметрические ФАЛ, сильно ограничивает его полезность. Для того, чтобы расширить класс функций, представляемых позиционными операторами, вводится другие позиционные операторы $R, W, \sigma, \Omega$ и т.д., они описаны в работах [5,6,7,8,9,10]. Каждый из этих операторов описывает определенный тип ФАЛ. Например, с помощью $\sigma$ операторов описываются арифметические операции, сложения и умножения двоичных чисел.

В следующем разделе будут рассмотрены фундаментальные позиционные FS-операторы, которые описывают всё множество ФАЛ на заданном наборе $X_n$.

**Фундаментальные позиционные FS-операторы**

Для работы с FS-операторами нам прежде всего необходимо ввести понятие совершенного набора. Для набора $X_n=(x_1, x_2, \ldots, x_n)$ совершенным называется такой двоичный набор $X_n^s=(x_1,x_2,x_2,x_3,x_3,x_3,x_3,\ldots,x_n,x_n,\ldots, x_n)$, в котором при i=1,2, …,n аргумент $x_i$ содержится $2^{i-1}$ раз, т.е. набор $X_n^s$ имеет длину $N=2^n-1$. Следовательно, простой оператор, согласованный с совершенным набором $X_n^s$ , должен содержать N+1 символов. Такие операторы и называются FS-операторами.

При записи FS операторов мы будем пользоваться следующим правилом: если $a \subset E(0,1)$, то $a^k=a\ldots aaa\ldots.a$, где в правой части a входит не k раз, а $2^k$ раза. Кроме того, фигурные скобки {} означают, что выражение заключенное в них будет повторено столько раз, сколько это необходимо для согласования FS оператора с набором $X_n^s$. Например, если n=3, то

$\{0^1 1^1\}=00110011$.

При изучении рекурсивных функций используются числовые функции проектирования, которые определяются следующим образом: для $1 \le i \le n$, $pr_i(X_n)=x_i$. Числовые функции проектирования нами будут использованы с дополнительным требованием $X_n$ – двоичный набор. Если значение логического выражения E совпадает со значением позиционного FS оператора R на наборе $X_n^s$, т.е. $E=R(X_n^s)$, то будем говорить, что выражение E имеет оператор R.

В этом случае для координат набора $X_n$ имеет место соотношение $pr_i(X_n)=\{0^{i-1}1^{i-1}\}(X_n^s)$, $1 \le i \le n$. Это соотношение и является эквивалентом функции проектирования, мы будем называть это соотношение селекторной функцией и обозначать символом s.

Используя селекторные функции можно построить FS-операторы для классических операций алгебры логики, которые приведены ниже (во всех представлениях i и k удовлетворяют условиям: $1 \le i \le n, 1 \le k \le n$):

$$x_i = \{0^{i-1}1^{i-1}\}$$

$$\neg x_i = \{1^{i-1}0^{i-1}\}$$

$$x_i \wedge x_{i+k} = \{0^{i+k-1}(0^{i-1}1^{i-1})\}$$

$$x_i \vee x_{i+k} = \{(0^{i-1}1^{i-1})1^{i+k-1}\}$$

## Представление FS-операторов с помощью системы взаимосвязанных алфавитов

Рассмотрим упорядоченную систему алфавитов, такую, что каждый следующий алфавит будет состоять из конкатенации символов предыдущего алфавита, причём длина его будет вдвое больше длины символа принадлежащего предыдущему алфавиту. Под длиной символа мы будем понимать количество двоичных разрядов, необходимых для представления этого символа. Например, длина символа из алфавита $a_0$ равна 1, а длина символа алфавита $a_1$ равна 2.

Для обозначения алфавита будем использовать заглавную букву $A_k$, где нижний индекс обозначает порядок алфавита. Таким образом $A_0$=(0,1), $A_1$=(00,01,10,11), $A_2$=(0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111). В каждом алфавите длина символа равна $2^k$, а число символов входящих в алфавит равно 2 в степени $2^k$.

Введём обозначение $0_k$ для символов состоящих только из нулей, а для символов состоящих только из единиц – $1_k$, где k имеет тот же смысл, что и при обозначении алфавита. Тогда селекторные функции можно выразить следующим образом:

$s_1=\{0_0 1_0\}$, $s_2=\{0_1 1_1\}$, $s_i=\{0_{i-1} 1_{i-1}\}$.

Как видно из определения системы алфавитов, каждый символ порядка k состоит из двух символов порядка k-1. Рассмотрим на примере, применение введенных определений и понятий к вычислению выполнимости формулы $F=(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \neg x_3)$, то есть решим такой вопрос: «существует ли такой набор в $X_3$, при подстановке которого F=1».

Запишем селекторные функции для $x_1$, $x_2$, $x_3$:

$x_1= s_1=\{0_0 1_0\}$, $x_2= s_2=\{0_1 1_1\}$, $x_3= s_3=\{0_2 1_2\}$.

Затем вычислим значение дизъюнктов: (число перед селекторной функцией указывает количество повторений)

1) $4s_1=4\{0_0 1_0\}$
$2s_2=2\{0_1 1_1\}$
$s_3= \{0_2 1_2\}$
---
$s_1 1_1 1_2$

2) $4s_1=4\{0_0 1_0\}$
$2s_2=2\{0_1 1_1\}$
$\neg s_3= \{1_2 0_2\}$
---
$1_2 s_1 1_1$

После этого вычислим конъюнкцию операторов:

$s_1 1_1 1_2$
$1_2 s_1 1_1$
---
$s_1 1_1 s_1 1_1$

Этот результирующий оператор и будет решением поставленной задачи. Кроме того, что по нему сразу видно, F выполнима, он дает все выполняющие наборы. Если этот оператор развернуть, получим $s_1 1_1 s_1 1_1$=01110111. Эта строка является набором представляющим собой таблицу истинности для формулы F.

## Позиционная запись FS операторов

Прежде всего, заметим, что запись оператора $s_1 1_1 s_1 1_1$ или $2(s_1 1_1)$ не является позиционной. Несмотря на то, что для построения FS оператора использованы понятия позиционности, в такой его записи значение символа не определяется его позицией, а определяется его индексом и коэффициентом повторения.

Для введения позиционности записи FS операторов определим следующие понятия:

1. в позиционной записи будем использовать строчные буквы;
2. селекторные функции s и $\neg s$ будем обозначать символами S и $\overline{S}$ соответственно, в крайней левой позиции S имеет значение S=(01), в следующей S=(0011), в i-ой слева – $S=(0^{i-1}1^{i-1})$, (ранее мы определили, что $a^k$=aaa…a, где a повторяется $2^k$ раз);
3. введём символ E, который будем использовать для обозначения единичного вектора длиной $2^i$, где i номер позиции, в которой находится символ E, значение символа E в крайней левой позиции равно E=(11), в следующей E=(1111) и т.д.;
4. введем пустой символ O, который будет использован для обозначения пустых позиций в позиционной записи FS операторов, фактически, этот символ будет выполнять роль нуля; при записи операторов;
5. введем следующие рекурсивные определения:

$$Z_kM = Z_kZ_k, Z_kW = Z_kZ_k^*, Z_kT = Z_kE_k, Z_k\overline{T} = Z_k\overline{E}_k,$$

$$Z_k\overline{M} = Z_k\overline{Z}_k, Z_k\overline{W} = Z_k\overline{Z}_k^*, Z_kF = E_kZ_k, Z_k\overline{F} = \overline{E}Z_k,$$

$$O_kS = \overline{E}_kE_k, O_k\overline{S} = E_k\overline{E}_k$$

запись вида $Z_k^*$ означает перезапись набора в обратном порядке, то есть $Z_k^* = (z_k, z_{k-1},..., z_1)$. Эта операция называется обращением.

Смысл этих формул станет ясен из дальнейшего изложения.

Рассмотрим на примере, позиционный способ записи FS-оператора для формулы $f=(x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (x_1 \vee x_4) \wedge (x_2 \vee x_3) \wedge (x_2 \vee x_4) \wedge (x_3 \vee x_4)$. Сначала построим таблицу истинности для этой формулы.

Таблица 4.1

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $(x_1 \vee x_2)$ | $(x_1 \vee x_3)$ | $(x_1 \vee x_4)$ | $(x_2 \vee x_3)$ | $(x_2 \vee x_4)$ | $(x_3 \vee x_4)$ | f |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Затем, используя введенные нами определения, запишем позиционные операторы для выражений:

1. $(x_1 \vee x_2) - STMM$
2. $(x_1 \vee x_3) - SMTM$
3. $(x_1 \vee x_4) - SMMT$
4. $(x_2 \vee x_3) - OSTM$
5. $(x_2 \vee x_4) - OSMT$
6 $(x_3 \vee x_4) - OOST$

Подобная запись называется продукцией.

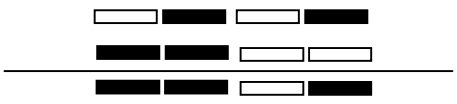
Теперь мы можем сравнить таблицы истинности, полученные классическим и операторным способом. Развернём позиционные операторы, используя введенные нами определения:
(символьная запись после знака равно "=" не является позиционной)

1. STMM=SEMM=SESEM=SESESESE=(0111011101110111)
2. SMTM=SSTM=SSEEM=SSEESSEE=(0101111101011111)
3. SMMT=SSMT=SSSST=SSSSEEEE=(0101010111111111)
4. OSTM=$\overline{\text{E}}$ETM=$\overline{\text{E}}$EEEM=$\overline{\text{E}}$EEE$\overline{\text{E}}$EEE=(0011111100111111)
5. OSMT=$\overline{\text{E}}$EMT=$\overline{\text{E}}$E$\overline{\text{E}}$ET=$\overline{\text{E}}$E$\overline{\text{E}}$EEEEE=(0011001111111111)
6. OOST=O$\overline{\text{E}}$ET=$\overline{\text{E}}$ $\overline{\text{E}}$EET=$\overline{\text{E}}$ $\overline{\text{E}}$EEEEEE=(0000111111111111)

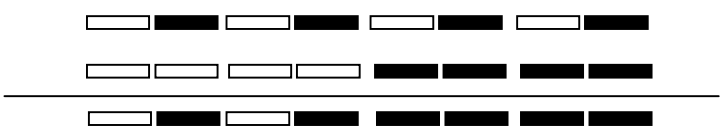
Для выполнения операций конъюнкции нам потребуется ввести ряд дополнительных понятий и определений. Но прежде, более детально, поясним смыл введенных рекурсивных определений, используя графические представления.

В ПАЛ в любом операторе всегда должны присутствовать селекторные функции для всех переменных, потому, что его запись позиционно зависима. Иначе говоря, должны быть символы показывающие входит ли данная переменная в соответствующую оператору булеву формулу или нет. Например, для формулы $x_1 \vee x_2 = ST$, почему это так, легко понять, если представить себе таблицу истинности. Изобразим это графически (светлые прямоугольники изображают нули, темные – единицы)

Как видно из рисунка, в результате дизъюнкции $x_1$ и $x_2$ мы приписываем единичный вектор к селекторной функции $S_1$ справа. Для формулы $x_1 \vee \overline{x_2} = SF$ картинка изменится следующим образом



То есть мы приписываем единичный вектор слева. В том случае, когда мы имеем функцию от трех переменных, например такую $x_1 \vee x_3 = SMT$, её изображение будет иметь следующий вид



Как видно из картинки, селекторная функция $S_1$ повторяется дважды в результирующем векторе. В этом и есть смысл символа M, он означает, что предшествующий ему вектор будет повторен. В таблице 4.2 приведен полный перечень используемых символов и соответствующие им операции над таблицей истинности.

Таблица 4.2

| | |
|---|---|
| T | Приписывает единичный вектор справа |
| $\overline{\mathrm{T}}$ | Приписывает нулевой вектор справа |
| F | Приписывает единичный вектор слева |
| $\overline{\mathrm{F}}$ | Приписывает нулевой вектор слева |
| M | Дублирует вектора |
| $\overline{\mathrm{M}}$ | Приписывает инвертированный вектор справа |
| W | Приписывает обращенный вектор справа |
| $\overline{\mathrm{W}}$ | Приписывает обращенный и инвертированный вектор слева |

Отметим, что операции T и F являются обратными.

**Юнкты и их операторы**

Для двоичной переменной $x_i$ и её отрицания $\overline{x}_i$ будем использовать общее наименование – литерал, а дизъюнкцию различных литералов – дизъюнктом. Наряду с дизъюнктом будем использовать понятие **юнкт**. Это позволит нам определить новые значения, записывая соответствующие знаки операций $(\wedge, \vee, \Leftrightarrow, \oplus, \downarrow, \perp)$ и через чёрточку – слово юнкт. Например: $\wedge$-юнкт, $\vee$-юнкт, $\oplus$-юнкт и т.д. Под всеми этими юнктами будем понимать операцию над литералами. Однолитеральные юнкты будут называться единичными юнктами. Пустые юнкты (юнкты без литералов) рассматриваться не будут. Юнкт, содержащий r-литералов, называется r-литеральным. Селекторные функции – это единичные юнкты.

При построении FS-операторов различных юнктов, важным является понятие области определения для рассматриваемого юнкта. Будем говорить, что область определения юнкта есть $X_m$, если индексы каждого из литералов не больше m и изучаемый юнкт рассматривается именно в этой области.

Если функция алгебры логики *f(x)* имеет FS-оператор *Z,* то следующие ниже функции с областью определения $X_{m+1}$ имеют FS-операторы:

$$f(X_{m+1}) = f(X_m) \rightleftarrows ZM$$
$$f(X_{m+1}) \vee x_{m+1} \rightleftarrows ZT$$
$$f(X_{m+1}) \vee \overline{x}_{m+1} \rightleftarrows ZF$$
$$f(X_{m+1}) \wedge x_{m+1} \rightleftarrows Z\overline{F}$$
$$f(X_{m+1}) \wedge \overline{x}_{m+1} \rightleftarrows Z\overline{T}$$
$$f(X_{m+1}) \oplus x_{m+1} \rightleftarrows Z\overline{M}$$
$$f(X_{m+1}) \oplus \overline{x}_{m+1} \rightleftarrows Z\overline{M}$$

Эти утверждения позволяют легко строить FS-операторы различных юнктов, а знание таких операторов и применение логических операций позволяет строить более сложные FS-операторы.

Например, пусть n=10, тогда

$$x_3 \vee x_4 \vee \overline{x}_6 \vee x_9 \vee \overline{x}_{10} = OOSTMFMMTF$$

$$\overline{x}_2 \vee x_5 \vee \overline{x}_7 \vee x_8 = O\overline{S}MMTMFTMM$$

$$x_3 \wedge x_4 \wedge \overline{x}_6 \wedge x_9 \wedge \overline{x}_{10} = OO\overline{F}M\overline{T}MM\overline{F}\overline{T}$$

$$\overline{x}_2 \vee x_5 \vee \overline{x}_7 \vee x_8 = O\overline{S}MM\overline{F}M\overline{T}\overline{F}MM$$

**Задача выполнимости и её FS-оператор**

Задача выполнимости (сокращенно ВЫП) заключается в распознавании выполнимости конъюнктивной нормальной формы (КНФ). Форма называется КНФ, если она является конъюнкцией одного или более конъюнктивных членов, каждый из которых является дизъюнктом.

Примеры:

Задача 7.1. Определить выполнима ли КНФ

$$f(X_7) = (x_1 \vee x_3 \vee x_4 \vee \overline{x}_7) \wedge (\overline{x}_1 \vee x_3 \vee \overline{x}_6) \wedge (\overline{x}_4 \vee x_5 \vee \overline{x}_6 \vee x_7) \wedge (x_2 \vee \overline{x}_3 \vee \overline{x}_5 \vee x_7)$$

Задача 7.2 Определить выполнима ли КНФ

$$f(X_7) = (\overline{x}_2 \vee \overline{x}_4 \vee x_5) \wedge (\overline{x}_2 \vee x_3 \vee x_5) \wedge (x_1 \vee \overline{x}_3 \vee x_4 \vee \overline{x}_5) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_4 \vee \overline{x}_5) \wedge$$

$$\wedge (x_2 \vee x_3 \vee \overline{x}_4 \vee \overline{x}_5) \wedge (x_1 \vee \overline{x}_3 \vee \overline{x}_5) \wedge (\overline{x}_1 \vee \overline{x}_3)$$

Нас интересуют задачи этого типа по той причине, что они являются классическим вариантом NP-полных задач. В 1971 году С.Кук показал в своей основополагающей работе [5], что задача выполнимости полна в классе NP, относительно полиномиальной редукции, или короче, NP полна. Грубо говоря, NP-полные задачи имеют максимальную сложность среди всех задач перебора. Отсюда следует, что задача выполнимости легка, то любая проблема перебора легка (то есть принадлежит к классу P). Из более точной формулировки теоремы Кука вытекает даже более сильное утверждение, а именно такое: быстрый алгоритм для решения проблемы выполнимости механически приводил бы к быстрой решающей процедуре для любой эффективно заданной проблеме перебора. Такой алгоритм служил бы отмычкой к проблемам перебора из всех областей математики.

В работе [6] приведены различные формы записи задачи выполнимости (ВЫП). Мы так же введем для задачи ВЫП запись, более удобную для нас. На основе выше сделанных определений мы можем ввести более удобно для нас формулировку задачи. Возьмём задачу (7.2) и преобразуем её к следующему виду:

Пусть заданы 7 $\vee$-юнктов на 5-ти литералах

$$z_1 = (\overline{x}_2 \vee \overline{x}_4 \vee x_5), z_2 = (\overline{x}_2 \vee x_3 \vee x_5), z_3 = (x_1 \vee \overline{x}_3 \vee x_4 \vee \overline{x}_5), z_4 = (\overline{x}_1 \vee \overline{x}_2 \vee x_4 \vee \overline{x}_5)$$

$$z_5 = (x_2 \vee x_3 \vee \overline{x}_4 \vee \overline{x}_5), z_6 = (x_1 \vee \overline{x}_3 \vee \overline{x}_5), z_7 = (\overline{x}_1 \vee \overline{x}_3)$$

Требуется распознать, выполнима ли КНФ

$$f(X_7) \rightleftarrows \wedge_{i=1}^{7} z_i$$

В такой формулировке мы можем перейти к описанию метода распознавания в задаче выполнимости, построенном на базе средств позиционной алгебры логики.

**Методы распознавания в задаче выполнимости**

Введем для задачи (7.2) некоторые обозначения и понятия, пригодные в общем случае. Запишем FS-операторы $\vee$-юнктов и сведем эти FS-операторы в таблицу 8.1, в которой номера строк соответствуют индексам $z_i$, а номера столбцов – это индексы литералов.

$$z_1 = (\bar{x}_2 \vee \bar{x}_4 \vee x_5) \rightleftarrows O\bar{S}MFT, z_2 = (\bar{x}_2 \vee x_3 \vee x_5) \rightleftarrows O\bar{S}TMT$$
$$z_3 = (x_1 \vee \bar{x}_3 \vee x_4 \vee \bar{x}_5) \rightleftarrows SMFTF, z_4 = (\bar{x}_1 \vee \bar{x}_2 \vee x_4 \vee \bar{x}_5) \rightleftarrows \bar{S}FMTF \quad (8.1)$$
$$z_5 = (x_2 \vee x_3 \vee \bar{x}_4 \vee \bar{x}_5) \rightleftarrows OSTFF, z_6 = (x_1 \vee \bar{x}_3 \vee \bar{x}_5) \rightleftarrows SMFMF$$
$$z_7 = (\bar{x}_1 \vee \bar{x}_3) \rightleftarrows \bar{S}MFMM$$

Таблица 8.1

| *i/j* | *1* | *2* | *3* | *4* | *5* |
|---|---|---|---|---|---|
| *1* | *O* | $\bar{S}$ | *M* | *F* | *T* |
| *2* | *O* | $\bar{S}$ | *T* | *M* | *T* |
| *3* | *S* | *M* | *F* | *T* | *F* |
| *4* | $\bar{S}$ | *F* | *M* | *T* | *F* |
| *5* | *O* | *S* | *T* | *F* | *F* |
| *6* | *S* | *M* | *F* | *M* | *F* |
| *7* | $\bar{S}$ | *M* | *F* | *M* | *M* |

Совершенно ясно, что перестановка строк местами не влияет на выполнимость КНФ (конъюнктивные сомножители коммутативны).

Перестановка двух столбцов таблицы ведет к переименованию переменных. Следовательно и такая операция не влияет на выполнимость (хотя эта операция изменяет выполняющий набор с точностью до переименования). Преобразование таблицы начинается, именно с перестановки столбцов. Перестановка столбцов в таблице должна осуществляться так, чтобы суммарное число литер *M* в столбце не возрастало слева направо. Иными словами и точнее можно сказать, что переименование литералов должно осуществляться так, чтобы индекс литерала не убывал, если не убывает его вхождение в КНФ. При этом необходимо обязательно учесть, что символ *O* при перемещении внутрь таблицы превращается в *M*, и наоборот. Символ S при перемещении внутрь таблицы превращается в T, а символ $\bar{S}$ - в символ F, и соответственно наоборот. Операцию перестановки столбцов по указанному критерию будем называть приведением, а саму таблицу – приведенной.

Результат приведения таблицы 8.1 приведён в таблице 8.2, в первой строке которой приведены старые номера столбцов.

Таблица 8.2

| | *1* | *2* | *4* | *3* | *5* |
|---|---|---|---|---|---|
| *i/j* | *1* | *2* | *3* | *4* | *5* |
| *1* | *O* | $\overline{S}$ | *F* | *M* | *T* |
| *2* | *O* | $\overline{S}$ | *M* | *T* | *T* |
| *3* | *S* | *M* | *T* | *F* | *F* |
| *4* | $\overline{S}$ | *F* | *T* | *M* | *F* |
| *5* | *O* | *S* | *F* | *T* | *F* |
| *6* | *S* | *M* | *M* | *F* | *F* |
| *7* | $\overline{S}$ | *M* | *M* | *F* | *M* |

Операция нормализации – это некоторая последовательность операций перестановки строк в таблице и её подтаблицах, с целью упорядочивания строк по некоторому признаку, задаваемому выбранным столбцом или его частью. Этот столбец будем называть ведущим.

Выделение строк по ведущему столбцу практически означает такой анализ таблицы, с помощью которого можно ответить: следует ли разбивать данную таблицу на две подтаблицы, какие это должны быть подтаблицы и что с ними делать дальше. Ведущим столбцом, с которого начинается анализ таблицы, является самый правый столбец. Затем становится ведущим следующий слева столбец, но рассматривается только та его часть, которая принадлежит анализируемой подтаблице, и т.д.

Задачей анализа ведущего столбца является определение одной из его характеристик. Каждая характеристика определяет дальнейший ход анализа в алгоритме просмотра таблицы. Проверка характеристик должна осуществляться в указанной ниже последовательности до первого ответа «да» на вопрос, содержится ли в ведущем столбце:

1. хотя бы по одной клетке с $S$ и $\overline{S}$ ;
2. хотя бы одна клетка с $S$ и имеются клетки с $F$ или $M$, либо $F$ и $M$ одновременно;
3. хотя бы одна клетка с $\overline{S}$ и имеются клетки с $T$ или $M$, либо $T$ и $M$ одновременно;
4. хотя бы одна клетка с $S$, а в остальных – $T$;
5. хотя бы одна клетка с $\overline{S}$ , а в остальных – $F$;
6. во всех клетках $M$;
7. во всех клетках $T$ или $M$;
8. во всех клетках $F$ или $M$;
9. во всех клетках $T$ или $F$;
10. во всех клетках $T$, $F$ или $M$;

В работе [Mine] доказана теорема. Её формулировка выглядит следующим образом. Пусть рассматривается некоторая подтаблица (совпадение её со всей таблицей не исключается) с ведущим столбцом, номер которого $k+1$, где $1 \le k \le n-1$. Пусть этой таблице соответствует FS-оператор $\langle Y_{k+1} \rangle$. Тогда значение $Y_{k+1}$ зависит от номера ответа так:

$1) Y_{k+1} = \overline{E}_{k+1}$ $2) Y_{k+1} = \overline{E}_k \nu_k$

$3) Y_{k+1} = \mu_k \overline{E}_k$ $4) Y_{k+1} = \eta_k E_k$

$5) Y_{k+1} = E_k \delta_k$ $6) Y_{k+1} = \varepsilon_k \varepsilon_k$

$7) Y_{k+1} = \mu_k \varepsilon_k$ $8) Y_{k+1} = \varepsilon_k \nu_k$

$9) Y_{k+1} = \eta_k \delta_k$ $10) Y_{k+1} = \mu_k \nu_k$

где

$$\langle \nu_k \rangle = \langle \varepsilon_k \rangle \wedge \langle \delta_k \rangle \quad \langle \mu_k \rangle = \langle \varepsilon_k \rangle \wedge \langle \eta_k \rangle$$

$$\langle \varepsilon_k \rangle = \wedge_{j \in M(k+1)} \langle A_k^{(j)} \rangle \quad \langle \eta_k \rangle = \wedge_{j \in T(k+1)} \langle A_k^{(j)} \rangle \quad \langle \delta_k \rangle = \wedge_{j \in F(k+1)} \langle A_k^{(j)} \rangle \qquad (8.2)$$

Результат этой теоремы схематически показан на рис.1, где оператор $\langle Y_{k+1} \rangle$ представлен в виде композиции двух символов из алфавита размерности равной k.

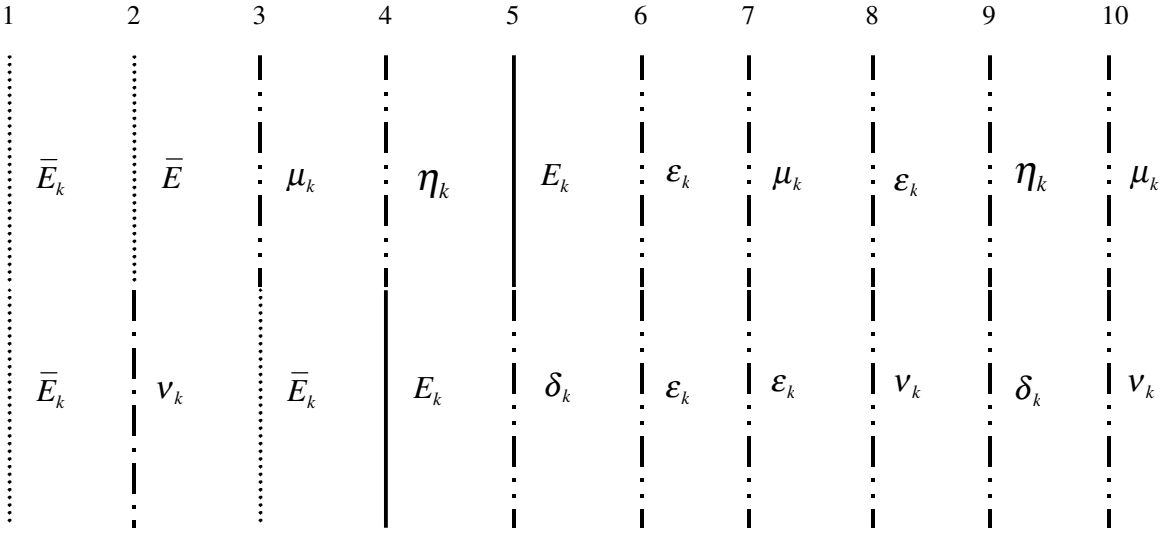



Рис. 1

Если в битовом представлении символа все биты равны 0, то на схеме это изображено пунктиром (отметим, что $0_k$ - это $2^k$ нулей), если же все – единичные, то сплошной линией ($1_k$ - это $2^k$ единиц). Штрих пунктирные линии означают, что битовую структуру необходимо выяснить, т.е. для них необходимо совершить такой же шаг, который был сделан для анализа $Y_{k+1}$. Но в этом случае верно более сильное утверждение.

Ветвление (разбиение таблицы на две подтаблицы) должно осуществляться лишь при 9-ом и 10-ом исходе. Причём при 9-ом исходе эти подтаблицы не пересекаются, т.е. не имеют общих строчек, а при 10-ом исходе подтаблицы пересекаются и общие строчки определяются номерами, составляющими множество $M_{k+1}$.

Одновременно отметим, что в случае 6 обе половины схемы определяются одной и той же подтаблицей. Задача выполнимости (ВЫП) сформулирована таким образом, что требуется ответить на вопрос, совпадает ли FS-оператор за-

дачи с $\langle O_n \rangle$. Вариант задачи ВЫП в котором необходимо представить решающие наборы называется задачей выполнимости с предъявлением. Ниже будет показано как построить все решающие наборы, если будет определено, что FS-оператор задачи не равен нулю.

В 7-ом и 8-ом случаях достаточно анализировать лишь ту подтаблицу, которая определяется оператором $\langle \varepsilon_k \rangle$. Если он равен $\langle 0_n \rangle$, то поскольку $\langle \nu_k \rangle = \langle \varepsilon_k \rangle \wedge \langle \delta_k \rangle$ и $\langle \mu_k \rangle = \langle \varepsilon_k \rangle \wedge \langle \eta_k \rangle$ следовательно $\langle \nu_k \rangle = \langle \mu_k \rangle = \langle 0_k \rangle$. В том случае, когда $\langle \varepsilon_k \rangle$ не равно $\langle 0_n \rangle$, это означает, что КНФ, представленная таблицей выполнима.

Если в k-ом ведущем столбце исход с номером 1 вызван строками $\langle A_k \rangle = \langle S_k \rangle$ и $\langle B_k \rangle = \langle \overline{S}_k \rangle$, то в столбцах с номером k и k+1 можно выполнить одно из следующих преобразований:

$$\langle S_k T \rangle \wedge \langle \overline{S}_k T \rangle = \langle S_{k+1} \rangle$$
$$\langle S_k F \rangle \wedge \langle \overline{S}_k F \rangle = \langle \overline{S}_{k+1} \rangle$$
$$\langle S_k M \rangle \wedge \langle \overline{S}_k M \rangle = \langle O_{k+1} M \rangle$$

$$\langle S_k T \rangle \wedge \langle \overline{S}_k M \rangle = \langle S_{k+1} \rangle \wedge \langle \overline{S}_k M \rangle$$
$$\langle S_k F \rangle \wedge \langle \overline{S}_k M \rangle = \langle \overline{S}_{k+1} \rangle \wedge \langle \overline{S}_k M \rangle$$

Это позволит нам сворачивать оператор.

Схемы рис.1 показывают, что переход от одного ведущего столбца к другому (движение выполняется справа налево) заканчивается лишь исходами с номерами 1, 4 и 5. В тех случаях, когда анализ заканчивается исходами 4 и 5, мы получаем решение задачи с ответом: КНФ, представленная таблицей, выполнима. В тех случаях, когда анализ заканчивается исходом 1, получается, что некоторый символ, составляющий часть FS-оператора КНФ, равен $\overline{E}_k$, где k – номер ведущего столбца. Это означает, что таблицу КНФ можно упростить. После этого необходимо повторить анализ оставшейся части таблицы. Этот процесс будет продолжаться либо до получения исходов 4 или 5, либо до полного вырождения таблицы. В последнем случае мы получаем ответ: КНФ, представленная таблицей, не выполнима.

### Примеры применения алгоритма анализа таблиц

Рассмотрим распознавание выполнимости на некоторых примерах. Анализ начнем с таблицы 9.1 которая получена из таблицы 8.1, но в ней добавлены служебные столбцы и строки.

Таблица 9.1

| | 5 | 6 | 8 | T | F | | + |
|---|---|---|---|---|---|---|---|
| | 1 | 6 | 7 | T | T | + | |
| | 1 | 2 | 4 | 3 | 5 | | |
| i/j | 1 | 2 | 3 | 4 | 5 | | |
| 1 | O | $\overline{S}$ | F | M | T | 7 | 5 |
| 2 | O | $\overline{S}$ | M | T | T | 7 | 6 |
| 3 | S | M | T | F | F | 5 | 7 |
| 4 | $\overline{S}$ | F | T | M | F | 5 | 7 |
| 5 | O | S | F | T | F | 6 | 7 |
| 6 | S | M | M | F | F | 3 | ! |
| 7 | $\overline{S}$ | M | M | F | M | 3 | 3 |
| 6 | 0 | O | O | $\overline{S}$ | F | | 7 |

Каждой добавленной строке соответствует добавленный столбец (на их пересечении указан знак +). Служебную строку будем заполнять справа на лево, указывая в ней номер исхода. Однако в случае исхода 9 и 10, исходов с ветвлением, будем записывать T или F, т.е. будем указывать на то, что строки с этими символами в ведущем столбце будут исключены. В соответствующем служебном столбце против строк указывается номер ведущего столбца, вызвавший исключение этой строки. Если строка заменяется новой, то знаком «!» указывается на то, что эта строка вычёркивается, а под её номером записывается новая.

Распишем по шагам алгоритм анализа таблиц, после номера исхода будет записана последовательность выполняемых действий.

1. Возвращаемся назад (слева направо) до первого столбца с номером исхода 9 или 10 и выполняем анализ второй полвины таблицы.
2. Отбрасываем ту часть подтаблицы, для которой $\langle Y_k \rangle = \langle O_k \rangle$ и применяем алгоритм анализа к той части подтаблицы, для которой $\langle Y_k \rangle = \langle \nu_k \rangle$.
3. Отбрасываем ту часть подтаблицы, для которой $\langle Y_k \rangle = \langle O_k \rangle$ и применяем алгоритм анализа к той части подтаблицы, для которой $\langle Y_k \rangle = \langle \mu_k \rangle$.
4. Выполнима, так как FS-оператор отличен от $\langle O_k \rangle$.
5. Выполнима, так как FS-оператор отличен от $\langle O_k \rangle$.
6. В этом случае две подтаблицы эквивалентны и все клетки подтаблицы содержат символ M, поэтому удалять какие-либо строки нет надобности.
7. В этом случае нам достаточно проанализировать только ту подтаблицу, которая определяется оператором $\langle \varepsilon_k \rangle$, т.е. остаются только строки с M, а строки с T отбрасываются. В том случае, когда строк с M нет, т.е. одна из подтаблиц вырождена, то такой исход эквивалентен исходу 4.
8. В этом случае нам достаточно проанализировать только ту подтаблицу, которая определяется оператором $\langle \varepsilon_k \rangle$, т.е. остаются только строки с M, а строки с F отбрасываются. В том случае, когда строк с M нет, т.е. одна из подтаблиц вырождена, то такой исход эквивалентен исходу 5.
9. В этом случае мы получаем две подтаблицы, первая определяется оператором $\langle \eta_k \rangle$, а вторая – оператором $\langle \delta_k \rangle$. Мы запоминаем точку ветвления и

выполняем анализ подтаблицы, определяемой оператором $\langle \eta_k \rangle$. Если в результате анализа получим исходы 4 или 5, то КНФ выполнима. Если получим исход 1, то необходимо вернуться в точку ветвления, подтаблице $\langle \delta_k \rangle$ и выполнить соответствующие процедуры анализа для этой подтаблицы.

10. В этом случае действия аналогичны предыдущему пункту, с той лишь разницей, что подтаблицы определяются операторами $\langle \mu_k \rangle$ и $\langle \nu_k \rangle$.

Рассмотрим на примерах применение алгоритма анализа на нескольких таблицах, соответствующих выполнимым и невыполнимым КНФ. Начнем с таблицы 9.1. Ей соответствует выполнимая КНФ. Чтобы было проще объяснять работу алгоритма, изобразим анализ таблицы с помощью графа. Вершины графа будут соответствовать ведущим столбцам, они изображены квадратами, внутри которых указаны номера исходов. Ребрам соответствуют подтаблицы, над каждым ребром указан оператор, соответствующий данной подтаблице. На рисунке 9.1 показан граф, соответствующий этой таблице.

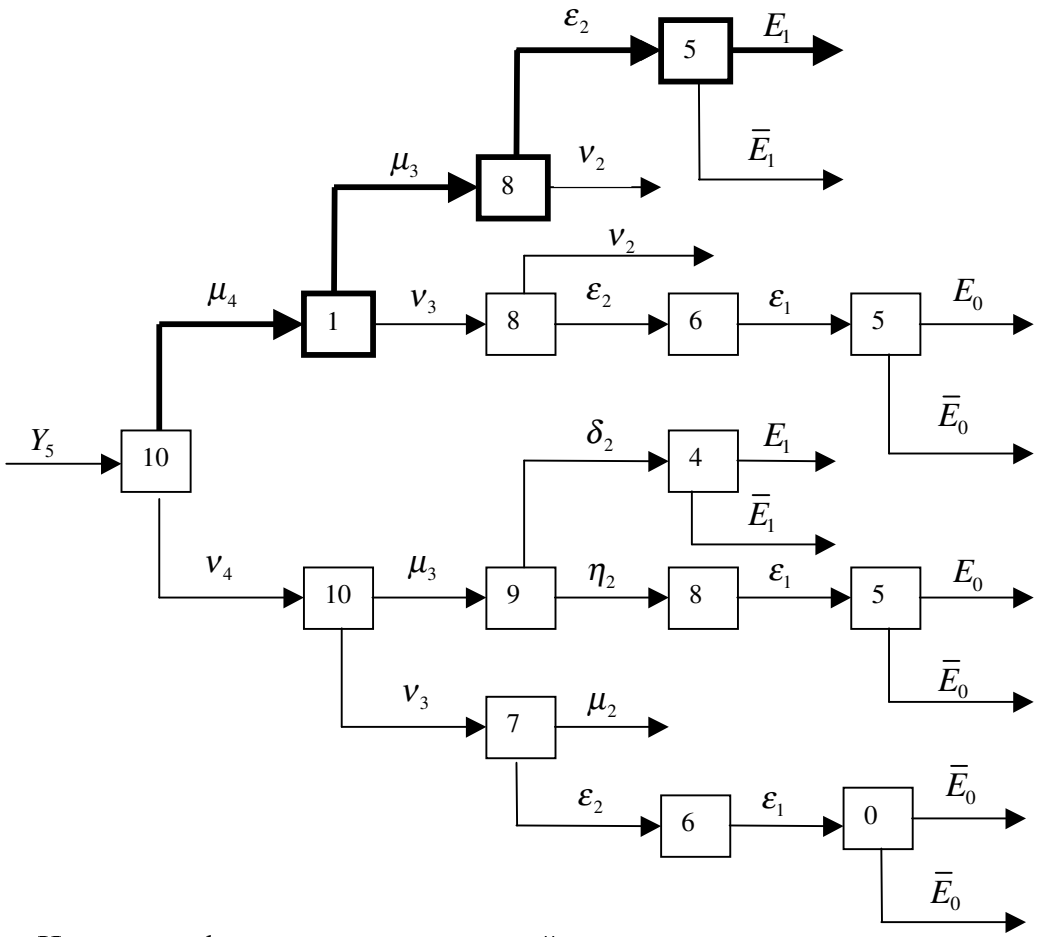



Рис 9.1

На этом графе выделен путь, который приводит к ответу о выполнимости КНФ (исход 5). Остальные пути пока демонстрации множества всех возможных путей. Оно нам понадобится в дальнейшем. Однако, если КНФ будет не выполнима, то придется развернуть весь граф и проанализировать все пути. Рассмотрим это на примере, таблице 9.2 соответствует не выполнимая КНФ.

Таблица 9.2

| | 3 | 3 | 6 | 3 | 2 | | + | |
|---|---|---|---|---|---|---|---|---|
| | | | 1 | 3 | F | + | | |
| i/j | 1 | 2 | 3 | 4 | 5 | | | |
| 1 | O | O | O | $\overline{S}$ | T | 6 | ! | ! |
| 2 | O | O | O | $\overline{S}$ | F | 7 | 6 | ! |
| 3 | S | M | M | T | F | 7 | 3 | |
| 4 | O | O | $\overline{S}$ | T | T | 5 | ! | ! |
| 5 | O | $\overline{S}$ | M | T | F | 7 | 5 | ! |
| 6 | $\overline{S}$ | T | M | T | F | 7 | 3 | ! |
| 7 | $\overline{S}$ | M | M | F | M | 5 | 7 | |
| 4 | 0 | O | O | S | T | | ! | ! |
| 1 | O | O | O | O | S | | 7 | |
| 6 | O | S | M | T | F | | | ! |
| 5 | O | O | O | S | F | | | |
| 2 | O | O | O | O | $\overline{S}$ | | | |

Невыполнимость КНФ видна из таблицы 9.2. FS-операторы новых строк с номерами1 и 2 равны $S$ и $\overline{S}$ соответственно, а их конъюнкция равна <$O$>. Граф, соответствующий этой таблице, показан на рисунке 9.2.

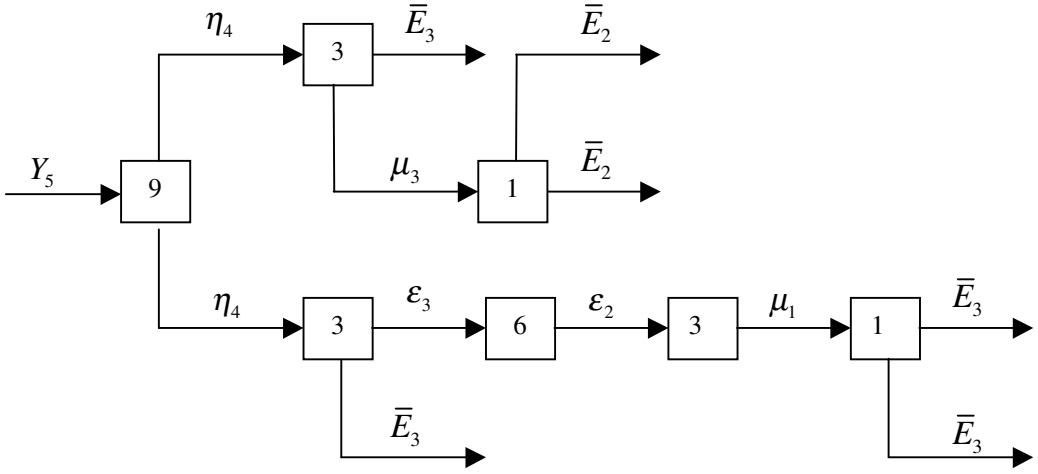



Рис 9.2

В следующем пункте я проведу оценку возможных путей в графе и влияние приведения на их количество. Пока отмечу, что приведение может значительно сократить число возможных путей.

**Анализ метода распознавания и оценка его сложности**

Метод распознавания, изложенный в предыдущем пункте, страдает двумя недостатками. Первый недостаток связан с тем, что если таблица не приведена, то число путей в графе может расти экспоненциально с ростом числа переменных ($n$). Точнее можно построить таблицу (частный пример представлен

таблицей 10.1), в которой имеется $m \geq n+1$ строк на $n$ литералах с графом, имеющим $2^{n-1}$. Граф таблицы 10.1 имеет 64 пути.

Таблица 10.1

| | *1* | *2* | *4* | *3* | *5* | *6* | *7* |
|---|---|---|---|---|---|---|---|
| *i/j* | *1* | *2* | *3* | *4* | *5* | | |
| *1* | *S* | *M* | *M* | *T* | *M* | *M* | *T* |
| *2* | *S* | *M* | *M* | *F* | *M* | *T* | *M* |
| *3* | $\overline{S}$ | *F* | *M* | *M* | *T* | *M* | *M* |
| *4* | $\overline{S}$ | *T* | *T* | *M* | *M* | *M* | *M* |
| *5* | $\overline{S}$ | *T* | *F* | *M* | *M* | *M* | *M* |
| *6* | $\overline{S}$ | *F* | *M* | *M* | *F* | *M* | *M* |
| *7* | *S* | *M* | *M* | *F* | *M* | *F* | *M* |
| *8* | *S* | *M* | *M* | *T* | *M* | *M* | *F* |

Преодоление этого недостатка может осуществляться приведением таблицы. Таблица 10.2 является результатом приведения таблицы 10.1.

Таблица 10.2

| | *1* | *2* | *4* | *3* | *5* | *6* | *7* |
|---|---|---|---|---|---|---|---|
| *i/j* | *1* | *2* | *3* | *4* | *5* | *6* | *7* |
| *1* | *S* | *M* | *M* | *M* | *T* | *M* | *T* |
| *2* | *O* | *S* | *M* | *M* | *F* | *M* | *T* |
| *3* | *O* | *O* | *S* | *M* | *M* | *F* | *F* |
| *4* | *O* | *O* | *O* | *S* | *M* | *T* | *F* |
| *5* | *O* | *O* | *O* | $\overline{S}$ | *M* | *T* | *F* |
| *6* | *O* | *O* | $\overline{S}$ | *M* | *M* | *F* | *F* |
| *7* | *O* | $\overline{S}$ | *M* | *M* | *F* | *M* | *T* |
| *8* | $\overline{S}$ | *M* | *M* | *M* | *T* | *M* | *T* |

В приведённой таблице число путей резко сокращается, в графе таблицы 10.2 их всего 4. Однако, преодоление этого недостатка однократным приведением таблицы недостаточно, так как приведенная таблица может содержать не приведенные подтаблицы. Приведение же всех подтаблиц может потребовать противоречивую перестановку столбцов. Второй недостаток связан с уже приведёнными таблицами, так как в приведенной таблице число путей может оказаться достаточно большим. Это связано с тем, что одна и та же строка, и даже подтаблица может участвовать неоднократно в формировании различных частей пути в графе.

Метод распознавания можно модифицировать. Модификация метода распознавания направлена на устранение указанных недостатков путём частичного изменения таблицы на эквивалентную ей, то есть без изменения её FS-оператора. Частичное изменение таблицы приводит к тому, что мы переходим к таблицам состоящим из юнктов, определенных в пункте 6. Такое обобщение задачи выполнимости (ВЫП) связано с тем, что в преобразованиях $\vee-$*юнктов* задачи ВЫП удобно иметь более широкий класс образований, какими являются юнкты. Иначе говоря необходимо уметь решать задачу выполнимости конъюнктивной формы (ВЫП-КФ) состоящей из юнктов, а не только из $\vee-$*юнктов*. Построение FS-операторов для юнктов выполняется по правилам, изложенным в теореме 6.1. Алгоритм преобразования таблицы задачи

ВЫП в задачу ВЫП-КФ называется «суперприведением». В результате суперприведения мы получим таблицу в которой будут исключены исходы с номером 1, а следовательно и ветвление в алгоритме распознавания. Можно сказать, что таблица станет однопроходной и сложность задачи распознования будет линейно зависеть от размерности задачи (n).

### Алгоритм суперприведения

Описание алгоритма будет вставлено сразу после выхода книги М.И.Тельпиза.

### Применение методов позиционной алгебры к решению задач верификации и дискретной оптимизации

Задача логической верификации является одной из ключевых в процессе проектирования СБИС [Norenkov] . Рассмотрим простой пример. Имеется исходная схема, например CLA сумматор и мы хотим заменить его в сумматором Линга [Ling], который работает быстрее. Классический подход проверки того, что CLA сумматор эквивалентен сумматору Линга основан на методах тестирования. Создаётся набор ключевых тестов, а затем каждый тест подается на вход моделей соответствующих сумматоров. Полученные результаты сравниваются, и если они совпадают, то можно ответить, что да, CLA сумматор эквивалентен сумматору Линга. Очевидно, что если мы рассматриваем 128-разрядный сумматор, то перебрать все возможные входные данные нет никакой возможности, их будет $2^{128}$. Поэтому используют различные методы сокращения тестовых наборов, но эти методы до настоящего времени не имеют формального определения для общего случая. Поэтому весьма вероятно, что может быть допущена ошибка. Подобная ситуация случилась с компанией Intel, которая выпустила процессор Pentium III с ошибкой в FPU.

В отличие от классического метода верификации позиционная алгебра предлагает совершенно другой подход. Тот факт, что с помощью методов ПАЛ можно для любой NP-полной задачи построить эквивалентную ей задачу полиномиальной сложности, позволяет подойти к решению задачи верификации следующим образом. Преобразуем логическое описание сумматоров CLA и Линга в эквивалентные им суперприведенные FS-операторы: FS-cla и FS-ling соответственно. А затем решим логическое уравнение FS-cla $\Leftrightarrow$ FS-ling=1. Согласно М.И.Тельпизу любая логическая операция над суперприведёнными операторами даёт суперприведенный оператор. Поэтому наша задача верификации будет иметь полиномиальную сложность и может быть решена в практически приемлемое время.

Такой подход к решению задач верификации открывает большие перспективы в процессе проектирования СБИС. Например, в тех случаях когда имеется исходная схема и разработчик решил внести в неё изменения с целью её улучшения, он может получить подтверждение того, что полученная схема эквивалентна исходной за секунды, минуты, в крайнем случае, за часы.

Задачи дискретной оптимизации решаются методами ПАЛ аналогичным образом. Учитывая то, что при проектировании СБИС применяется модульный подход, мы всегда можем построить суперприведенный FS-оператор для каждого блока, а затем объединить их и выполнить суперприведение результирующего FS-оператора.

Литература

[1] Кук С.А. Сложность процедур вывода теорем. — Киб. сб. нов. сер., вып. 12. — М.: Мир, 1975, с. 5 — 15.

[2] Карп Р.М. Сводимость комбинационных задач. — Киб. сб. нов. сер., вып. 12. — М.: Мир, 1975, с. 16 — 38.

[3] Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи: Пер. с англ. — М.: Мир, 1982, 416 с.

[4] М.И. Тельпиз . Семинар в ИКИ РАН. http://www.iki.rssi.ru/pal/seminars.htm "О принципе позиционности в логических преобразованиях." .М.: 20.09.2001

[5] Тельпиз М.И. Позиционные принципы представления функций алгебры логики. Препринт. — АН СССР. Научный совет по комплексной проблеме "Кибернетика", М.: 1984, 76 с.

[6] Тельпиз М.И. Представления функций алгебры логики. — Кибернетика. Киев: 1985, N 4, с. 37 — 40, 51.

[7] Тельпиз М.И. Позиционные операторы и преобразования в алгебре логики. Препринт. — АН СССР. Научный совет по комплексной проблеме "Кибернетика", М.: 1985, 60 с.

[8] Тельпиз М.И. Алгебра позиционных операторов и эквивалентных преобразований. Препринт. — АН СССР. Научный совет по комплексной проблеме "Кибернетика", М.: 1988, 64 с.

[9] Тельпиз М.И., Демидчик С.М., Щербанский Л.М. Принцип позиционности в трёхзначной алгебре логики. Пр. — 1436. ИКИ АН СССР, М.: 1988, 20 с.

[10] Тельпиз М.И. Позиционные фундаментальные симметрические операторы и задачи логического распознавания. Пр. — 1601. ИКИ АН СССР, М.: 1989, 72 с.