Языки описания алгоритмов

Формальный язык для описания алгоритмов был первоначально ориентирован на задачи оптимизации программ для ЭВМ, а в последствии оказался удобным и для оптимизации микропрограмм. Задачи этих классов широко распространены как в области проектирования средств вычислительной техники, так и при построении систем автоматического управления, связи и многих других.

Логические схемы алгоритмов

Реализация алгоритма есть последовательное выполнение команд ЭВМ, каждая из которых в свою очередь является последовательностью элементарных действий-микрокоманд, выполняемых за один машинный такт.

Функцию задания алгоритма русский математик Алексей Андреевич Ляпунов предложил записывать определенным способом, а именно в виде конечной строки, состоящей из символов элементарных действий A_i , где i изменяется от 1 до n (n – целое положительное число), логических функций α_p (p =1,m) и специальных символов начала и конца стрелок с индексом, где индекс также целое положительное число.

Символы $A_1,A_2,\ldots,A_n$ он предложил называть операторами, а символы $\alpha_1,\alpha_2,\ldots$- логическими условиями. Логическое условие (ЛУ) - это такая функция, которая может принимать лишь два значения – 0 или 1. Оказалось, что с помощью введенных понятий возможно формально описать любой алгоритм, используя терминологию логики. Действительно, допустим символ α_1 $(x_1=x_2)$ означает, что $\alpha_1=1$, если равенство истинно и $\alpha_1=0$ в противном случае.

Рассмотрим некоторый алгоритм как определенную последовательность операторов.

$$U = A_1\ \alpha_1{\uparrow}^1\ A_2\ \alpha_2\ {\uparrow}^2\ A_3\ \alpha_3\ {\uparrow}^3 \ldots {\downarrow}^2 A_{10}\ \alpha_{10}{\uparrow}^3 \ldots A_k$$

Будем считать, что ЛУ, за исключением α_2, равны единице. Тогда выражение можно переписать в виде

$$U = A_1\ A_2\ \alpha_2{\uparrow}^2\ A_3 \ldots {\downarrow}^2 A_{10} \ldots A_k\,,$$

так как действия стрелок трактуются следующим образом: после выполнения оператора A_1 проверяется его ЛУ α_1; если $\alpha_1=1$, то выполняется следующий оператор, т.е. A_2, а если $\alpha_1=0$, то необходимо в строке отыскать конец стрелки ${\downarrow}^1$ и выполнить стоящее справа от неё элементарное выражение. Так как все ЛУ, кроме α_2, равны 1, то это и позволяет нам перейти от первого выражения ко второму.

Определение. Логической схемой алгоритма (ЛСА) будем называть конечную строку, состоящую из символов операторов A_1, A_2, …, A_n, логических условий со стрелками $\alpha_1{\uparrow}^1$, $\alpha_2{\uparrow}^2$, … $\alpha_m{\uparrow}^m$ и концов стрелок ${\downarrow}^1$, ${\downarrow}^2$, … ${\downarrow}^m$ такую, что для каждого начала стрелки с индексом i найдётся один и только один конец стрелки с тем же индексом.

Из этого определения следует, что несколько начал стрелок могут иметь одинаковые индексы, но все они привязаны к единственному концу стрелки с этим индексом.

Каждый отдельный оператор или логическое условие назовём “элементарным выражением”, а фрагмент строки из элементарных выражений со стрелками, где для каждого индекса i имеется не более одного начала стрелки и одного конца с индексом i – “выражением”.

Из ранее приведённых рассуждений вполне понятно, что в любой ЛСА порядок выполнения операторов определяется состоянием всех ЛУ. Будем для определённости считать, что ЛУ могут изменять своё состояние, но только во время выполнения некоторого оператора.

Таким образом, описание алгоритма в общем виде в терминах ЛСА может быть представлено:

$$U = U\,(A_1,\ldots,A_n;\ p_1,\ldots,p_m)$$

здесь A_i (i = 0, n) - множество операторов,

p_j (j = (1, к)) - множество логических переменных, входящих в функции $\alpha_t(p_1,p_2,\ldots,p_m)$.

Всевозможные наборы состояний логических переменных $p_1, \ldots, p_m$ обозначим

$$\Delta_1\,,\ \Delta_2\,,\ldots,\ \Delta_m\,,\ \Delta_{m+1}\,,\ \ldots$$

Тогда процесс реализации ЛСА для этой произвольной последовательности наборов определяется следующим образом:

1. Выбираем начальный набор независимых логических переменных Δ_1.
2. Анализируем самое левое элементарное выражение ЛСА: если это ЛУ, то может быть два продолжения: при p_i =1 переходим к анализу следующего элементарного выражения, а при p_i = 0 переходим к анализу элементарного выражения, стоящего справа от конца стрелки с индексом i. При выполнении оператора совершается переход от набора Δ_1 к Δ_s , где s – индекс выполняемого оператора.

Строка операторов, полученная в результате описанных действий, является значением ЛСА для заданной выше последовательности наборов.

Пример. Дана ЛСА $U = A_0 p_2 \uparrow^1 p_3 \uparrow^2 \downarrow^1 A_2 A_3 \downarrow^2 A_4 A_5 A_K$
Рассмотрим процесс реализации ЛСА при всевозможных наборах ЛУ.

$\Delta_1 = p_2 p_3$ $\quad U_1 = A_0A_2A_3A_4A_5A_K$;
$\Delta_2 = p_2 !p_3$ $\quad U_2 = A_0A_4A_5A_K$;
$\Delta_3 = !p_2 p_3$ $\quad U_3 = A_0A_2A_3A_4A_5A_K = U_1$;
$\Delta_4 = !p_2 !p_3$ $\quad U_4 = A_0A_2A_3A_4A_5A_K = U_1$.

Матричные схемы алгоритмов

Ещё одним формальным языком для описания алгоритмов является система, представленная на языке матрицы, в которой элементы есть логические функции, связывающие операторы между собой. При этом запись зависимости порядка выполнения операторов оказывается более простой и наглядной.

$$
\mathcal{U} = \begin{array}{c} \\ A_0 \\ A_1 \\ \vdots \\ A_i \\ \vdots \\ A_n \end{array}
\begin{array}{c}
\begin{array}{ccccccc} A_1 & A_2 & \dots & A_j & \dots & A_n & A_к \end{array} \\
\begin{bmatrix}
\alpha_{01} & \alpha_{02} & \dots & \alpha_{0j} & \dots & \alpha_{0n} & \alpha_{0к} \\
\alpha_{11} & \alpha_{12} & \dots & \alpha_{1j} & \dots & \alpha_{1n} & \alpha_{1к} \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots \\
\alpha_{i1} & \alpha_{i2} & \dots & \alpha_{ij} & \dots & \alpha_{in} & \alpha_{iк} \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots \\
\alpha_{n1} & \alpha_{n2} & \dots & \alpha_{nj} & \dots & \alpha_{nn} & \alpha_{nк}
\end{bmatrix}
\end{array}
$$

где A_0, A_k – операторы начальной и конечной установки алгоритма,

$A_i \ (i = \overline{1,n})$ – операторы реализации алгоритма,

$\alpha_{ij} \ (i = \overline{0,n};\ j = \overline{1,к})$ – логические функции связи между операторами, каждая из которых может принимать три значения:

1) $\alpha_{ij} = 0$, что означает отсутствие связи между операторами A_i и A_j;
2) $\alpha_{ij} = 1$, что означает непосредственное следование оператора A_j за оператором A_i;
3) α_{ij} при некоторых наборах ЛУ равна 0, а при других – 1.

Определение. Матричной схемой алгоритма (МСА) будем называть квадратную матрицу, в которой строки соответствуют операторам $A_0, A_1, ..., A_n$, столбцы – операторам $A_1, A_2, ..., A_k$, а элементы – логические функции связи между операторами алгоритма.

Пример

Дана ЛСА $\mathcal{U}_2 = A_0 A_1 p_1 \uparrow^1 \overline{p}_2 \uparrow^2 p_3 \uparrow^3 A_2 A_3 \downarrow^3 A_4 A_5 \downarrow^1 A_6 \downarrow^2 A_к$.

С учётом вышесказанного построим равносильную ей МСА:

$$
\mathcal{U}_1 = \begin{array}{c} \\ A_0 \\ A_1 \\ A_2 \\ A_3 \\ A_4 \\ A_5 \\ A_6 \end{array}
\begin{array}{c}
\begin{array}{ccccccc} A_1 & A_2 & A_3 & A_4 & A_5 & A_6 & A_к \end{array} \\
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & p_1 \overline{p}_2 p_3 & 0 & p_1 \overline{p}_2 \overline{p}_3 & 0 & \overline{p}_1 & p_1 p_2 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\end{array}
$$

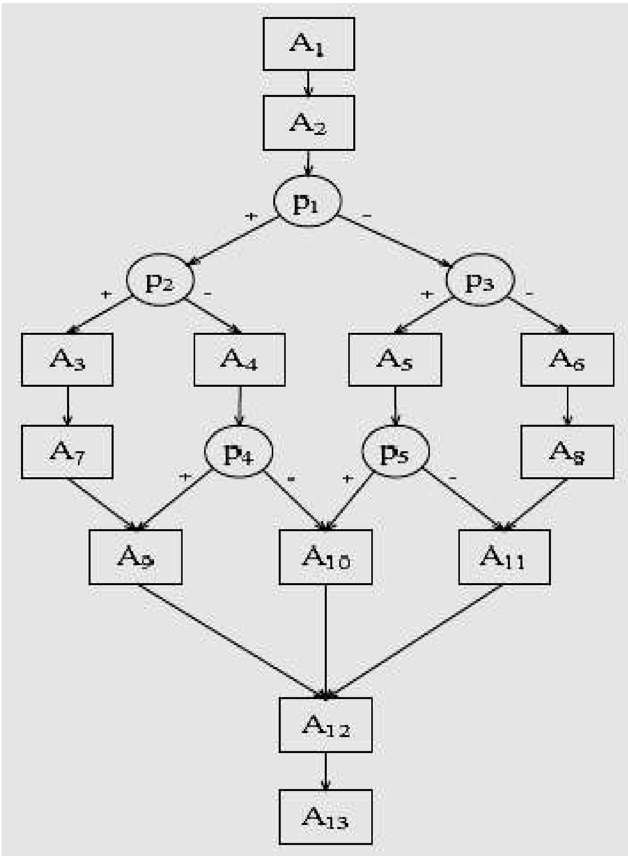
Граф-схемы алгоритмов

Иногда с целью большей наглядности, особенно в случае простых алгоритмов, удобно представлять их с помощью языка графов. Такое формальное описание было предложено российским математиком Львом Аркадьевичем Калужниным.

Определение

Граф-схема алгоритма (ГСА) есть конечный связный ориентированный граф G, удовлетворяющий следующим условиям:

1. В графе G имеется два отмеченных узла – входной, из которого выходит не более одной стрелки, и выходной, не имеющий ни одной выходящей стрелки.
2. Из каждого узла, отличного от входного и выходного, исходит либо одна стрелка (γ-узел), либо две стрелки (β-узел). Стрелки β-узла помечены – одна плюсом, другая минусом или одна 0, другая 1.
3. Имеется два конечных множества функциональных элементов – множество преобразователей информации $Q = \{A_1, A_2, ..., A_n\}$ и множество распознавателей $\mathscr{P} = \{p_1, p_2, ..., p_m\}$.
4. Каждому γ-узлу однозначно сопоставлен преобразователь $A_i \in Q$, а каждому β-узлу – распознаватель $\alpha_l(p_1, p_2, ...) \in \mathscr{P}$. В зависимости от алгоритма преобразователи и распознаватели могут повторяться в графе G



Эквивалентные преобразования схем алгоритмов

Преобразования ГСА → ЛСА, ЛСА → ГСА, ЛСА → МСА и ГСА → МСА интуитивно понятны и не требуют комментариев. Преобразование МСА → ГСА осуществляется через ЛСА. Рассмотрим преобразование МСА → ЛСА, которое широко применяется при объединении алгоритмов.

1) По строкам МСА записываем систему формул перехода S_1:

$$\left.\begin{array}{l} A_0 \neg \overset{n}{\underset{1}{\vee}} \alpha_{0j} A_j \\ A_1 \neg \overset{n}{\underset{1}{\vee}} \alpha_{1j} A_j \\ \cdots\cdots\cdots\cdots \\ A_n \neg \overset{n}{\underset{1}{\vee}} \alpha_{nj} A_j \end{array}\right\}$$

Логическая функция α_{ij} является «приведенной» по переменной, если она представлена в виде

$$\alpha_{ij} = p_s \alpha'_{ij} \vee \overline{p_s} \alpha''_{ij}$$

где p_s – логическая переменная,

$\alpha'_{ij}, \alpha''_{ij}$ - логические функции не содержащие переменной p_s и её отрицания

2) Приводим систему формул перехода по всем переменным p_s , т. е. выносим за скобки p_s и их отрицание, получая скобочную систему формул перехода S_2.

3) Переходим к схемной системе формул перехода S_3 за 3 операции:

1. Опустить все скобки в формуле (если они имеются).
2. Заменить символ логического ИЛИ на «*».
3. Заменить связанные пары переменных p_i и $\overline{p_i}$ (или $\overline{p_i}$ и p_i) на пары $p_i \uparrow^s$ и $\downarrow^s$ (или $\overline{p_i} \uparrow^s$ и $\downarrow^s$).

Схемные формулы перехода состоят из «блоков», разделенных « * », с помощью которых будет построена ЛСА.

4) Проводим эквивалентные преобразования системы схемных формул перехода с целью устранения повторяющихся операторов A_j и минимизации количества логических переменных p_s, получая системы схемных формул перехода S_3’, S_3’’ и т. д.

При этом используем следующие правила тождественных преобразований схемных формул перехода:

I. 1. $(A_i \neg N) = (A_i \neg 1 \uparrow^i N \downarrow^i)$.

2. $(A_i \neg N) = (A_i \neg \downarrow^i N 1 \uparrow^i)$.

II. 1. $(A_i \neg R_{i1} \downarrow^{s1} N * R_{i2} \downarrow^{s2} N * w) =$
$= (A_i \neg R_{i1} \downarrow^i \downarrow^{s1} \downarrow^{s2} N * R_{i2} \omega \uparrow^i * w) =$
$= (A_i \neg R_{i1} \omega \uparrow^i * R_{i2} \downarrow^i \downarrow^{s1} \downarrow^{s2} N * w)$.

2. $(A_{i1} \neg p_1 p_1 \uparrow^i p_2 A_m * \downarrow^i p_3 A_i) =$
$= (A_{i_1} \neg p_1 \overline{p}_i \uparrow^i p_3 A_i * \downarrow^i A_m)$.

III.	$\begin{cases} A_{i1}\neg w_{i1} * \downarrow^{s1} N \\ \vdots \\ A_{in}\neg w_{is} * R \downarrow^{s2} N \end{cases} = \begin{cases} A_{i1}\neg w_{i1} \\ \vdots \\ A_{in}\neg w_{is} * \downarrow^{s1}\downarrow^{s2} N \end{cases}$	
IV.	$(A_i\neg P_1 p_r \uparrow^r P_2 N_k * \downarrow^r P_3 N_l * w) =$ $= (A_i\neg P_1 p_r \uparrow^r P_2 N_k * \downarrow^r p_r \uparrow^s N_q * \downarrow^s P_3 N_l w),$ Здесь N_q – любая произвольно выбранная подформула.	
V.	$(A_i\neg Rp \uparrow^i N * \downarrow^i \downarrow^s \lambda * w) =$ $= (A_i\neg R \downarrow^i \downarrow^s p \uparrow^i N * w)$, Здесь 1 – пустая подформула.	
VI.	Концы стрелок, стоящие рядом, можно использовать в любом порядке $\downarrow^i\downarrow^l = \downarrow^l\downarrow^i$	
VII.	Пару стрелок $\uparrow^i, \downarrow^i$ входящих в формулы (подформулу), можно переименовать, если новые символы не совпадут с уже использованными ранее.	

где N_l – подформулы,
R_l – остаток выражения, не вошедший в подформулу,
w – выражение, не вошедшее ни в одну подформулу и в остаток R_l

5) Строим ЛСА по минимизированной системе схемных формул перехода.

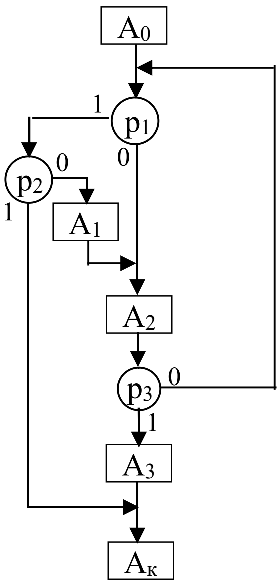
Вначале в строке выбирается оператор A_0. К нему справа приписывается начальное выражение из его схемной формулы.

Если при этом последним символом является оператор A_i, то справа к нему приписывают начальное выражение его схемной формулы. Но если последний символ – $\omega\uparrow^s$ (тождественно равная нулю логическая функция), то проверяют наличие ещё не выписанных начальных выражений из любой формулы перехода, и в случае их присутствия – выписывают одно из них.

Ещё одной особенностью является ситуация, когда оператор $A_к$ появляется раньше, чем окончена строка ЛСА. Хотя теоретически в этом никакой некорректности нет, всё же удобнее иметь ЛСА, в которой A_0 стоит в начале строки, а $A_к$ – в конце. Поэтому, когда $A_к$ из полученной системы S3 выписывают раньше, чем желательно, приходится принудительно переносить его в предполагаемый конец строки ЛСА с помощью введения дополнительной функции $\omega\uparrow^i$, где i – неиспользованный в S3 номер стрелки.

Пример. Составить граф-схему алгоритма, выводящего номер третьего элемента массива, большего 5, или 0, если его нет. Провести эквивалентное преобразование ГСА в ЛСА, затем ЛСА в МСА и МСА в ЛСА. Массив состоит из 10 целых чисел. Начальный оператор A_0 – ввод массива и инициализация переменных, конечный оператор $A_к$ – вывод результата на экран.

ГСА



A_0 : Ввод массива, i=1, c=0
A_1 : c++
A_2 : i++
A_3 : i=0
$A_к$: Вывод i

p_1 : M[i]>5 ?
p_2 : c>2 ?
p_3 : i>10 ?

ЛСА

$$U = A_0\downarrow^3 p_1\uparrow^1 p_2\uparrow^2 \omega\uparrow^4\downarrow^2 A_1\downarrow^1 A_2\, p_3\uparrow^3 A_3\downarrow^4 A_к$$

МСА

	A_1	A_2	A_3	$A_к$
A_0	$p_1!p_2$	$!p_1$	-	p_1p_2
A_1	-	1	-	-
A_2	$!p_3p_1!p_2$	$!p_3!p_1$	p_3	$!p_3p_1p_2$
A_3	-	-	-	1

Система формул перехода S_1

$A_0 \rightarrow p_1!p_2A_1$ V $!p_1A_2$ V $p_1p_2A_к$
$A_1 \rightarrow A_2$
$A_2 \rightarrow !p_3p_1!p_2A_1$ V $!p_3!p_1A_2$ V p_3A_3 V $!p_3p_1p_2A_к$
$A_3 \rightarrow A_к$

Скобочная система формул перехода S_2

$A_0 \rightarrow p_1(p_2A_к$ V $!p_2A_1)$ V $!p_1A_2$
$A_1 \rightarrow A_2$
$A_2 \rightarrow p_3\, A_3$ V $!p_3(p_1(p_2A_к$ V $!p_2A_1)$ V $!p_1A_2)$
$A_3 \rightarrow A_к$

Схемная система формул перехода S_3

$A_0 \rightarrow p_1\uparrow^1 p_2\uparrow^2 A_к * \downarrow^2 A_1 * \downarrow^1 A_2$
$A_1 \rightarrow A_2$
$A_2 \rightarrow p_3\uparrow^3 A_3 * \downarrow^3 p_1\uparrow^1 p_2\uparrow^2 A_к * \downarrow^2 A_1 * \downarrow^1 A_2$
$A_3 \rightarrow A_к$

Преобразованная схемная система формул перехода S_3’

$A_0 \rightarrow \downarrow^3 p_1 \uparrow^1 p_2 \uparrow^2 A_к * \downarrow^2 A_1$
$A_1 \rightarrow \downarrow^1 A_2$
$A_2 \rightarrow p_3 \uparrow^3 A_3$
$A_3 \rightarrow A_к$

Минимизированная схемная система формул перехода S_3''
$A_0 \rightarrow \downarrow^3 p_1 \uparrow^1 p_2 \uparrow^2 \omega \uparrow^4 * \downarrow^2 A_1$
$A_1 \rightarrow \downarrow^1 A_2$
$A_2 \rightarrow p_3 \uparrow^3 A_3$
$A_3 \rightarrow \downarrow^4 A_к$

ЛСА, построенная по минимизированной схемной системе формул перехода
$U = A_0 \downarrow^3 p_1 \uparrow^1 p_2 \uparrow^2 \omega \uparrow^4 \downarrow^2 A_1 \downarrow^1 A_2\, p_3 \uparrow^3 A_3 \downarrow^4 A_к$

Построенная ЛСА совпадает с исходной. Преобразование выполнено верно.

Задача. Составить граф-схему алгоритма, выводящего номер предпоследнего элемента массива, большего 5, или 0, если его нет. Провести эквивалентное преобразование ГСА в ЛСА, затем ЛСА в МСА и МСА в ЛСА. Массив состоит из 10 целых чисел. Начальный оператор A_0 – ввод массива и инициализация переменных, конечный оператор $A_к$ – вывод результата на экран.

Объединение 2-х алгоритмов

Задача. Автомат должен выполнять два алгоритма $\boldsymbol{U_1}$ и $\boldsymbol{U_2}$. Если логическое условие **r** истинно, то выполняется $\boldsymbol{U_1}$, иначе - $\boldsymbol{U_2}$.

Получить минимизированный объединенный алгоритм $\boldsymbol{U_{об}}$, в котором отсутствуют повторяющиеся операторы.

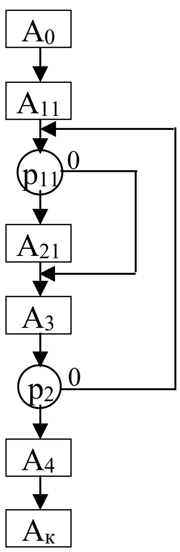
Алгоритм решения.

1. Составить граф-схему и логическую схему каждого алгоритма.
2. Оценить эффективность минимизации объединенного алгоритма по количеству общих (одинаковых) операторов и логических переменных исходных алгоритмов. Если это количество меньше 2, то минимизация неэффективна, $U_{об} = r\uparrow^1 U_1 \omega\uparrow^2 \downarrow^1 U_2 \downarrow^2$, иначе перейти к п. 3.
3. Составить объединенную МСА, в которой операторы не повторяются. Если исходный оператор присутствует в обоих МСА, то проставляем **r** при переходе к оператору из 1-го алгоритма и **!r** при переходе к оператору из 2-го алгоритма.
4. Составить систему формул перехода, провести ее минимизацию и упрощение.
5. По минимизированной системе схемных формул перехода составить объединенную ЛСА.
6. Проверить выполнение каждого алгоритма в объединенной ЛСА, подставляя соответствующее значение **r**.

Пример. $\boldsymbol{U_1}$ – вывести максимум массива из 10 элементов.
$\boldsymbol{U_2}$ – вывести номер первого элемента массива, равного 4.

1)

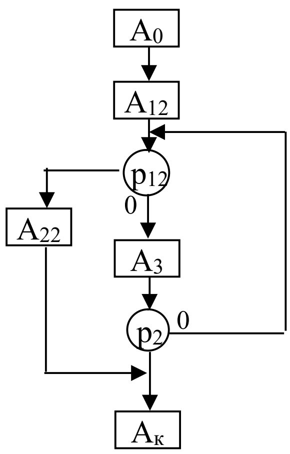
Алгоритм $\boldsymbol{U_1}$



A_0 : Ввод массива, i=1
A_{11} : max=M[1]
A_{21} : max=M[i]
A_3 : i++
A_4 : prn=max
$A_к$: print(prn)

p_{11} : M[i]>max ?
p_2 : i>10 ?

Алгоритм $\boldsymbol{U_2}$



A_0 : Ввод массива, i=1
A_{12} : prn=0
A_{22} : prn=i
A_3 : i++
$A_к$: print(prn)

p_{12}: M[i]=4 ?
p_2 : i>10 ?

$U_1 = A_0 A_{11} \downarrow^2 p_{11} \uparrow^{11} A_{21} \downarrow^{11} A_3 p_2 \uparrow^2 A_4 A_к$

$U_2 = A_0 A_{12} \downarrow^2 p_{12} \uparrow^{12} A_{22} \omega \uparrow^3 \downarrow^{12} A_3 p_2 \uparrow^2 \downarrow^3 A_к$

2) Общие операторы и логические переменные – A_0 , A_3 , p_2 , $A_к$. Минимизация должна быть эффективна.

3) На базе ГС или ЛС исходных алгоритмов строим объединенную МСА, в заголовках строк и столбцов которой операторы обоих алгоритмов. Переходы между операторами одного алгоритма соответствуют ЛСА или ГСА этого алгоритма. При переходе от общего оператора к оператору определенного алгоритма вставляется соответствующее логическое условие r или !r.

Объединенная МСА

	A_{11}	A_{12}	A_{21}	A_{22}	A_3	A_4	$A_к$
A_0	r	!r					
A_{11}			p_{11}		$!p_{11}$		
A_{12}				p_{12}	$!p_{12}$		
A_{21}					1		
A_{22}							1
A_3			$r\,!p_2 p_{11}$	$!r\,!p_2 p_{12}$	$r\,!p_2\,!p_{11} \vee$ $!r\,!p_2\,!p_{12}$	$r\,p_2$	$!r\,p_2$
A_4							1

4) По объединенной МСА строим системы скобочных и схемных формул перехода и затем строим объединенную ЛСА.

Скобочная система формул перехода S_2

$A_0 \rightarrow rA_{11} \vee\ !rA_{12}$

$A_{11} \rightarrow p_{11} A_{21} \vee !p_{11}A_3$

$A_{12} \rightarrow p_{12} A_{22} \vee !p_{12}A_3$

$A_{21} \rightarrow A_3$

$A_{22} \rightarrow A_к$

$A_3 \rightarrow p_2 (rA_4 \vee !rA_к) \vee !p_2 (r (p_{11}A_{21} \vee !p_{11}A_3) \vee !r (p_{12}A_{22} \vee !p_{12}A_3))$

$A_4 \rightarrow A_к$

Схемная система формул перехода S_3

$A_0 \rightarrow r\uparrow^4 A_{11} * \downarrow^4 A_{12}$

$A_{11} \rightarrow p_{11}\uparrow^{11} A_{21} * \downarrow^{11} A_3$

$A_{12} \rightarrow p_{12}\uparrow^{12} A_{22} * \downarrow^{12} A_3$

$A_{21} \rightarrow A_3$

$A_{22} \rightarrow A_к$

$A_3 \rightarrow p_2\uparrow^2 r\uparrow^5 A_4 * \downarrow^5 A_к * \downarrow^2 r\uparrow^6 p_{11}\uparrow^{11} A_{21} * \downarrow^{11} A_3 * \downarrow^6 p_{12}\uparrow^{12} A_{22} * \downarrow^{12} A_3$

$A_4 \rightarrow A_к$

Преобразованная схемная система формул перехода S_3'

$A_0 \rightarrow r\uparrow^4 A_{11} * \downarrow^4 A_{12}$

$A_{11} \rightarrow \downarrow^2 r\uparrow^6 p_{11}\uparrow^7 A_{21}$

$A_{12} \rightarrow \downarrow^6 p_{12}\uparrow^7 A_{22}$

$A_{21} \rightarrow \downarrow^7 A_3$

$A_{22} \rightarrow \omega\uparrow^5$

$A_3 \rightarrow p_2\uparrow^2 r\uparrow^5 A_4$
$A_4 \rightarrow \downarrow^5 A_к$

5) Объединенная ЛСА
$U_{об} = A_0\ r\uparrow^4 A_{11}\downarrow^2\ r\uparrow^6 p_{11}\uparrow^7 A_{21}\downarrow^7 A_3\ p_2\uparrow^2 r\uparrow^5 A_4\ \omega\uparrow^5\downarrow^4 A_{12}\downarrow^6 p_{12}\uparrow^7 A_{22}\downarrow^5 A_к$

6) Проверка:
r=1 $U_1 = A_0 A_{11}\downarrow^2 p_{11}\uparrow^7 A_{21}\downarrow^7 A_3 p_2\uparrow^2 A_4 A_к$
r=0 $U_2 = A_0 A_{12}\downarrow^6 p_{12}\uparrow^7 A_{22}\omega\uparrow^5\downarrow^7 A_3 p_2\uparrow^6\downarrow^5 A_к$

При подстановке соответствующих значений **r** оба алгоритма выполняются, следовательно объединенная ЛСА составлена правильно.

В объединенной ЛСА 14 элементарных выражений, а в необъединенной – 16. Минимизация эффективна.

Объединить 2 алгоритма работы с массивом из 10 элементов:
***U*1** – вывести сумму элементов массива;
***U*2** – вывести номер первого элемента массива, большего 5.
Первый оператор A_0 – инициализация массива и переменных.
Последний оператор $A_к$ – вывод искомого значения на экран.

Объединение 3-х и более алгоритмов

Задача. Автомат должен выполнять три алгоритма U_1, U_2 и U_3 в зависимости от значений набора логических условий $\mathbf{r_1}$, $\mathbf{r_2}$.

Получить минимизированный объединенный алгоритм $U_{об}$, в котором отсутствуют повторяющиеся операторы.

Алгоритм решения.

1. Составить граф-схему и логическую схему каждого алгоритма.

2. Оценить эффективность минимизации объединенного алгоритма по количеству общих (одинаковых) операторов и логических переменных исходных алгоритмов. Если это количество меньше 2, то минимизация неэффективна, $U_{об} = r_1\uparrow^1 r_2\uparrow^2 U_1\omega\uparrow^3 \downarrow^1 U_2\omega\uparrow^3\downarrow^2 U_3\downarrow^3$, иначе перейти к п. 3.

3. Составить матричную схему каждого алгоритма.

4. Построить объединенную МСА, каждый элемент которой

$$\hat{\alpha}_{jq} = \bigvee_{i=1}^{l} \alpha^i_{jq} \beta^i_j$$

где α^i_{jq} – логический элемент матрицы МСА $\mathcal{U}_i$, где q – номер строки, а j – номер столбца в $\mathcal{U}_i$;
β^i_j – определяющая функция оператора A_j из ЛСА $\mathcal{U}_i$;

Определяющие функции вычисляются по формуле:

$$\beta^i_j = R_i \vee \frac{R_1}{0} \vee \frac{R_2}{0} \vee \ldots \vee \frac{R_{2^l}}{0}$$

Здесь $\mathbf{R_i}$ - определяющие конъюнкции для всех объединяемых МСА.

Оператор $\mathbf{A_i}$ входит только в ЛСА U_i , поэтому есть выбор **R/0**, который позволяет минимизировать систему формул перехода и объединенную ЛСА.

Практика объединения алгоритмов показывает, что МСА с наибольшим числом совпадающих элементов желательно приписывать определяющие конъюнкции с соседним кодированием.

Совпадающие элементы:

- элементы, состоящие из одинаковых логических функций;
- элементы строки оператора $\mathbf{A_i}$, если он отсутствует в другой МСА.

4.1. Построить неориентированный граф, вершины которого помечены обозначениями алгоритмов, а ребра - числом совпадающих элементов.

4.2. Закодировать определяющие конъюнкции алгоритмов набором логических переменных r_1r_2, используя соседнее кодирование для пар МСА с наибольшим числом совпадающих элементов.

4.3. Построить набор определяющих функций для каждого оператора каждого алгоритма.

4.4. Построить объединенную МСА.

5. Составить систему формул перехода, провести ее минимизацию и упрощение.

6. По системе схемных формул перехода составить объединенную ЛСА.

7. Проверить выполнение каждого алгоритма в объединенной ЛСА, подставляя соответствующие значения $\mathbf{r_1}$, $\mathbf{r_2}$. Если алгоритм имеет большое количество элементарных выражений и безусловных переходов, то проверку рекомендуется проводить по объединенной ГСА.

Пример. ***U*₁** – вывести максимум массива из 10 элементов.

***U*₂** – вывести минимум массива из 10 элементов.

***U*₃** – вывести номер первого элемента массива, равного 4.

1)

$U_1 = A_0A_{11}\downarrow^2 p_{11}\uparrow^{11}A_{21}\downarrow^{11}A_3p_2\uparrow^2A_{41}A_к$

$U_2 = A_0A_{12}\downarrow^2 p_{12}\uparrow^{12}A_{22}\downarrow^{12}A_3p_2\uparrow^2A_{42}A_к$

$U_3 = A_0A_{13}\downarrow^2 p_{13}\uparrow^{13}A_{23}\omega\uparrow^3\downarrow^{13}A_3p_2\uparrow^2\downarrow^3A_к$

2) Общие операторы и логические переменные – A_0 , A_3 , p_2 , $A_к$. Минимизация должна быть эффективна.

3)

***U*₁**

	A_{11}	A_{21}	A_3	A_{41}	$A_к$
A_0	1				
A_{11}		p_{11}	$!p_{11}$		
A_{21}			1		
A_3		$!p_2p_{11}$	$!p_2!p_{11}$	p_2	
A_{41}					1

***U*₂**

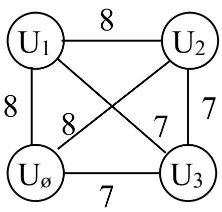
	A_{12}	A_{22}	A_3	A_{42}	$A_к$
A_0	1				
A_{12}		p_{12}	$!p_{12}$		
A_{22}			1		
A_3		$!p_2p_{12}$	$!p_2!p_{12}$	p_2	
A_{42}					1

***U*₃**

	A_{13}	A_{23}	A_3	$A_к$
A_0	1			
A_{13}		p_{13}	$!p_{13}$	
A_{23}				1
A_3		$!p_2p_{13}$	$!p_2!p_{13}$	p_2

4) Определяющие конъюнкции и функции

4.1) Граф с числами совпадающих элементов



***U*₁-*U*₂**: строки $A_{11}+A_{21}+A_{12}+A_{22}+A_{41}+A_{42} = 2+1+2+1+1+1 = 8$.

***U*₁-*U*₃**: строки $A_{11}+A_{21}+A_{41}+A_{13}+A_{23} = 2+1+1+2+1 = 7$.

***U*₂-*U*₃**: строки $A_{12}+A_{22}+A_{42}+A_{13}+A_{23} = 2+1+1+2+1 = 7$.

Для большей минимизации введена пустая МСА ***U*ø** , число совпадающих с ней элементов равно числу элементов всей матрицы.

4.2) Определяющие конъюнкции

Наибольшее число совпадающих элементов у пар матриц ***U*₁-*U*₂** , ***U*₁-*U*ø** , ***U*₂-*U*ø** .

Закодируем определяющие конъюнкции для ***U*₁** , ***U*₂** , ***U*ø** соседними кодами:

$R_1 = r_1r_2$, $R_2 = r_1!r_2$, $R_ø = !r_1r_2$, $R_3 = !r_1!r_2$.

4.3) Определяющие функции

Определяющие функции операторов A_0 и A_3 первого алгоритма (присутствуют во всех алгоритмах кроме пустого):

$$\beta_0^1 = \beta_3^1 = R_1 \vee \frac{R_\phi}{0} = r_1r_2 \vee \frac{!r_1r_2}{0} = \frac{r_1}{1}r_2 .$$

Определяющие функции остальных операторов первого алгоритма (присутствуют только в нем):

$$\beta_{11}^1 = \beta_{21}^1 = \beta_{41}^1 = R_1 \vee \frac{R_2}{0} \vee \frac{R_3}{0} \vee \frac{R_\phi}{0} = r_1r_2 \vee \frac{r_1!r_2}{0} \vee \frac{!r_1!r_2}{0} \vee \frac{!r_1r_2}{0} = 1 .$$

Аналогично для остальных алгоритмов:

$$\beta_0^2 = \beta_3^2 = R_2 \vee \frac{R_\phi}{0} = r_1!r_2 \vee \frac{!r_1 r_2}{0} = r_1!r_2 \;;$$

$$\beta_0^3 = \beta_3^3 = R_3 \vee \frac{R_\phi}{0} =!r_1!r_2 \vee \frac{!r_1 r_2}{0} =!r_1 \frac{!r_2}{1} \;;$$

$$\beta_{12}^2 = \beta_{22}^2 = \beta_{42}^2 = \beta_{13}^3 = \beta_{23}^3 = 1 \;.$$

4.4) Объединенная недоопределенная МСА

	A_{11}	A_{12}	A_{13}	A_{21}	A_{22}	A_{23}	A_3	A_{41}	A_{42}	$A_к$
A_0	$(r_1/1)r_2$	$r_1!r_2$	$!r_1(!r_2/1)$							
A_{11}				p_{11}			$!p_{11}$			
A_{12}					p_{12}		$!p_{12}$			
A_{13}						p_{13}	$!p_{13}$			
A_{21}							1			
A_{22}							1			
A_{23}										1
A_3				$!p_2p_{11}$ $(r_1/1)r_2$	$!p_2p_{12}r_1!r_2$	$!p_2p_{13}$ $!r_1(!r_2/1)$	$!p_2!p_{11}(r_1/1)r_2$v $!p_2!p_{12}r_1!r_2$ v $!p_2!p_{13}!r_1(!r_2/1)$	$p_2(r_1/1)r_2$	$p_2r_1!r_2$	$p_2!r_1$ $(!r_2/1)$
A_{41}										1
A_{42}										1

Доопределяем МСА, сокращая число и сохраняя полноту логических условий.

	A_{11}	A_{12}	A_{13}	A_{21}	A_{22}	A_{23}	A_3	A_{41}	A_{42}	$A_к$
A_0	r_1r_2	$r_1!r_2$	$!r_1$							
A_{11}				p_{11}			$!p_{11}$			
A_{12}					p_{12}		$!p_{12}$			
A_{13}						p_{13}	$!p_{13}$			
A_{21}							1			
A_{22}							1			
A_{23}										1
A_3				$!p_2p_{11}$ r_1r_2	$!p_2p_{12}r_1!r_2$	$!p_2p_{13}!r_1$	$!p_2!p_{11}r_1r_2$v $!p_2!p_{12}r_1!r_2$ v $!p_2!p_{13}!r_1$	$p_2r_1r_2$	$p_2r_1!r_2$	$p_2!r_1$
A_{41}										1
A_{42}										1

5) Скобочная система формул перехода S_2

$A_0 \rightarrow r_1r_2A_{11}$ V $r_1!r_2A_{12}$ V $!r_1A_{13} \rightarrow r_1(r_2A_{11}$ V $!r_2A_{12})$ V $!r_1A_{13}$
$A_{11} \rightarrow p_{11}A_{21}$ V $!p_{11}A_3$
$A_{12} \rightarrow p_{12}A_{22}$ V $!p_{12}A_3$
$A_{13} \rightarrow p_{13}A_{23}$ V $!p_{13}A_3$
$A_{21} \rightarrow A_3$
$A_{22} \rightarrow A_3$
$A_{23} \rightarrow A_к$

$A_3 \rightarrow !p_2r_1r_2(p_{11}A_{21} \vee !p_{11}A_3) \vee !p_2r_1!r_2(p_{12}A_{22} \vee !p_{12}A_3) \vee !p_2!r_1(p_{13}A_{23} \vee !p_{13}A_3) \vee p_2(r_1(r_2A_{41} \vee !r_2A_{42}) \vee !r_1A_к) \rightarrow p_2(r_1(r_2A_{41} \vee !r_2A_{42}) \vee !r_1A_к) \vee !p_2(r_1(r_2(p_{11}A_{21} \vee !p_{11}A_3) \vee !r_2(p_{12}A_{22} \vee !p_{12}A_3)) \vee !r_1(p_{13}A_{23} \vee !p_{13}A_3))$

$A_{41} \rightarrow A_к$

$A_{42} \rightarrow A_к$

Схемная система формул перехода S_3

$A_0 \rightarrow r_1\uparrow^1 r_2\uparrow^2 A_{11} * \downarrow^2 A_{12} * \downarrow^1 A_{13}$

$A_{11} \rightarrow p_{11}\uparrow^{11} A_{21} * \downarrow^{11} A_3$

$A_{12} \rightarrow p_{12}\uparrow^{12} A_{22} * \downarrow^{12} A_3$

$A_{13} \rightarrow p_{13}\uparrow^{13} A_{23} * \downarrow^{13} A_3$

$A_{21} \rightarrow A_3$

$A_{22} \rightarrow A_3$

$A_{23} \rightarrow A_к$

$A_3 \rightarrow p_2\uparrow^3 r_1\uparrow^4 r_2\uparrow^5 A_{41} * \downarrow^5 A_{42} * \downarrow^4 A_к * \downarrow^3 r_1\uparrow^6 r_2\uparrow^7 p_{11}\uparrow^{11} A_{21} * \downarrow^{11} A_3 * \downarrow^7 p_{12}\uparrow^{12} A_{22} * \downarrow^{12} A_3 * \downarrow^6 p_{13}\uparrow^{13} A_{23} * \downarrow^{13} A_3$

$A_{41} \rightarrow A_к$

$A_{42} \rightarrow A_к$

Преобразованная схемная система формул перехода S_3'

$A_0 \rightarrow r_1\uparrow^1 r_2\uparrow^2 A_{11} * \downarrow^2 A_{12} * \downarrow^1 A_{13}$

$A_{11} \rightarrow \downarrow^8 p_{11}\uparrow^9 A_{21}$

$A_{12} \rightarrow \downarrow^7 p_{12}\uparrow^9 A_{22}$

$A_{13} \rightarrow \downarrow^6 p_{13}\uparrow^9 A_{23}$

$A_{21} \rightarrow \omega\uparrow^9$

$A_{22} \rightarrow \downarrow^9 A_3$

$A_{23} \rightarrow \omega\uparrow^{10}$

$A_3 \rightarrow p_2\uparrow^3 r_1\uparrow^{10} r_2\uparrow^5 A_{41} * \downarrow^5 A_{42} * \downarrow^3 r_1\uparrow^6 r_2\uparrow^7 \omega\uparrow^8$

$A_{41} \rightarrow \omega\uparrow^{10}$

$A_{42} \rightarrow \downarrow^{10} A_к$

6) Объединенная ЛСА

$U_{об} = A_0\, r_1\uparrow^1 r_2\uparrow^2 A_{11}\downarrow^8 p_{11}\uparrow^9 A_{21}\omega\uparrow^9\downarrow^2 A_{12}\downarrow^7 p_{12}\uparrow^9 A_{22}\downarrow^9 A_3 p_2\uparrow^3 r_1\uparrow^{10} r_2\uparrow^5 A_{41}\omega\uparrow^{10}\downarrow^1 A_{13}\downarrow^6 p_{13}\uparrow^9 A_{23}\omega\uparrow^{10}\downarrow^3 r_1\uparrow^6 r_2\uparrow^7\omega\uparrow^8\downarrow^5 A_{42}\downarrow^{10} A_к$

7) Проверка:

$r_1=1,\ r_2=1 \quad U_1 = A_0A_{11}\downarrow^3 p_{11}\uparrow^9 A_{21}\downarrow^9 A_3p_2\uparrow^3 A_{41}A_к$

$r_1=1,\ r_2=0 \quad U_2 = A_0A_{12}\downarrow^3 p_{12}\uparrow^9 A_{22}\downarrow^9 A_3p_2\uparrow^3 A_{42}A_к$

$r_1=0,\ r_2=0 \quad U_3 = A_0A_{13}\downarrow^3 p_{13}\uparrow^9 A_{23}\omega\uparrow^{10}\downarrow^9 A_3p_2\uparrow^3\downarrow^{10} A_к$

При подстановке соответствующих значений **r** все 3 алгоритма выполняются правильно, следовательно объединенная ЛСА составлена верно.

В объединенной ЛСА 21 элементарное выражение, а в необъединенной – 25. Минимизация эффективна.

Задача для подготовки.

Объединить 3 алгоритма:

$\boldsymbol{U_1}$ – вывести максимум массива из 10 элементов.

$\boldsymbol{U_2}$ – вывести номер первого элемента массива, большего 5.

$\boldsymbol{U_3}$ – упорядочить массив в порядке возрастания и вывести последний элемент.

Первый оператор A_0 – инициализация массива и переменных.

Последний оператор $A_к$ – вывод искомого значения на экран.