

АЛГОРИТМЫ РЕШЕНИЯ ЗАДАЧ ДИСКРЕТНОЙ МАТЕМАТИКИ





Сибирский ордена Трудового Красного Знамени
физико-технический институт им. В.Д.
Кузнецова при Томском государственном
университете

АЛГОРИТМЫ РЕШЕНИЯ
ЗАДАЧ ДИСКРЕТНОЙ МАТЕМАТИКИ

Выпуск 2

Под редакцией Г.П. Агibalова



Издательство Томского университета

Томск - 1987

1-03087a7.к

УДК 519.7.681.3

Алгоритмы решения задач дискретной математики, вып.2.-
Томск: Изд-во Томского ун-та, 1987. - 203 с. I р. 60 к. 500 экз.
I502000000.

Представлены результаты исследований по дискретной математике и ее приложениям в системах автоматизированного проектирования цифровых автоматов на базе интегральных схем, в том числе элементы теории функциональных систем на полурешетках, условия существования некоторых нетривиальных декомпозиций конечных автоматов, алгоритмы синтеза и имитационного моделирования функциональных схем и микропроцессорных систем, алгоритмы обхода и раскраски графов, алгоритмы конструирования схем, включая их декомпозицию, размещение и трассировку, алгоритмы технической диагностики и некоторые другие.

Для специалистов по дискретной математике и автоматизированному проектированию цифровой электронной техники.

БИБ. 2482

Рецензент - С.Б.Быкова

I502000000 40-87
А I77(012)-87

Г. П. Агибалов

ФУНКЦИОНАЛЬНЫЕ СИСТЕМЫ НА ПОЛУРЕШЕТКАХ

1. Введение

Множество функциональных объектов (функций, отображений, автоматов, алгоритмов и т. п.), на котором заданы некоторые операции, принято называть функциональной системой с операциями [1]. Если функциональные объекты в системе действуют на наборы элементов или последовательностей элементов из некоторого множества L , то говорят, что эта система определена на L . Как предмет для исследования функциональные системы возникают обычно в связи с изучением тех или иных технических систем, моделями которых они служат. Важнейшими функциональными системами, связанными с цифровой электронной техникой, являются алгебра логики, конечнозначная логика, система автоматных функций, класс вычислимых функций и некоторые другие, изучаемые в соответствующих разделах дискретной математики [1]. Все они ориентированы в приложениях на моделирование статического (при фиксированном входном состоянии) поведения цифровых устройств. Их характерная особенность состоит в их определенности на неупорядоченных или линейно упорядоченных множествах: алгебра логики — на множестве $\{0, 1\}$, k -значная логика и система конечно-автоматных функций — на множестве $\{0, 1, \dots, k-1\}$, $k \geq 2$, класс вычислимых функций — на множестве $\{0, 1, \dots, k, \dots\}$ и т. д. Потребность в отображении динамических явлений, возникающих в устройствах при изменении входных состояний, приводит к необходимости рассмотрения функциональных систем более общего вида, а именно систем, определенных на частично упорядоченных множествах с произвольным (не обязательно линейным) порядком. Отношение порядка в них является отношением сравнения величин по степени неопределенности, обязанной явлению состязаний представляемых ими сигналов в устройстве. Результат состязания последних отображается в такой системе суммой соответствующих величин, превращая их множество в полурешетку.

Предлагаемую вниманию работу следует рассматривать как краткое введение в теорию функциональных систем на полурешетках.

2. Полурешетки

Частично упорядоченное множество, в котором всякое двухэлементное подмножество имеет точную верхнюю грань, называется аддитивной полурешеткой [2]. Множество элементов в ней называется носителем, а их число — мощностью полурешетки. На любой аддитивной полурешетке можно ввести операцию сложения, определив ее результат (сумму) $a + b$ для элементов a и b как $\sup(a, b)$ — точную верхнюю грань пары $\{a, b\}$. Эта операция обладает обычными для структурных операций свойствами идемпотентности, коммутативности и ассоциативности и с соответствующим отношением порядка $\leq$ находится в соотношении: $a + b = b \leftrightarrow a \leq b$. Обратно, всякое множество с бинарной операцией $+$, обладающей указанными свойствами, является аддитивной полурешеткой, в которой отношение порядка определяется последним соотношением, при этом оказывается, что $a + b = \sup(a, b)$. Всюду далее под полурешеткой понимается конечная (конечной мощности) аддитивная полурешетка. В ней обязательно существует максимальный элемент, равный сумме всех элементов, и имеются минимальные элементы, т.е. такие элементы a , для которых из $b \leq a$ следует $b = a$. Множество минимальных элементов полурешетки называется базисом, а элементы в нем — базисными элементами полурешетки. В случае $a \leq b$ говорят, что a и b сравнимы, а предшествует b и b следует за a . Непустое подмножество полурешетки называется подполурешеткой, если оно замкнуто относительно операции сложения. Всякая подполурешетка по определению является полурешеткой. Подполурешетка S называется выпуклой, если для любых элементов a, b, c полурешетки из $a \leq c \leq b$ и $a, b \in S$ следует $c \in S$. Говорят, что подполурешетка порождается своим подмножеством A , если каждый ее элемент представим в виде суммы некоторых элементов из A . Подполурешетка, порожденная множеством A , обозначается через $\langle A \rangle$. Это есть минимальная подполурешетка, целиком содержащая A . При заданном A ее элементами являются всевозможные суммы по k различных слагаемых из A для $k = 1, 2, \dots, |A|$. Полурешетка называется нормальной, если она порождается своим базисом.

Полурешетка, элементами которой являются подмножества некоторого основного множества M , а отношением порядка — отношение включения, называется полурешеткой подмножеств (на M). Произвольная совокупность $\mathcal{L}$ подмножеств множества M будет полурешеткой

по этому отношению тогда и только тогда, когда в ней для любых двух подмножеств найдется одно и только одно такое подмножество, включающее оба их в себя, которое не включает другого подмножества с тем же свойством, т.е. если и только если

$$\forall a, b \in L \exists c \in L (a \cup b \subseteq c \wedge \forall d \in L (a \cup b \subseteq d \rightarrow c \subseteq d)).$$

Во всякой полурешетке подмножеств если $a + b = c$, то $a \cup b \subseteq c$ и, возможно, $a \cup b \neq c$. Множество всех непустых подмножеств множества M является примером полурешетки, в которой $a + b = a \cup b$. В дальнейшем мы обозначаем ее через $\tilde{M}$.

Важный класс полурешеток подмножеств составляют полурешетки интервалов. Под интервалом в общем случае понимается декартово произведение конечного числа конечных и непустых подмножеств некоторых множеств. Сомножители в нем называются компонентами, а их число — размерностью интервала. Интервал, являющийся декартовым произведением $\alpha_1 \times \alpha_2 \times \dots \times \alpha_z$, записывается далее как $\alpha_1 \alpha_2 \dots \alpha_z$. Интервал, все компоненты которого являются подмножествами некоторого одного и того же множества B , называется интервалом над множеством B , а интервал, который сам есть подмножество некоторого множества M , — интервалом на M . Для одномерного интервала понятия „над множеством“ и „на множестве“ совпадают. Более того, всякое подмножество множества M является определенно одномерным интервалом на M . В случае же многомерных интервалов это утверждение не имеет места, именно: если M есть декартово произведение z сомножителей, $z \geq 2$ и $A \subseteq M$, то A — не обязательно z -мерный интервал на M .

В дальнейшем под интервалом на декартовом произведении из z сомножителей всегда подразумевается z -мерный интервал. Имея это в виду, заметим, что объединение интервалов может не быть интервалом, в то время как непустое пересечение любого числа интервалов есть всегда интервал: его компоненты суть общие части соответствующих компонент пересекающихся интервалов. Из последнего предложения, в частности, следует, что минимальный интервал, покрывающий заданное непустое подмножество, единственен и равен пересечению всех интервалов, целиком содержащих данное подмножество. Его i -я компонента является множеством i -х компонент всех элементов этого подмножества.

Полурешетка, элементами которой являются интервалы некоторой одной и той же размерности, называется полурешеткой интервалов. Во всякой такой полурешетке если $(a_1 \dots a_n) + (b_1 \dots b_n) = (c_1 \dots c_n)$, то $c_i \supseteq a_i$ для каждого $i = 1, \dots, n$ и не исключено, что

$c_i \neq a_i \vee b_i$ для некоторого $i \in \{1, \dots, n\}$. Множество всех интервалов декартова произведения M является примером полурешетки интервалов, в которой сумма интервалов совпадает с их покомпонентным объединением. В дальнейшем мы обозначаем ее через $\hat{M}$. В общем случае $\hat{M}$ не есть подполурешетка полурешетки $\hat{M}$.

Полурешетки $\hat{M}$ и $\hat{M}$ нормальные; их базисами является множество всех одноэлементных подмножеств множества M . В дальнейшем с целью упрощения выражений, рассматривая полурешетки подмножеств, в том числе полурешетки интервалов, мы не будем делать различий между одноэлементными подмножествами и их элементами. Имея это в виду, можно сказать, что базисами полурешеток $\hat{M}$ и $\hat{M}$ является множество M .

Для любых полурешеток $L_1, \dots, L_n$ декартово произведение $L_1 \times \dots \times L_n$ мы рассматриваем как полурешетку, в которой сложение определено покомпонентно:

$$(a_1, \dots, a_n) + (b_1, \dots, b_n) = (a_1 + b_1, \dots, (a_n + b_n)).$$

В этой полурешетке

$$(a_1, \dots, a_n) \leq (b_1, \dots, b_n) \leftrightarrow \forall i = 1, \dots, n (a_i \leq b_i).$$

Очевидно также, что если полурешетки $L_1, \dots, L_n$ нормальные, то нормальной будет и полурешетка $L_1 \times \dots \times L_n$.

Если A и B — подмножества элементов полурешетки L , то под $A + B$ будем понимать подмножество в L , состоящее из всех элементов вида $a + b$, где $a \in A$, $b \in B$, и будем называть $A + B$ суммой подмножеств A и B . Легко проверяются коммутативный и ассоциативный законы для сложения подмножеств. Кроме того, если A и B — подполурешетки, то $A + B$ — также подполурешетка и $A + A = A$. Эквивалентность ρ , определенная на полурешетке L , называется конгруэнцией на L , если для любых ее смежных классов A и B существует смежный класс C такой, что $A + B \subseteq C$. Легко проверить, что каждый смежный класс конгруэнции ρ является выпуклой подполурешеткой в L , а их множество L/ρ — полурешеткой с операцией сложения $\oplus$, определяемой как

$$A \oplus B = C \leftrightarrow A + B \subseteq C.$$

Полурешетка L/ρ называется фактор-полурешеткой полурешетки L по конгруэнции ρ . В ней для любых $a, b \in L$

$$[a]_\rho \oplus [b]_\rho = [a + b]_\rho,$$

где через $[a]_\rho$ обозначается смежный класс конгруэнции ρ , содержащий элемент a . Нетрудно убедиться также, что $[a]_\rho \leq [b]_\rho$, ес-

ли и только если существуют $c \in [a]_p$ и $d \in [b]_p$ такие, что $c \leq d$, поэтому класс $[a]_p$ тогда и только тогда является минимальным элементом фактор-полурешетки, когда вместе с каждым своим элементом $b \in [a]_p$ он содержит и всякий элемент $c \leq b$. Из этого, в частности, следует, что в любом минимальном элементе фактор-полурешетки имеется хотя бы один минимальный элемент полурешетки и что смежный класс, состоящий только из минимальных элементов полурешетки, непременно является минимальным элементом фактор-полурешетки.

Говорят, что полурешетка K моделирует полурешетку L , если существует сюръективное отображение $\varphi: L \rightarrow K$, при котором $\varphi(a+b) = \varphi(a) + \varphi(b)$ для всех $a, b \in L$; в этом случае φ называется гомоморфизмом L на K , а K — гомоморфным образом, или моделью полурешетки L при гомоморфизме φ . Взаимно однозначный гомоморфизм полурешеток называется изоморфизмом, а полурешетки, допускающие изоморфизм одной на другую, — изоморфными полурешетками.

Изоморфные полурешетки принято не различать. Объясняется это тем, что свойства полурешеток, которыми обычно интересуются, определяются не свойствами их отдельно взятых элементов, а соотношениями между элементами, выраженными в операции сложения, и являются, таким образом, общими для всех изоморфных полурешеток. Приложения же, которые мы имеем в виду в наших исследованиях, не позволяют, рассматривая, например, полурешетки подмножеств, абстрагироваться от состава последних. Именно в элементном составе подмножества, которым представляется состояние устройства, отображается характер возможных в устройстве состояний. У разных полурешеток, в том числе изоморфных, „способности“ для отображения этих явлений различны. Нас интересуют полурешетки подмножеств, с помощью которых возможно адекватное отображение состояний устройств. В определяемом ниже понятии адекватной модели полурешетки формализуется интуитивное представление об адекватности любой модели как о ее безошибочности (адекватная модель никогда не „лжёт“), которое в нашем конкретном случае выражается в требовании существования в полурешетке для каждого возможного состояния устройства подмножества, содержащего это состояние.

Пусть ρ — конгруенция на полурешетке L . Рассмотрим множество K максимальных элементов всех смежных классов L по ρ . Определим на нем операцию сложения (здесь мы ее обозначим через $+$) следующим образом: $a \oplus b = c$, если $[c]_\rho = [a+b]_\rho$. Вместе с этой операцией множество K является полурешеткой, изоморфной

фактор-полурешетке L/ρ . Известна теорема о гомоморфизме полурешеток как алгебраических систем [2], согласно которой все гомоморфные образы (по нашему - модели) любой заданной полурешетки с точностью до изоморфизма исчерпываются совокупностью всех фактор-полурешеток данной полурешетки по ее различным конгруенциям. В силу этой теоремы K есть модель полурешетки L . Мы называем ее адекватной моделью последней. Конгруенция ρ , по которой она построена, в известном смысле характеризует точность данной модели: чем уже отношение ρ , тем точнее модель. В этой связи мы будем говорить, что K есть адекватная модель полурешетки L с точностью до конгруенции ρ .

Для любых элементов $a, b \in K$ имеет место $(a+b)\rho = (b)\rho \leftrightarrow a+b=b$, поэтому $a \leq b \leftrightarrow a \leq b$, где $\leq$ - отношение порядка в L . Таким образом, отношения порядка в K и в L совпадают. Это не значит, что K - подполурешетка в L , но это значит, в частности, что если L - полурешетка подмножеств или интервалов, то ее адекватная модель является также полурешеткой подмножеств или интервалов соответственно.

Ввиду теоремы о гомоморфизме полурешеток можно утверждать, что любая модель полурешетки L изоморфна некоторой ее адекватной модели. Можно также заметить, что различные адекватные модели полурешетки L характеризуются разной степенью точности, но могут быть изоморфными. Наконец, обратим внимание на свойство транзитивности отношения „быть адекватной моделью“, именно: если K - адекватная модель L с точностью ρ , J - адекватная модель K с точностью σ , то J - адекватная модель L с точностью δ , где $a \delta b \leftrightarrow \exists k_1, k_2 \in K (a \rho k_1, \sigma k_2 \rho b)$.

Если на множестве M задана некоторая эквивалентность $\mathcal{R}$, то на множестве M можно определить эквивалентность $\tilde{\mathcal{R}}$, считая два подмножества эквивалентными (по $\tilde{\mathcal{R}}$), если каждый элемент одного из них эквивалентен (по $\mathcal{R}$) некоторому элементу другого, и наоборот, каждый элемент другого эквивалентен (по $\mathcal{R}$) некоторому элементу первого. Легко проверить, что $\tilde{\mathcal{R}}$ является конгруенцией на полурешетке $\tilde{M}$. Будем называть ее соответствующей заданной эквивалентности $\mathcal{R}$. Пусть далее $\{R\}$ обозначает подполурешетку полурешетки $\tilde{M}$, порожденную множеством смежных классов эквивалентности $\mathcal{R}$.

Теорема 1. Полурешетка $\{R\}$ является адекватной моделью полурешетки $\tilde{M}$ с точностью до конгруенции $\tilde{\mathcal{R}}$.

Доказательство. По определению $\tilde{K}$ между смежными классами эквивалентности $\tilde{K}$ и непустыми подмножествами множества смежных классов эквивалентности K существует взаимно однозначное соответствие, при котором каждый смежный класс эквивалентности $\tilde{K}$ состоит из всех таких подмножеств множества M , которые имеют непустое пересечение с каждым смежным классом соответствующего подмножества смежных классов эквивалентности K и не пересекаются ни с каким другим смежным классом последней. С другой стороны, по определению полурешетки $\{R\}$ между ее элементами и непустыми подмножествами множества смежных классов эквивалентности K существует взаимно однозначное соответствие, при котором каждый элемент полурешетки $\{R\}$ является объединением всех смежных классов соответствующего подмножества смежных классов эквивалентности K . Таким образом, между элементами фактор-полурешетки $\tilde{M}/\tilde{K}$ и элементами полурешетки $\{R\}$ существует взаимно однозначное соответствие, при котором каждый элемент первой состоит из всех таких подмножеств множества M , которые пересекаются с теми и только теми смежными классами эквивалентности K , объединение которых равно соответствующему элементу первой. Легко видеть, что при этом соответствии максимальный элемент любого смежного класса $\tilde{M}$ по $\tilde{K}$ является соответствующим этому классу элементом полурешетки $\{R\}$, т.е. множество максимальных элементов всех смежных классов $\tilde{M}$ по $\tilde{K}$ совпадает с носителем полурешетки $\{R\}$. Сложение в полурешетке $\{R\}$ совпадает с объединением, поэтому для любого соотношения $a + b = c$ в ней с тривиальностью выполняется соотношение $[a \cup b]_{\tilde{K}} = [c]_{\tilde{K}}$ в $\tilde{M}/\tilde{K}$. Таким образом, полурешетка $\{R\}$ действительно удовлетворяет определению адекватной модели полурешетки $\tilde{M}$ с точностью до конгруэнции $\tilde{K}$. Теорема доказана.

Поскольку конгруэнция $\tilde{K}$ определяется однозначно соответствующей эквивалентностью K , полурешетку $\{R\}$ будем называть далее моделью полурешетки $\tilde{M}$ по (или с точностью до) эквивалентности K .

В случае, когда M -декартово произведение некоторых непустых множеств, можно ввести в рассмотрение множество $\langle R \rangle = \tilde{M} \cdot \{R\}$. Его элементами являются всевозможные интервалы на $\tilde{M}$, совпадающие с объединениями некоторых смежных классов эквивалентности K . Мы хотим показать, что вместе с отношением включения это множество образует полурешетку, адекватно моделирующую полурешетку $\tilde{M}$.

С этой целью к совокупности S_0 смежных классов эквивалентности K , рассматриваемых как подмножества одноэлементных интервалов на M и называемых далее начальными подмножествами, применим

следующие элементарные операции:

- 1) образование суммы пары (не обязательно различных) подмножеств, определяемой как сумма подмножеств элементов полурешетки $\hat{M}$;
- 2) объединение пары пересекающихся подмножеств.

Первая операция заключается в присоединении к имеющейся совокупности суммы некоторых ее членов, вторая - в замещении пересекающихся подмножеств их объединением. Будем применять данные операции в произвольном порядке до тех пор, пока не будет достигнута совокупность подмножеств S_n , в которой все подмножества попарно не пересекаются и сумма любых двух из них, в том числе и одинаковых, входит в некоторое подмножество этой совокупности. По построению в подмножествах в S_n присутствуют всевозможные суммы элементов, принадлежащих подмножествам в S_0 , т.е. всевозможные суммы одноэлементных интервалов на M , поэтому их объединение совпадает с $\hat{M}$. Таким образом, S_n есть разбиение множества $\hat{M}$, а задаваемая им эквивалентность есть конгруэнция на полурешетке $\hat{M}$. Будем называть ее порожденной данной эквивалентностью R и обозначать через $\hat{R}$. Фактически это есть минимальная конгруэнция на $\hat{M}$ такая, что для любого подмножества $A \in S_0$ существует смежный класс этой конгруэнции, целиком содержащий A . Ее каждый смежный класс является объединением некоторых сумм начальных подмножеств (смежных классов эквивалентности R), причем начальное подмножество тогда и только тогда является слагаемым какой-либо из таких сумм, когда оно (как подмножество в M) пересекается с максимальным элементом этого класса.

Теорема 2. По отношению включения множество $\langle R \rangle$ является полурешеткой, адекватно моделирующей полурешетку $\hat{M}$ с точностью до конгруэнции $\hat{R}$.

Доказательство. Для доказательства теоремы достаточно убедиться в равенстве $\langle R \rangle = K$, где K - множество максимальных элементов всех смежных классов $\hat{M}$ по $\hat{R}$.

Множество $\langle R \rangle$ состоит из всех интервалов на M , равных объединению некоторых начальных подмножеств. Объединение подмножеств декартова произведения M тогда и только тогда является интервалом, когда оно совпадает с суммой (на $\hat{M}$) всех элементов этих подмножеств. Таким образом, интервал на M является элементом множества $\langle R \rangle$, если и только если он равен сумме всех элементов в объединении некоторых начальных подмножеств. Максимальный элемент α любого смежного класса эквивалентности $\hat{R}$ является суммой всех элементов тех начальных подмножеств, которые являются слагаемыми

в их суммах, объединенных в этот смежный класс, поэтому $\alpha \in \langle R \rangle$. Следовательно, $K \in \langle R \rangle$.

Если объединение некоторых классов эквивалентности R равно их сумме (как подмножеств полурешетки $\hat{M}$) и является, таким образом, интервалом на M , то каждый член смежного класса эквивалентности $\hat{K}$, содержащего в качестве элемента этот интервал, является частью последнего, так как в противном случае он содержит элементы других смежных классов эквивалентности R , что противоречит их непересекаемости. Следовательно, каждый элемент множества $\langle R \rangle$ является максимальным элементом в своем смежном классе $\hat{M}$ по $\hat{K}$, т.е. $\langle R \rangle \subseteq K$.

Теорема доказана.

Теорема 3. Полурешетка $\langle R \rangle$ является адекватной моделью полурешетки $\{R\}$.

Доказательство. Пусть в пределах данного доказательства через $[a]$ обозначается минимальный интервал в $\langle R \rangle$, целиком содержащий подмножество $a \in M$. Рассмотрим отображение $h: \{R\} \rightarrow \langle R \rangle$, при котором $h(a) = [a]$. Ясно, что если $a \in \langle R \rangle$, то $h(a) = [a] = a$. Таким образом, h — сюръективное отображение.

Минимальный интервал в $\langle R \rangle$, целиком содержащий объединение некоторых двух подмножеств из M , является вместе с тем минимальным интервалом в $\langle R \rangle$, содержащим одновременно как минимальный интервал в $\langle R \rangle$, содержащий первое подмножество, так и минимальный интервал в $\langle R \rangle$, содержащий второе подмножество, т.е. $[a \cup b] = [a] \cup [b]$. Результат сложения в полурешетке $\langle R \rangle$ есть минимальный интервал в $\langle R \rangle$, целиком содержащий объединение слагаемых, поэтому $[a] \cup [b] = [a] + [b]$. Сложение в полурешетке $\langle R \rangle$ совпадает с объединением слагаемых, поэтому

$$h(a+b) = [a+b] = [a] + [b].$$

Таким образом, h — гомоморфизм и $\langle R \rangle$ — модель полурешетки $\{R\}$. Множество максимальных элементов всех смежных классов $\{R\}$ по ядерной конгруэнции ρ этого гомоморфизма (определяемой как $a \rho b \leftrightarrow h(a) = h(b)$) равно $\langle R \rangle$. Следовательно, $\langle R \rangle$ — адекватная модель полурешетки $\{R\}$ с точностью ρ . Теорема доказана.

Следствие 1. Полурешетка $\hat{M}$ является адекватной моделью полурешетки $\hat{M}$.

Это есть утверждение теоремы в случае, когда R — тождество. Кроме того, ввиду транзитивности отношения „быть адекватной моделью“, имеет место

Следствие 2. Полурешетка $\langle R \rangle$ является адекватной моделью полурешетки $\tilde{M}$.

Выше мы отмечали, что полурешетки $\tilde{M}$ и $\hat{M}$ нормальные. То же самое можно сказать и про полурешетку $\{R\}$. Заметим также, что подполурешетки и фактор-полурешетки нормальных полурешеток не обязаны быть нормальными. Последнее всецело относится и к полурешетке $\langle R \rangle$, которая по определению изоморфна некоторой фактор-полурешетке полурешетки $\tilde{M}$.

Любой элемент фактор-полурешетки нормальной полурешетки представляется суммой некоторых смежных классов, содержащих минимальные элементы полурешетки, поэтому фактор-полурешетка нормальной полурешетки нормальная, если каждый ее элемент, пересекающийся с базисом полурешетки, является минимальным элементом фактор-полурешетки или, что то же самое, если любые два ее различных элемента, пересекающихся с базисом полурешетки, не находятся между собой в отношении сращения. Применительно к полурешетке $\langle R \rangle$ это предложение можно переформулировать следующим образом.

теорема 4. Полурешетка $\langle R \rangle$ нормальная, если каждый смежный класс эквивалентности R является интервалом на M .

В этом случае минимальными элементами полурешетки $\langle R \rangle$ являются все смежные классы эквивалентности R и только они, а всякий другой ее элемент, будучи интервалом и объединением некоторых минимальных элементов, равен сумме последних.

Относительно нормальных полурешеток имеет место следующее утверждение, аналогичное теореме Стона для булевых алгебр [2].

Теорема 5. Каждая нормальная полурешетка L изоморфна полурешетке подмножеств своего базиса B .

В самом деле, для каждого элемента $a \in L$ определим подмножество $B_a \subseteq B$, состоящее из всех минимальных элементов $x \leq a$. Ввиду нормальности L сумма всех элементов в B_a равна a , причем B_a есть максимальное множество минимальных элементов L , сумма которых есть a , поэтому $B_a = B_b \leftrightarrow a = b$ и $B_a \subseteq B_b \leftrightarrow a \leq b$. Этим доказано, что соответствие, при котором элементу $a \in L$ сопоставляется подмножество $B_a \subseteq B$, является изоморфизмом частично упорядоченного множества L на частично упорядоченное (по включению) множество $\bar{L} = \{B_a : a \in L\}$. Известно, однако, что при изоморфизме двух частично упорядоченных множеств, одно из которых полурешетка, второе множество также полурешетка, и эти полурешетки изоморфны [3]. Таким образом, $\bar{L}$ — полурешетка подмножеств базиса B , изоморфная полурешетке L . Теорема доказана.

Ввиду этой теоремы полурешетки $\tilde{M}$, $\hat{M}$ и $\{K\}$, а также полурешетку $\langle K \rangle$, когда она нормальная, будем представлять далее как полурешетки подмножеств множеств их минимальных элементов.

3. Полурешетки проводимостей и состояний

К рассмотрению полурешеток проводимостей и состояний приводят попытки адекватного описания функционирования схем из элементов с управляемой проводимостью [4-6].

Зафиксируем алфавит символов $E = \{0, 1, X\}$. Его непустые подмножества $\{0\}$, $\{1\}$, $\{X\}$, $\{0, 1\}$, $\{0, X\}$, $\{1, X\}$, $\{0, 1, X\}$ будем называть проводимостями. Отождествляя для простоты выражений одноэлементные подмножества с их элементами, будем обозначать эти проводимости символами $0, 1, X, \Phi, \Theta, \Psi, *$ соответственно. Будем трактовать их как проводимости электрических цепей следующим образом: $0, 1$ и X – проводимости соответственно разомкнутой, замкнутой и резистивной цепей; Φ – проводимость разомкнутой или замкнутой (т.е. не резистивной) цепи; Θ – проводимость разомкнутой и резистивной (т.е. не замкнутой) цепи; Ψ – проводимость замкнутой или резистивной (т.е. не разомкнутой) цепи и $*$ – проводимость произвольной (разомкнутой, замкнутой или резистивной) цепи. Кроме того, условимся считать каждый из символов $0, 1$ и X представителем самого себя и символа Φ , символы 0 и 1 – представителями символа Φ , символы 0 и X – представителями символа Θ и, наконец, символы 1 и X – представителями символа Ψ .

Полурешетки подмножеств множества E , т.е. полурешетки, элементами которых являются проводимости, а отношением порядка – отношение включения, будем называть полурешетками проводимостей. Наибольшая из них состоит из всех семи проводимостей и обозначается далее буквой C . Операция сложения в ней совпадает с теоретико-множественным объединением; эта полурешетка нормальная: в ней $0, 1$ и X – минимальные элементы, $\Phi = 0+1$, $\Theta = 0+X$, $\Psi = 1+X$ и $*$ – $0+1+X$. Все другие полурешетки проводимостей являются подмножествами (не обязательно подполурешетками) полурешетки C . Среди них имеются и не нормальные, как, например, полурешетка $\{0, \Phi\}$. Примером нормальной полурешетки проводимостей, не являющейся подполурешеткой в C , может служить полурешетка $\{0, 1, X, \Phi\}$, в которой сумма любых двух и более различных элементов равна $*$.

Пусть заданы натуральное число $z \geq 1$ и некоторое основное множество $M = E_1 \times \dots \times E_z$, представляющее собой декартово произведение

2 непустых подмножеств множества E , т.е. 2 -мерный интервал над E или, что одно и то же, элемент множества C^2 - 2 -й декартовой степени алфавита $C = \{0, 1, X, 0, \Phi, X, \Phi\}$. Непустые подмножества множества M будем называть состояниями (на M), а те из них, которые сами являются интервалами и принадлежат, следовательно, множеству C^2 , - интервальными состояниями, или состояниями в интервальной форме. Число 2 называется размерностью состояния.

Состояния в нашем определении - это дискретные модели физических состояний узлов в электрических схемах с 2 -полюсным источником питания, в которых проводимости цепей от j -го полюса питания до любого заданного узла принадлежат множеству E_j , где E_j - j -я компонента основного множества M . Они представляют собой подмножества наборов проводимостей со значениями в E , которые возможны в схеме от полюсов питания до данного узла. Так, в схеме на рис. 1 с двухполюсным питанием состояние p узла p определяется проводимостями a и b и моделируется интервалом $ab \in C^2$. В частности, интервалами $10, 01, 00, X0$ и $0X$ моделируются такие физические состояния этого узла, которые обычно отождествляются с логическими нулем, единицей, высоким импедансом, слабым нулем и слабой единицей соответственно. Отсюда легко понять, что подмножеством $\{10, 01\}$ представляется состояние "логический нуль" или "логическая единица", интервалом 00 - состояние "логический нуль" или "высокий импеданс" и т.д. Состояние q узла q в данной схеме зависит не только от переменных a и b , но и от переменной c . Например, если $c = 0$, то $q = 00$ - высокий импеданс; если $c = 1$, то $q = p$; если же $c = \Phi$ и $ab = 1X$, то $q = \{00, 1X\}$ и т.д.

Полурешетки подмножеств множества M , т.е. полурешетки, элементами которых являются состояния, а отношением порядка отношение включения, будем называть полурешетками состояний. Размерность состояний в полурешетке называется также размерностью самой полурешетки. Приведем несколько примеров полурешеток состояний. Во всех этих примерах предполагается, что $2 = 2$. Состояния, обозначаемые в них символами $0, 1, Z, \delta$ и $\bar{1}$, можно трактовать как логический нуль, единица, высокий импеданс, слабый нуль и слабая единица соответственно, если по-прежнему в стоящих за ними парах проводимостей первую компоненту рассматривать как проводимость от земли, а вторую - от полюса питания с высоким потенциалом и высокий потенциал отождествлять с логической единицей, низкий - с логическим нулем, отсутствие потенциала - с высоким импедансом, а резистор считать, как это принято, например в [6], аттенуатором, ослаб-

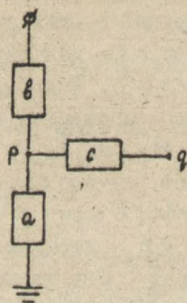


Рис. 1

Таблица 1

x	0	1	X	⊖	⊗	*	⊗
$\bar{x}$	1	0	X	⊖	*	⊗	⊗

Таблица 2

x	0	0	0	0	0	0	1	1	1	1	1	X	X	X	X	⊖	⊖	⊖	⊖	*
y	1	X	⊖	⊗	*	⊗	X	⊖	⊗	*	⊗	⊖	⊗	*	⊗	⊖	*	⊗	*	⊗
$x \wedge y$	0	0	0	0	0	0	X	⊖	⊗	*	⊗	⊗	⊗	X	⊗	⊗	⊗	⊗	⊗	⊗
$x \vee y$	1	X	⊖	⊗	*	⊗	1	1	1	1	1	*	X	*	X	⊗	*	⊗	*	*

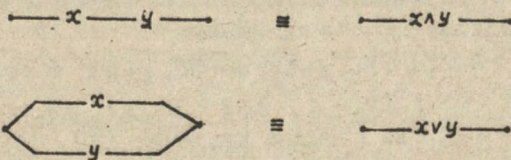


Рис. 2

6) $M = \otimes \otimes$ и классами R на M являются: $\{10, 1x\}$, $\{01, x1\}$, $\{xx\}$, $\{0x\}$, $\{x0\}$, $\{00\}$, $\{11\}$. В этом случае полурешётку $\langle R \rangle$ образуют интервалы: $\underline{0} = 10$, $\underline{1} = 01$, $\tilde{0} = x0$, $\tilde{1} = 0x$, $z = 00$, $u = 11$, $x = xx$, $1\otimes = \{\underline{0}, u\}$, $\otimes 1 = \{\underline{1}, u\}$, $\otimes\otimes = \{\tilde{0}, \tilde{1}, z, x\}$, $\otimes 0 = \{\tilde{0}, z\}$, $0\otimes = \{\tilde{1}, z\}$, $x\otimes = \{\tilde{0}, x\}$, $\otimes x = \{\tilde{1}, x\}$, $\otimes x = \{\underline{1}, \tilde{1}, x\}$, $\otimes\otimes = \{\underline{0}, \tilde{1}, z\}$, $\otimes\otimes = \{\tilde{0}, \underline{1}, z\}$, $x\otimes = \{\underline{0}, \tilde{0}, x\}$, $\otimes\otimes = \{\tilde{1}, z, x, \underline{0}, \tilde{0}\}$, $\otimes\otimes = \{\tilde{0}, \underline{1}, \tilde{1}, z, x\}$, $A = \{\underline{0}, \underline{1}, \tilde{0}, \tilde{1}, z, u, x\}$

4. Функции

Мы рассматриваем функции, отображающие некоторую полурешетку $\mathcal{D}$ в некоторую полурешетку V . Будем называть их полурешётчными функциями. Полурешётка $\mathcal{D}$ называется областью определения, полурешетка V — областью значений рассматриваемых функций. Для каждой конкретной функции f они обозначаются через $\mathcal{D}_f$ и V_f соответственно. В частности, если f зависит от n переменных со значениями в полурешетке L , то $\mathcal{D}_f$ есть n -я декартова степень полурешетки L , т.е. полурешетка L^n .

Множество всех функций со значениями в полурешетке V от конечного числа переменных со значениями в полурешетке L обозначается через P_{LV} . Вместо P_{LL} пишем P_L . Функции в P_L называются функциями на полурешетке L . Функции на полурешетках проводимостей или состояний называются функциями проводимости или состояний соответственно. Таким образом, функция проводимости — это функция, которая вместе со своими переменными принимает значения в алфавите S . Если области $\mathcal{D}_f$ и V_f являются полурешетками интервалов, то функция f называется интервальной функцией. Если элементы полурешеток $\mathcal{D}_f$ и V_f суть интервалы над некоторым алфавитом, то f называется интервальной функцией над этим алфавитом. По определению, функции проводимости и состояний интервальные, причем функции проводимости — над алфавитом E , а функции состояний — над алфавитом E^2 для подходящего $\tau \geq 1$. Если L — полурешетка состояний и V — полурешетка проводимостей, то функции в P_{LV} называются функциями проводимости в V от состояний в L , и наоборот, если L — полурешетка проводимостей, а V — полурешетка состояний, то функции в P_{LV} называются функциями состояний в V от проводимостей в L . Наконец, переменные со значениями в полурешетках проводимостей или состояний называются переменными проводимости или состояний соответственно.

В табл. I и 2 приведены примеры элементарных функций проводимости — логических отрицания $\neg$, конъюнкции $\wedge$ и дизъюнкции $\vee$.



Последние две функции определены как проводимости соответственно последовательно и параллельно соединенных электрических цепей (см. рис. 2). В табл. 2 они даны с учетом их свойств идемпотентности ($x \wedge x = x$, $x \vee x = x$) и коммутативности ($x \wedge y = y \wedge x$, $x \vee y = y \vee x$).

В дальнейшем нам понадобятся выражения для конъюнкции произвольной проводимости и произвольного состояния и для дизъюнкции любых равноразмерных состояний. Сейчас мы их введём.

Пусть $\alpha \in E^z$, $\beta \in E^z$, $\alpha = \alpha_1 \alpha_2 \dots \alpha_z \in E^z$, $\beta = \beta_1 \beta_2 \dots \beta_z \in E^z$, $e \in E$ и $c \in C$. Положим $e \wedge \alpha = e_1 e_2 \dots e_z$, где $e_i = e \wedge \alpha_i$, и $\alpha \vee \beta = e_1 e_2 \dots e_z$, где $e_i = \alpha_i \vee \beta_i$, для каждого $i = 1, \dots, z$. Определим

$$c \wedge \alpha = \{e \wedge \alpha : e \in c, \alpha \in \alpha\}, \quad \alpha \vee \beta = \{\alpha \vee \beta : \alpha \in \alpha, \beta \in \beta\}.$$

Заметим, что каждая из этих операций коммутативна и ассоциативна, и их результатом являются состояния той же размерности, что и размерность участвующих в них состояний. Обратим внимание также на тот легко проверяемый факт, что свойство интервальности состояния в конъюнкции с проводимостью может не сохраняться (например, $\emptyset \wedge 1x = \{00, 1x\}$), в то время как в дизъюнкции состояний это свойство сохраняется всегда. Иначе говоря, если $\alpha \in C^z$, $\beta \in C^z$, то $\alpha \vee \beta \in C^z$, но, возможно, $c \wedge \alpha \notin C^z$. Более того, если $\alpha = \alpha_1 \alpha_2 \dots \alpha_z \in C^z$ и $\beta = \beta_1 \beta_2 \dots \beta_z \in C^z$, то $\alpha \vee \beta = \gamma_1 \gamma_2 \dots \gamma_z$, где $\gamma_i = \alpha_i \vee \beta_i$, и $c \wedge \alpha \in c \otimes \alpha = \gamma_1 \gamma_2 \dots \gamma_z$, где $\gamma_i = c \wedge \alpha_i$, для всех $i = 1, \dots, z$. Введённая последним соотношением операция $\otimes$ называется покомпонентной конъюнкцией (проводимости и интервального состояния). В отличие от $\wedge$, её результатом всегда является интервальное состояние.

Для любой пары полурешёточных функций f и g таких, что $\mathcal{D}_f \subseteq \mathcal{D}_g$, естественным образом определяются отношение порядка и операция сложения:

$$f \leq g \leftrightarrow \forall \alpha \in \mathcal{D}_f (f(\alpha) \leq g(\alpha)),$$

$$f + g = h \leftrightarrow (\mathcal{D}_h = \mathcal{D}_f) \wedge \forall \alpha \in \mathcal{D}_h (h(\alpha) = f(\alpha) + g(\alpha)).$$

Полурешёточная функция f называется аддитивной, если $f(\alpha) = \sum f(\beta)$, где

$$\beta \in B_f(\alpha) \leftrightarrow \beta \leq \alpha \wedge \beta \in B_f$$

и B_f — базис полурешетки $\mathcal{D}_f$. Любая аддитивная функция полностью задается своими значениями на минимальных элементах области определения; на любом другом элементе α последней ее значение совпадает с суммой значений на минимальных элементах, предшествующих α .

В соответствии с этим любую абстрактную (с абстрактными областями определения M и значений V) функцию можно рассматри-

вать как аддитивную полурешёточную функцию, у которой $V_f = \tilde{V}$ и $\mathcal{D}_f$ - некоторая полурешетка подмножеств множества M с базисом M .

Полурешёточная функция f называется монотонной, если из $\alpha \leq \beta$ следует $f(\alpha) \leq f(\beta)$. Говорят, что функция g реализует функцию f , если $g \leq f$. Функция f называется квазимоноотонной, если она реализуется монотонной функцией.

Теорема 6. Всякая аддитивная функция f монотонна. Суперпозиция монотонных или квазимоноотонных функций в P_A есть функция соответственно монотонная или квазимоноотонная.

В самом деле, если $\alpha \leq \beta$, то $V_f(\alpha) \subseteq V_f(\beta)$ и, по определению аддитивной функции, $f(\alpha) \leq f(\beta)$, т.е. функция f монотонная. Если $f = f_0(f_1, \dots, f_m)$ и функции $f_0, f_1, \dots, f_m$ монотонные, то

$$\alpha \leq \beta \rightarrow f_1(\alpha) \dots f_m(\alpha) \leq f_1(\beta) \dots f_m(\beta) \leq f_0(f_1(\alpha) \dots f_m(\alpha)) \leq f_0(f_1(\beta) \dots f_m(\beta)) \rightarrow f(\alpha) \leq f(\beta),$$

т.е. функция f монотонная. Наконец, если $f = f_0(f_1, \dots, f_m)$ и функции $f_0, f_1, \dots, f_m$ реализуются монотонными функциями $g_0, g_1, \dots, g_m$ соответственно, то функция f реализуется монотонной функцией $g = g_0(g_1, \dots, g_m)$. Теорема 6 доказана.

Суперпозиция аддитивных функций может не быть аддитивной функцией. Так, функции $x, \bar{x}$ и $x \vee y$ аддитивные, в то время как функция $x \vee \bar{x}$ таковой не является. Вместе с тем из теоремы 6 вытекает

Следствие. Суперпозиция аддитивных функций является монотонной функцией.

Теорема 7 (критерий квазимоноотонности). Полурешёточная функция f квазимоноотонна, если и только если для любого k -элементного подмножества $K \in \mathcal{D}_f$, где $2 \leq k \leq |V_f|$, из существования нижней грани множества K следует существованию нижней грани множества всех значений функции f на элементах в K .

Доказательство. Необходимость. Пусть монотонная функция g реализует функцию f и α - нижняя грань множества элементов $\alpha_1, \dots, \alpha_k \in \mathcal{D}_f$. Тогда $g(\alpha) \leq g(\alpha_i) \leq f(\alpha_i)$ для каждого $i = 1, \dots, k$, поэтому $g(\alpha)$ есть нижняя грань множества значений $f(\alpha_1), \dots, f(\alpha_k)$.

Достаточность. Рассмотрим произвольный элемент $\alpha \in \mathcal{D}_f$ и все элементы $\alpha_1, \dots, \alpha_n \in \mathcal{D}_f$, которые следуют за α . Среди последних число k -элементов с различными значениями функций f не превосходит $|V_f|$. Элемент α является их общей нижней гранью, по-

этому ввиду условия теоремы множество $M_\alpha = \{f(\alpha_1), \dots, f(\alpha_n)\}$ имеет нижнюю грань. Примем за значение функции g на элементе α наибольшую из нижних граней множества M_α . (Она равна сумме всех таких граней). Поскольку $\alpha \in \{\alpha_1, \dots, \alpha_n\}$, то $f(\alpha) \in M_\alpha$, поэтому $g(\alpha) \leq f(\alpha)$, т.е. функция g реализует функцию f . Кроме того, если $\alpha \leq \beta$ и $\beta_1, \dots, \beta_m$ - все те элементы в $\mathcal{D}_f$, которые следуют за β , то $\{\beta_1, \dots, \beta_m\} \subseteq \{\alpha_1, \dots, \alpha_n\}$ и $M_\beta \subseteq M_\alpha$, поэтому $g(\alpha) \leq g(\beta)$, т.е. функция g монотонная. Теорема 7 доказана.

Следствие. Функция проводимости квазимоноотонна, если и только если ее значения на любых двух или трех пересекающихся элементах из области определения пересекаются.

Функция g называется адекватной моделью функции f , если полурешетки $\mathcal{D}_g$ и V_g являются адекватными моделями полурешеток $\mathcal{D}_f$ и V_f соответственно и для любого $\alpha \in \mathcal{D}_g$ значение $g(\alpha)$ есть максимальный элемент в смежном классе $[f(\alpha)]_\rho$, где ρ - конгруенция на V_f , с точностью до которой V_g адекватно моделирует V_f . Пусть далее f^a обозначает адекватную модель функции f . По определению $f^a(\alpha) \geq f(\alpha)$ для любого $\alpha \in \mathcal{D}_f$.

Говорят, что функция f сохраняет пару бинарных отношений (σ, ρ) , где $\sigma \in \mathcal{D}_f^2$ и $\rho \in V_f^2$, если из $\alpha \sigma \beta$ следует $f(\alpha) \rho f(\beta)$. Если области значений переменных и функции f совпадают, т.е. $\mathcal{D}_f = V_f^n$ для некоторого $n \geq 1$, а σ есть результат покомпонентного распространения ρ на $\mathcal{D}_f$, т.е. $(\alpha_1, \dots, \alpha_n) \sigma (\beta_1, \dots, \beta_n) \Leftrightarrow \forall i = 1, \dots, n (\alpha_i \rho \beta_i)$, то в случае сохранения функцией f пары (σ, ρ) говорят, что она сохраняет отношение ρ ; в этом случае говорят также, что ρ стабильно относительно функции f [2].

Теорема 8. 1. Если функция f аддитивная, монотонная или квазимоноотонная, то функция f^a также аддитивная, монотонная или квазимоноотонная соответственно. 2. Пусть $f = f_0(f_1, \dots, f_m)$ и $g = g_0(g_1, \dots, g_m)$. Тогда если функция f_i монотонная, то $f_i^a \leq g_i$; если же функция f_0 сохраняет пару конгруенций (σ, ρ) , где σ - точность, с которой $\mathcal{D}_{f_i^a}$ адекватно моделирует $\mathcal{D}_{f_i}$, а ρ - точность, с которой $V_{g_i^a}$ адекватно моделирует V_{g_i} , то $f^a = g$.

Доказательство. 1. Пусть ρ - конгруенция адекватного моделирования полурешетки V_f полурешеткой V_{f^a} и $\alpha, \beta \in \mathcal{D}_{f^a}$. Если $f(\alpha + \beta) = f(\alpha) + f(\beta)$, то $f^a(\alpha + \beta) = \max [f(\alpha) + f(\beta)]_\rho = \max ([f(\alpha)]_\rho + [f(\beta)]_\rho) = \max [f(\alpha)]_\rho + \max [f(\beta)]_\rho = f^a(\alpha) + f^a(\beta)$. Этим доказана аддитивность f^a при условии аддитивности f .

Далее используется следующий легко проверяемый факт: если $[a]_\rho \leq [b]_\rho$, то $\max [a]_\rho \leq \max [b]_\rho$.

Если функция f монотонная и $\alpha \leq \beta$, то $f(\alpha) \leq f(\beta)$, $[f(\alpha)]_p \leq [f(\beta)]_p$, $\max [f(\alpha)]_p \leq \max [f(\beta)]_p$, $f^a(\alpha) \leq f^a(\beta)$, т.е. функция f монотонная.

Если $f(\alpha) \leq f(\beta)$, то $[f(\alpha)]_p \leq [g(\alpha)]_p$, $\max [f(\alpha)]_p \leq \max [g(\alpha)]_p$ и $f^a(\alpha) \leq g^a(\alpha)$, что ввиду утверждения о монотонности доказывает квазимонотонность f^a при условии квазимонотонности f .

$$2. f^a(\alpha) = \max [f(\alpha)]_p = \max [f_0(f_1(\alpha) \dots f_m(\alpha))]_p = f_0^a(f_1^a(\alpha) \dots f_m^a(\alpha)).$$

Если функция f_0 монотонная, то ввиду $f_i(\alpha) \leq f_i^a(\alpha)$ для каждого $i = 1, \dots, n$

$$f_0^a(f_1(\alpha) \dots f_m(\alpha)) \leq f_0^a(f_1^a(\alpha) \dots f_m^a(\alpha)) = g^a(\alpha) \quad \text{и,}$$

следовательно, $f^a \leq g$; если же f_0 сохраняет пару (σ, ρ) , то ввиду $(f_1(\alpha) \dots f_m(\alpha)) \sigma (f_1^a(\alpha) \dots f_m^a(\alpha))$ имеет место

$$f_0(f_1(\alpha) \dots f_m(\alpha)) \rho f_0(f_1^a(\alpha) \dots f_m^a(\alpha)), \text{ откуда } f_0^a(f_1(\alpha) \dots f_m(\alpha)) = \max [f_0(f_1^a(\alpha) \dots f_m^a(\alpha))]_p = f_0^a(f_1^a(\alpha) \dots f_m^a(\alpha)) = g^a(\alpha) \text{ и,}$$

следовательно, $f^a = g$. Теорема доказана.

5. Автоматы

Будем рассматривать автоматы, в которых каждый из алфавитов (входной, выходной и внутренний) является некоторой полурешеткой. Будем называть их полурешеточными автоматами. В любом таком автомате $A = (X, Q, Y, \psi, \varphi)$ с входным алфавитом X , выходным алфавитом Y и внутренним алфавитом Q функция переходов ψ принадлежит множеству $R_{X \times Q, Q}$, а функция выходов φ — множеству $R_{X \times Q, Y}$. Автомат A называется аддитивным, монотонным или квазимонотонным, если его функции ψ и φ соответственно аддитивные, монотонные или квазимонотонные. Автомат A называется интервальным, если его алфавиты X, Q и Y являются полурешетками интервалов. Если, кроме того, каждая компонента последних принадлежит некоторой полурешетке состояний или проводимостей, то автомат A называется переключательным.

Состояние (входное, выходное или внутреннее) переключательного автомата называется высокоимпедансным, если любая компонента в нём имеет вид 0^z для некоторого $z \geq 1$. Входные, выходные и внутренние состояния (элементы в X, Q и Y) интервального автомата A являются векторами некоторых размерностей n, m и k . Его функции переходов и выходов также векторные: $\psi = (\psi_1 \dots \psi_m)$, $\varphi = (\varphi_1 \dots \varphi_m)$. Система уравнений вида

$$y_i = \varphi_i(x_1, \dots, x_m, z_1, \dots, z_l), \quad i = 1, \dots, m;$$

$$z_j = \psi_j(x_1, \dots, x_m, z_1, \dots, z_l), \quad j = 1, \dots, k$$

называется канонической системой уравнений интервального автомата A . В них переменные x , y и z принимают значения соответствующих компонент соответственно входного, выходного и внутреннего состояний автомата A и называются его входными, выходными и внутренними переменными.

С каждым автоматом A связываются два способа функционирования – синхронный и асинхронный. Первый заключается в отображении

$$f : X \times Q \rightarrow Q \times Y,$$

где $f(x, q) = sy$, если $s = \psi(x, q)$ и $y = \varphi(x, q)$; определение второго дается ниже.

Пусть всюду далее для любой полурешётчатой функции f и любого $\alpha \in \mathcal{D}_f$

$$f^+(\alpha) = \sum_{\rho \in \mathcal{D}_f, \rho \leq \alpha} f(\rho).$$

Обратим внимание на монотонность функции f^+ . Кроме того, если функция f монотонная, то $f^+ = f$.

Пусть $xq \in X \times Q$. Рассмотрим последовательность состояний $q(1), q(2), \dots$, где $q(1) = q$ и $q(t+1) = q(t) + \psi^+(x, q(t))$, $t \geq 1$. В ней $q(1) \leq q(2) \leq \dots$, поэтому существует натуральное $\ell \geq 1$ такое, что $q(\ell) = q(\ell+1) = \dots$. Рассмотрим последовательность состояний $p(1), p(2), \dots$, где $p(1) = q(\ell)$ и $p(t+1) = \psi^+(x, p(t))$, $t \geq 1$. Ввиду $q(\ell) = q(\ell+1)$ в ней $p(1) \geq p(2)$ и, следовательно, ввиду монотонности ψ^+ $p(1) \geq p(2) \geq p(3) \geq \dots$. Поэтому существует натуральное $z \geq 1$ такое, что $p(z) = p(z+1) = \dots$. Положим по определению $\psi^\ell(x, q) = p(z)$. Непосредственно проверяется монотонность определенной так функции $\psi^\ell : X \times Q \rightarrow Q$. Заметим также, что в частном случае, когда $\psi^+(x, q) \geq q$ или $\psi^+(x, q) \leq q$, вычисление $\psi^\ell(x, q)$ в данном определении сводится к одной процедуре: $\psi^\ell(x, q) = q(s)$ для наименьшего s такого, что $q(s) = q(s+1)$, где $q(1) = q$ и $q(t+1) = \psi^+(x, q(t))$, $t \geq 1$.

Если же функция ψ монотонная, то в любом случае $\psi^+ = \psi$. Асинхронным функционированием автомата A называется отображение

$$f : X \times Q \rightarrow Q \times Y,$$

где $f(x, q) = sy$, если $s = \psi^\ell(x, q)$ и $y = \varphi(x, s)$.

Полурешётчатый автомат A , в котором алфавиты являются полурешётками подмножеств некоторых множеств, в частности любой интервальный автомат A мы рассматриваем как функциональную динамическую модель некоторого класса реальных устройств, в которых

Таблица 3

	1	2	3	4	12	13	14	23	24	34	123	124	134	234	1234
00	1,0	2,0	1,0	2,0	12,0	1,0	12,0	12,0	2,0	12,0	12,0	12,0	12,0	12,0	12,0
01	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0
10	1,0	3,1	3,1	4,0	13,0	13,0	14,0	3,1	34,0	34,0	13,0	134,0	134,0	34,	134,0
11	4,0	3,1	3,1	4,0	34,0	34,0	4,0	3,1	34,0	34,0	34,0	34,0	34,0	34,	34,0
00	12,0	2,0	12,0	2,0	12,0	12,0	12,0	12,0	2,0	12,0	12,0	12,0	12,0	12,0	12,0
00	1,0	23,0	13,0	24,0	123,0	13,0	124,0	123,0	234,0	1234,0	123,0	1234,0	1234,0	1234,	1234,0
10	14,0	3,1	3,1	4,0	134,0	134,0	14,0	3,1	34,0	34,0	134,0	134,0	134,0	34,	134,0
01	24,0	23,0	23,0	24,0	234,0	234,0	24,0	23,0	234,0	234,0	234,0	234,0	234,0	234,	234,0
00	124,0	23,0	123,0	24,0	1234,0	1234,0	124,0	123,0	234,0	1234,0	1234,0	1234,0	1234,0	1234,	1234,0

a

	1	2	3	4	12	13	14	23	24	34	123	124	134	234	1234
00	1,0	2,0	1,0	2,0	12,0	1,0	12,0	12,0	2,0	12,0	12,0	12,0	12,0	12,0	12,0
01	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0
10	1,0	3,1	3,1	4,0	13,0	13,0	14,0	3,1	34,0	34,0	13,0	134,0	134,0	34,0	134,0
11	4,0	3,1	3,1	4,0	34,0	34,0	4,0	3,1	34,0	34,0	34,0	34,0	34,0	34,0	34,0
00	12,0	2,0	12,0	2,0	12,0	12,0	12,0	12,0	2,0	12,0	12,0	12,0	12,0	12,0	12,0
00	1,0	123,0	13,0	1234,0	123,0	13,0	1234,0	123,0	1234,0	1234,0	123,0	1234,0	1234,0	1234,0	1234,0
10	14,0	3,1	3,1	4,0	134,0	134,0	14,0	3,1	34,0	34,0	134,0	134,0	134,0	34,0	134,0
01	234,0	23,0	23,0	234,0	234,0	234,0	234,0	23,0	234,0	234,0	234,0	234,0	234,0	234,0	234,0
00	1234,0	1234,0	1234,0	1234,0	1234,0	1234,0	1234,0	1234,0	1234,0	1234,0	1234,0	1234,0	1234,0	1234,0	1234,0

b

входные, выходные и внутренние состояния являются элементами подмножеств, представленных символами алфавитов X , Y и Q соответственно. В его асинхронном функционировании $f(x, q) = sy$ состояние $s = \varphi^{\varepsilon}(x, q)$ обозначает подмножество внутренних состояний устройства, в которых оно может оказаться в результате произвольного изменения входного состояния в пределах подмножества x , т.е. после (сколь угодно длительного) воздействия на него (в любом порядке) входных состояний из x , если до этого его внутреннее состояние принадлежало подмножеству q , а выходной символ $y = \varphi^{+}(x, q)$ обозначает область изменения выходного состояния устройства, вызванного указанным изменением входного состояния при начальных внутренних состояниях в подмножестве q .

Любой конечный автомат можно рассматривать как интервальный полурешеточный автомат. В частности автомат, функции которого определены на декартовом произведении абстрактных множеств, трактуется как аддитивный интервальный автомат, функции которого определены на полурешетке интервалов в этом декартовом произведении: значения их компонент на произвольном таком интервале вычисляются по правилу аддитивности — путем объединения значений на элементах, входящих в данный интервал. Применительно к нему наше определение асинхронного функционирования соответствует общепринятому понятию поведения асинхронного автомата [II-13]. Табл.3, а и б демонстрируют аддитивный интервальный автомат, полученный указанным образом из абстрактного автомата, заданного в [12, табл. I.3], и его асинхронное функционирование f . В них интервал внутренних состояний обозначен набором входящих в него элементов и $\Phi = \{0, 1\}$.

Пусть f и f' — асинхронные функционирования полурешеточных автоматов A и $A' = (X, Q', Y, \varphi', \varphi')$ соответственно. Говорят, что автомат A (асинхронно) реализуется автоматом A' , если существует отображение $h: Q \rightarrow Q'$ такое, что для любой пары $q \in X \times Q$ из $f(x, q) = sy$ следует $f'(x, h(q)) = h(s)y'$ для некоторого $y' \in Y$.

Для полурешеточного автомата A , внутренний алфавит которого является декартовым произведением некоторых полурешеток, ставится задача минимизации размерности внутреннего состояния как задача построения такого автомата A' с минимальным числом внутренних переменных, который асинхронно реализует данный автомат A . Задача решается с помощью следующих трёх операций.

1. Отождествление неразличимых переменных. Внутренние переменные называются неразличимыми, если правые части канонических уравнений для них совпадают. Отождествление таких переменных состоит в том, что во всех уравнениях канонической системы они за-

меняются некоторой одной из них, а получающиеся при этом одинаковые уравнения исключаются.

2. Исключение недостижимых переменных. Пусть $\mathcal{U}$ - множество всех внутренних переменных, входящих (существенно) в правые части канонических уравнений автомата. Переменные в $\mathcal{U}$ называются переменными памяти. Определим подмножество $\mathcal{U}_0 \in \mathcal{U}$ с помощью следующего индуктивного определения:

а) переменные в $\mathcal{U}$ из уравнений для выходных переменных принадлежат $\mathcal{U}_0$;

б) если $u \in \mathcal{U}_0$, то переменные в $\mathcal{U}$ из уравнения для u также принадлежат $\mathcal{U}_0$;

в) других переменных в $\mathcal{U}_0$ нет.

Переменные в $\mathcal{U}_0$ называются достижимыми, а внутренние переменные, не принадлежащие $\mathcal{U}_0$, - недостижимыми (от выходных переменных). Исключение недостижимых переменных состоит в исключении из канонической системы всех уравнений для этих переменных.

3. Операция суперпозиции. Построим граф G зависимостей переменных в $\mathcal{U}_0$: вершины графа G суть переменные в $\mathcal{U}_0$, и пара переменных u, v есть дуга в G , если v входит (существенно) в уравнение для u (т.е. u зависит от v). Подмножество вершин графа G назовем его замыкающим множеством, если каждый контур данного графа содержит хотя бы одну из этих вершин. Подмножество переменных памяти автомата, включающее в себя все вершины некоторого замыкающего множества графа G , называется замыкающим множеством данного автомата. Замыкающее множество минимальной мощности называется наименьшим. Зафиксировав некоторое замыкающее множество переменных Z автомата, мы можем, подставляя в правые части канонических уравнения вместо остальных переменных памяти их выражения из соответствующих им уравнений, выразить все выходные и внутренние переменные автомата через его входные переменные и переменные памяти в множестве Z . После этого всякое уравнение для внутренней переменной, не принадлежащей Z , может быть исключено из канонической системы как уравнение для переменной, недостижимой от входных переменных. Выполненное таким образом преобразование системы уравнений автомата назовем суперпозицией по замыкающему множеству Z . Суперпозиции по наименьшим замыкающим множествам приводят к системам с минимальным числом канонических уравнений. Требуемые для них замыкающие множества графа G можно получить как кратчайшие столбцовые покрытия матрицы контуров этого графа. Действительные и эвристические алгоритмы их построения можно найти, например в [9,10].

Тот факт, что данные операции сохраняют асинхронную реализацию заданного автомата, утверждается в следующей теореме.

Теорема 9. Пусть каноническая система уравнений автомата A' получена из канонической системы уравнений автомата A при помощи операций отождествления неразличимых переменных, исключения недостижимых переменных и суперпозиции. Пусть также f' и f суть асинхронные функционирования автоматов A' и A соответственно и для любого набора q значений внутренних переменных автомата A символ q' обозначает набор значений внутренних переменных автомата A' , указанных в q . Тогда если $f(x, q) = sy$, то $f'(x, q') = s'y'$, где $y' \leq y$ и, если автомат A монотонный, то $y' = y$.

Доказательство. Установим утверждение теоремы относительно операции суперпозиции, так как утверждение относительно других операций очевидно. Пусть без ограничения общности канонические системы уравнений автоматов A и A' имеют следующий вид:

$$\begin{aligned} \text{I. } y &= \varphi(x, z_1, z_2), & \text{II. } y &= \delta(x, z_1) \equiv \varphi(x, z_1, \psi_2(x, z_1)), \\ z_1 &= \psi_1(x, z_2), & z_1 &= \lambda(x, z_1) \equiv \psi_1(x, \psi_2(x, z_1)). \\ z_2 &= \psi_2(x, z_1). \end{aligned}$$

Предположим, что для $q = q_1, q_2$, где q_1 и q_2 — значения переменных z_1 и z_2 соответственно, имеет место $q = q + \varphi^+(x, q)$, т.е.

$$q_1 = q_1 + \psi_1^+(x, q_2), \quad q_2 = q_2 + \psi_2^+(x, q_1).$$

Тогда $q_1 \geq \psi_1^+(x, q_2)$ и $q_2 \geq \psi_2^+(x, q_1)$, откуда, ввиду монотонности ψ_1^+ , можно записать

$$q_1 \geq \psi_1^+(x, \psi_2^+(x, q_1)) \geq (\psi_1(x, \psi_2(x, q_1)))^+ = \lambda^+(x, q_1)$$

и, следовательно, $q_1 = q_1 + \lambda^+(x, q_1)$. Аналогичным образом показывается, что если $q = \varphi^+(x, q)$, то $q_1 = \lambda^+(x, q_1)$. Используя эти факты в процедурах вычисления $\psi^{\varepsilon}(x, q)$ и $\lambda^{\varepsilon}(x, q')$, можно сделать следующие заключения: если $q(\varepsilon) = q(\varepsilon+1) = q(\varepsilon) + \varphi^+(x, q(\varepsilon))$, то $q'(\varepsilon) = q'(\varepsilon+1) = q'(\varepsilon) + \lambda^+(x, q'(\varepsilon))$, и если $p(z+1) = p(z) = \varphi^+(x, p(z))$, то $p'(z+1) = p'(z) = \lambda^+(x, p'(z))$. Из них по определению $\psi^{\varepsilon}(x, q)$ и $\lambda^{\varepsilon}(x, q')$ (как величин, равных $p(z)$ и $p'(z)$ соответственно) следует, что если $\psi^{\varepsilon}(x, q) = s$, то $\lambda^{\varepsilon}(x, q') = s'$. В этом случае

$$y = \varphi^+(x, s) = \varphi^+(x, s', \psi_2^+(x, s')) \geq (\varphi(x, s', \psi_2(x, s'))) = \delta^+(x, s') = y'.$$

Из последнего соотношения, кроме того, видно, что если функции φ и ψ_2 монотонные, то $y = y'$. Теорема доказана.

Заметим, что применительно к синхронному функционированию автоматов это утверждение не имеет места, что является отражением известного факта зависимости подобного функционирования релейных устройств от способа выбора в них обратных связей (см., например, [14, § 14.3]).

Автомат A с пустым размыкающим множеством внутренних переменных называется комбинационным. Его асинхронное функционирование вырождается в отображение из X в Y .

В дальнейшем в автоматах наряду с функционированием нас будут интересовать их части — функционирования из заданных начальных состояний, определяемые следующим образом. Пусть $A = (X, Q, Y, \psi, \varphi)$ — произвольный полурешеточный автомат, f — его функционирование (синхронное или асинхронное) и q_0 — внутреннее состояние, $q_0 \in Q$. Представим f как векторную функцию $f_1 f_2$, в которой для любых $x \in X$ и $q \in Q$ из $f(x, q) = sy$ следует

$$f_1(x, q) = s \quad \text{и} \quad f_2(x, q) = y. \quad \text{Рассмотрим автомат}$$

$B = (X, Q, Y, f_1, f_2)$ и его подавтомат $B' = (X, Q', Y, f'_1, f'_2)$, в котором Q' — подмножество внутренних состояний B , достижимых в B (посредством входных слов) из состояния q_0 . Автомат B естественно называть автоматом данного функционирования f . Очевидно, что если f — синхронное функционирование автомата A , то $f_1 = \psi$, $f_2 = \varphi$ и $B = A$. В этом случае синхронное функционирование автомата B' мы называем синхронным функционированием автомата A из состояния q_0 . В случае же асинхронного f синхронное функционирование автомата B' совпадает с его асинхронным функционированием и называется асинхронным функционированием автомата A из состояния q_0 . Заметим, что для монотонного автомата A асинхронное функционирование из состояния q_0 совпадает по определению с асинхронным функционированием из q_0 его подавтомата $(X, Q_0, Y_0, \psi_0, \varphi_0)$, в котором Q_0 — подмножество внутренних состояний автомата A , достижимых в нём из состояния q_0 .

Далее, если f — любое из функционирований автомата $A = (X, Q, Y, \psi, \varphi)$, синхронное или асинхронное, из заданного состояния или нет, и полурешётки X', Q', Y' суть адекватные модели полурешёток X, Q, Y по некоторым конгруенциям λ, ρ, σ соответственно, то можно определить отображение $f': X' \times Q' \rightarrow Q' \times Y'$, положив для всех $x' \in X'$, $q' \in Q'$

$$f'(x', q') = \max [f(x', q')]_{\rho \times \sigma},$$

где, по-прежнему, $\max [a]_{\alpha}$ обозначает максимальный элемент в смежном классе $[a]_{\alpha}$. По определению функция f' является адекватной моделью функции f . Назовём её функционированием, соответственно синхронным или асинхронным, из заданного состояния или нет, автомата A , вычисленным с точностью до заданных конгруенций λ, ρ, σ . Если, кроме того, множества X, Q, Y и X', Q', Y' являются декартовыми произведениями некоторых полурешёток, а кон-

груенции λ, ρ, σ - декартовы произведения некоторых конгруенций на сомножителях первых, то f' называем также функционированием автомата A с точностью до этих последних конгруенций. Функционирование с точностью до заданных конгруенций называется далее функционированием с заданной точностью. Обратим внимание на тот факт, что согласно данному определению функционирование с точностью до больших конгруенций в действительности означает функционирование с меньшей точностью. Наименьшим конгруенциям, каковыми являются тождества, соответствует функционирование с максимально возможной (абсолютной) точностью, названное просто функционированием.

Автомат $A' = (X', Q', Y', \psi', \varphi')$ называется адекватной моделью полурешёточного автомата $A = (X, Q, Y, \psi, \varphi)$ по заданным конгруенциям λ, ρ, σ , или с точностью до них, если X', Q', Y' - адекватные модели полурешёток X, Q, Y по конгруенциям λ, ρ, σ соответственно и для любых $x' \in X'$ и $q' \in Q'$ имеет место

$$\psi'(x', q') = \max [\psi(x', q')]_{\rho}, \quad \varphi'(x', q') = \max [\varphi(x', q')]_{\sigma}.$$

В случае, если алфавиты автомата A являются декартовыми степенями некоторой полурешётки S , а λ, ρ, σ - соответствующими степенями некоторой конгруенции δ на S , то адекватная модель автомата A по конгруенциям λ, ρ, σ называется его адекватной моделью по конгруенции δ .

Наконец, если W - подмножество области определения какого-либо функционирования f автомата A и g - ограничение f на W , т.е. g определена на W и совпадает на W с f , то g называют соответствующим функционированием автомата A в заданной рабочей области W . Можно говорить, в частности, о синхронном и асинхронном функционировании A в заданной рабочей области, в том числе из заданного начального состояния или с заданной точностью. Рабочая область

W называется направленной, если на любом её элементе xq значение функции переходов автомата A сравнимо со значением внутреннего состояния q в нём, т.е. $\psi(x, q) \leq q$ или $\psi(x, q) \geq q$. Нетрудно заметить, что если автомат A монотонный, то любое его функционирование с некоторой точностью (λ, ρ, σ) в любой направленной рабочей области совпадает с таким же функционированием с абсолютной точностью в этой области для адекватной модели автомата A , построенной с той же точностью (λ, ρ, σ) .

Для монотонных автоматов это утверждение может быть принято за определение функционирования с заданной точностью в заданной

направленной рабочей области.

6. Схемы

Схемы в нашем понимании – это структурные модели цифровых автоматов на функционально-логическом или принципиально-электрическом уровнях представления. Ориентированные на объекты с иерархической структурой, каковыми являются, например, большие интегральные схемы, сами они имеют иерархическую структуру, т.е. составлены из блоков, которые, в свою очередь, являются схемами. Глубина этой иерархии может быть сколь угодно большой. На самом нижнем ярусе ее находятся элементы – схемы, не имеющие внутренней структуры.

Формально схемный элемент – это конечное множество полюсов. Понятие полюса первичное и не определяется. В геометрическом представлении полюс изображается точкой.

Ниже дается индуктивное определение схемы, ее блоков, элементов и узлов.

1. Элемент есть схема. Он называется блоком и одновременно элементом этой схемы. Одноэлементные подмножества его полюсов называются узлами данной схемы.

2. Совокупность конечного числа схем без общих узлов, рассматриваемая вместе с некоторым отношением эквивалентности на множестве всех их узлов, есть схема. Схемы в совокупности называются блоками, их элементы – элементами, а классы эквивалентности – узлами этой схемы.

В геометрическом представлении каждый узел схемы изображается системой отрезков, соединяющих некоторым образом входящие в него узлы блоков, а каждый ее блок – окружностью, прямоугольником или другой аналогичной фигурой с примыкающими к ней узлами блока.

Эквивалентные узлы блоков далее не различаются и отождествляются с содержащим их узлом схемы. Тем самым, в частности, каждый полюс элемента схемы отождествлен с некоторым узлом этой схемы.

Пусть $M = E^2$ для некоторого натурального $z \geq 1$. Элемент схемы называется переключательным элементом z -го ранга, если в нем выделено подмножество полюсов, называемых его управляющими полюсами, и в соответствие элементу поставлена матрица, называемая матрицей его собственных непосредственных проводимостей, в которой строки и столбцы находятся во взаимно однозначном соответствии со всеми полюсами элемента и на пересечении

строки и столбца, соответствующих полюсам i и j , записана монотонная функция из множества R_{dc} - собственная непосредственная проводимость (с.н.п.) элемента от полюса i к полюсу j , зависящая от аргументов, поставленных во взаимно однозначное соответствие управляющим полюсам элемента.

Переключательные элементы ранга 2 в нашей теории являются дискретными математическими моделями физических приборов и устройств с управляемой проводимостью, применяемых в электрических схемах с 2-полюсным источником питания. Типичным примером такого рода приборов может служить МДП-транзистор. В нем проводимость между истоком (и) и стоком (с) является некоторой функцией $T(s)$ от состояния s затвора (з), а проводимости между другими парами различных полюсов равны 0. Таким образом, матрица собственных непосредственных проводимостей любого МДП-транзистора имеет следующий вид:

$$\begin{array}{c|ccc} & \text{и} & \text{с} & \text{з} \\ \hline \text{и} & 1 & T(s) & 0 \\ \text{с} & T(s) & 1 & 0 \\ \text{з} & 0 & 0 & 1 \end{array}$$

Проводимости в схемах мы моделируем символами алфавита C , а состояния узлов в схемах с 2-полюсным источником питания - подмножествами множества E^2 ; поэтому в модели МДП-транзистора функция $T(s)$ определена на полурешетке $\tilde{E}^2$ и принимает значения в полурешетке C . В действительности эта функция аддитивная, задается на множестве E^2 и на множестве $\tilde{E}^2$ распространяется по правилу аддитивности: ее значение на $\alpha \in E^2$ определяется как объединение значений на всех $\beta \in \alpha$. Конкретный вид функции T зависит от типа МДП-транзистора. В табл.4 перечислены функции проводимости некоторых известных типов МДП-транзисторов, в том числе со встроенным и индуцированным каналами р- и n-типа, используемых в качестве симметричных ключей, и тех же типов транзисторов, используемых в качестве управляемых резисторов [7,8]. Указанные функции выписаны для случая $2=2$ в предположении, что схема, содержащая МДП-транзистор, питается от источника с двумя традиционными полюсами - "Питание" и "Земля", и компоненты в состоянии $s = ab$ суть (полные) проводимости этой схемы до его затвора: первая - от полюса "Земля", вторая - от полюса "Питание" (см.рис.1). Ввиду свойства аддитивности они выписаны только на минимальных элементах полурешетки E^2 .

Постоянные резисторы и диоды являются также примерами фи-

Таблица 4

S	T_1	T_2	T_3	T_4	T_5	T_6
00	0	1	0	1	0	1
01	1	1	0	0	X	X
10	0	0	1	1	0	1
11	0	0	0	0	0	*
0X	1	1	0	0	X	X
X0	0	0	1	1	0	1
1X	0	0	1	1	0	1
X1	1	1	0	0	X	X
XX	0	0	0	0	0	*

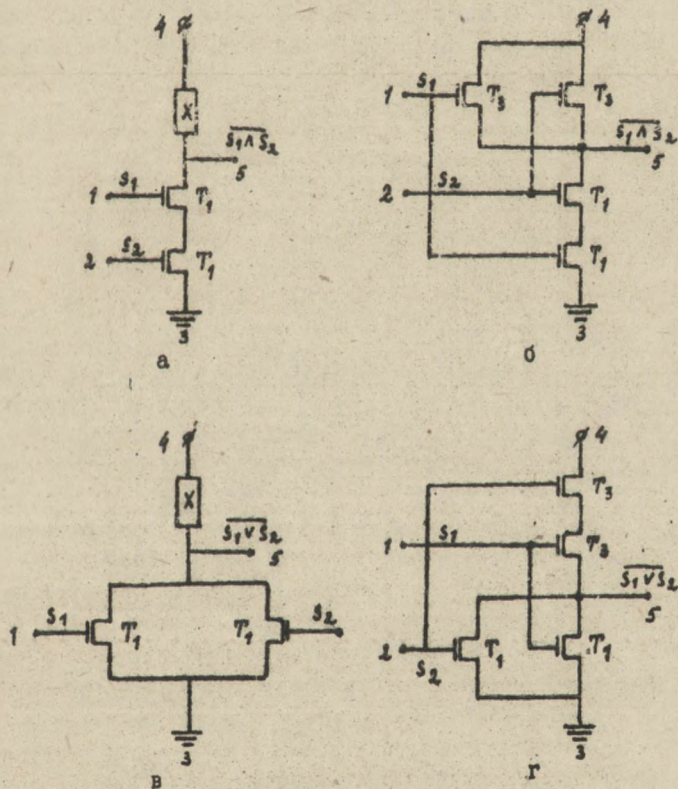


Рис. 3

зических приборов с управляемой проводимостью, подпадающих под модель переключательного элемента. Их матрицы собственных непосредственных проводимостей имеют вид

$$\begin{matrix} & \begin{matrix} 1 & 2 \end{matrix} \\ \begin{matrix} 1 \\ 2 \end{matrix} & \begin{bmatrix} 1 & x \\ x & 1 \end{bmatrix} \end{matrix} \quad \text{и} \quad \begin{matrix} & \begin{matrix} 1 & 2 \end{matrix} \\ \begin{matrix} 1 \\ 2 \end{matrix} & \begin{bmatrix} 1 & d(s_1, s_2) \\ \bar{d}(s_1, s_2) & 1 \end{bmatrix} \end{matrix}$$

соответственно, где $d(s_1, s_2)$ - аддитивная функция проводимости от состояний полюсов диода, равная 1, когда полюс 1 находится в состоянии s_1 , моделирующем высокий потенциал, а полюс 2 в состоянии s_2 , моделирующем низкий потенциал, и равная 0 в противоположной ситуации.

Наконец, как переключательные элементы можно представлять электрические схемы из транзисторов и резисторов, выполняющие простейшие логические функции. На рис.3 изображены некоторые из таких схем, выполняющих функции И-НЕ и ИЛИ-НЕ. В них полюсы 1 и 2 управляющие. Их матрицы собственных непосредственных проводимостей имеют следующий вид:

$$\begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & a \\ 0 & 0 & 0 & 1 & b \\ 0 & 0 & a & b & 1 \end{bmatrix} \end{matrix}$$

В этой матрице для элементов с n -МОП схемой (рис.3а,в) $b=x$, а для элементов с K -МОП схемой (рис.3б,г) функция b двойственна функции a , которая для элементов И-НЕ (рис.3а,б) есть $n(s_1) \wedge n(s_2)$, а для элементов ИЛИ-НЕ (рис.3в,г) есть $n(s_1) \vee n(s_2)$. Здесь s_1 и s_2 - состояния управляющих полюсов 1 и 2 соответственно, $n(s)$ - функция проводимости транзистора n -типа, а выражение для двойственной функции получается подстановкой знаков $\vee$, $\wedge$ и вместо знаков $\wedge$, $\vee$ и n соответственно.

Схема, в которой каждый элемент переключательный некоторого ранга z , называется переключательной схемой z -го ранга. Множество $M = E^z$ называется её основным множеством. Узлы переключательной схемы, которые являются (отождествлены с) управляющими полюсами ее элементов, называются внутренними полюсами данной схемы. Каждому узлу переключательной схемы с основным множеством M приписывается переменная со значениями из $\tilde{M}$ так, что разным узлам приписываются разные переменные. Они называются далее собствен-

ными переменными, а те из них, которые приписаны внутренним полюсам - внутренними переменными переключательной схемы. Индукцией по построению схемы определяется матрица непосредственных проводимостей переключательной схемы $\|f_{ij}\|$: в ней строки и столбцы находятся во взаимно однозначном соответствии с узлами схемы и элемент f_{ij} - непосредственная проводимость схемы от узла i к узлу j - есть дизъюнкция всех непосредственных проводимостей блоков от их узлов $a \in i$ до их узлов $b \in j$, а непосредственная проводимость элемента от узла a к узлу b - это его собственная непосредственная проводимость от полюса a к полюсу b , у которой каждый аргумент представлен (замещен) внутренней переменной схемы, приписанной тому ее узлу, который содержит соответствующий этому аргументу управляющий полюс данного элемента. Таким образом, непосредственные проводимости f_{ij} переключательной схемы являются функциями из $P_{\mathcal{M}S}$, зависящими от ее внутренних переменных. Ввиду теоремы 6 все они монотонные.

Пары узлов ij с $f_{ij} \neq 0$ называются ребрами переключательной схемы. Функция f_{ij} называется функцией проводимости или проводимостью ребра ij . Последовательность различных узлов $i_1, i_2, \dots, i_\ell$, в которой для каждого $k = 1, \dots, \ell-1$ пара $i_k i_{k+1}$ есть ребро схемы, называется цепью (от узла i_1 к узлу i_ℓ). Ребра $i_k i_{k+1}$ для $k = 1, \dots, \ell-1$ и только они называются ребрами данной цепи.

Если q - набор значений внутренних переменных некоторой схемы $\mathcal{N}$ и $F = \|f_{ij}\|$ - матрица её непосредственных проводимостей, то пусть $F(q) = \|f_{ij}(q)\|$ и $\mathcal{M}(q)$ обозначает множество всех матриц с элементами из E , которые можно получить из матрицы $F(q)$, замещая её элементы их представителями в E . Для произвольной матрицы $H = \|h_{ij}\| \in \mathcal{M}(q)$ и для любой цепи $i_1, i_2, \dots, i_\ell$ схемы $\mathcal{N}$ конъюнкцию $h_{i_1 i_2} \wedge h_{i_2 i_3} \wedge \dots \wedge h_{i_{\ell-1} i_\ell}$ будем называть проводимостью этой цепи в матрице H . Цепь называется замкнутой, разомкнутой или резистивной в H , если её проводимость в H равна 0, 1 или X соответственно. Пусть далее

- $$f_{ij}(q) = \begin{cases} 0, & \text{если все цепи схемы } \mathcal{N} \text{ от узла } i \text{ к узлу } j \text{ разомкнуты во всех матрицах из } \mathcal{M}(q); \\ 1, & \text{если для каждой матрицы в } \mathcal{M}(q) \text{ имеется замкнутая в ней цепь схемы } \mathcal{N} \text{ от } i \text{ к } j; \\ X, & \text{если для каждой матрицы в } \mathcal{M}(q) \text{ имеется резистивная и нет замкнутой в ней цепи схемы } \mathcal{N} \text{ от } i \text{ к } j; \\ 0, & \text{если в } \mathcal{M}(q) \text{ есть матрица, в которой замкнута хо-} \end{cases}$$

тя бы одна цепь $\mathcal{N}$ от i к j , и есть матрица, в которой разомкнуты все цепи $\mathcal{N}$ от i к j ;

$= \otimes$, если в $\mathcal{M}(q)$ есть матрица, в которой резистивна хотя бы одна цепь $\mathcal{N}$ от i к j и не замкнуты все другие такие цепи, и есть матрица, в которой разомкнуты все цепи $\mathcal{N}$ от i к j ;

$= *$, если в $\mathcal{M}(q)$ есть матрица, в которой резистивна хотя бы одна цепь от i к j и не замкнуты все другие такие цепи, и есть матрица, в которой замкнута хотя бы одна цепь $\mathcal{N}$ от i к j ;

$= \oplus$ во всех остальных случаях.

Определённая так функция $\tilde{f}_{ij} \in P_{\mathcal{M}S}$ называется полной проводимостью схемы $\mathcal{N}$ от узла i к узлу j , а матрица $\tilde{F} = \|\tilde{f}_{ij}\|$, где i и j пробегает множество всех узлов схемы $\mathcal{N}$, — матрицей полных проводимостей этой схемы. Ввиду теоремы 6 все полные проводимости переключательной схемы являются монотонными функциями.

Определение функционирования любой схемы предполагает задание у неё входов и выходов. В переключательной схеме ими служат некоторые её узлы, называемые соответственно её входными и выходными полюсами. Их множества предполагаются непустыми. (Полюсы источника питания реальных схем в их моделях — переключательных схемах, считаются входными полюсами с соответствующими постоянными состояниями: в схеме с z полюсами питания состояние i -го полюса из них есть единичный z -мерный вектор с 1 на i -м месте, $i = 1, \dots, z$, и с 0 на остальных местах. Заметим также, что широко распространённые на практике схемы, в которых каждый резистор одним своим концом присоединён к полюсу питания, в частности, все $\mathcal{N}$ -МОП схемы, представляются переключательными схемами без резисторов, в которых состояние соответствующего входного полюса на месте 1 содержит X .) Каждому входному полюсу схемы с основным множеством M приписывается некоторая переменная со значениями из $\tilde{M}$ так, что разным полюсам приписываются разные переменные, отличные, кроме того, и от всех собственных переменных схемы. Они называются её входными переменными. Собственные переменные схемы, приписанные выходным полюсам, называются также выходными переменными данной схемы.

Собственные переменные схемы, в том числе выходные и внутренние, не являются независимыми; они находятся в зависимости от входных и внутренних же переменных. Эта зависимость устанавливается при помощи так называемой собственной системы уравнений

схемы, которая записывается как

$$z_j = \bigvee_i x_i \wedge \tilde{f}_{ij}(z_{ij}), \quad (1)$$

где i пробегает множество входных полюсов, а j - множество всех узлов схемы; x_i - входная, а z_j - собственная переменные, приписанные соответственно входному полюсу i и узлу j схемы; $\tilde{f}_{ij}$ - полная проводимость схемы от узла i к узлу j и z_{ij} - набор её аргументов из множества внутренних переменных схемы. Часть этой системы, составленная из уравнений только для выходных и внутренних переменных, т.е. из всех таких уравнений, левые части которых суть выходные или внутренние переменные схемы, называется функциональной системой уравнений данной схемы.

Содержательно функционирование переключательной схемы заключается в том, что в начальный момент времени (до подачи на входы каких-либо сигналов) схема находится в отключенном состоянии, в котором состояние каждого её узла высокоимпедансное, т.е. равно 0¹, если 2 - ранг схемы, а в любой другой момент времени состояние z_j любого её узла j устанавливается равным значению выражения (1) для заданных в этот момент состояний x_i входных полюсов схемы и для состояний z_{ij} её внутренних полюсов, установленных в предыдущий момент. Формально оно определяется как функционирование из высокоимпедансного внутреннего состояния переключательного автомата, у которого каноническая система уравнений и множества входных, выходных и внутренних переменных совпадают соответственно с функциональной системой уравнений и множествами входных, выходных и внутренних переменных схемы. В дальнейшем мы называем его функциональным автоматом данной схемы. В соответствии с делением способов функционирования автоматов на синхронный и асинхронный можно говорить также о синхронном и асинхронном функционировании переключательных схем.

На практике часто одни и те же схемы применяются в разных рабочих областях, а к точности отображения их функций в моделях нередко предъявляются различные требования. Реальность такова, что отображение функционирования схемы в более широкой области или с более высокой точностью требует, как правило, более сложного описания, а имеющиеся ресурсы для его представления обычно ограничены. В этой связи в теории целесообразно так ввести определение функционирования переключательной схемы, чтобы оно отражало поведение схемы в заданной рабочей области с заданной степенью точности, которая в общем случае может быть своей для

каждого узла схемы. При этом естественно потребовать, чтобы функционирования одной и той же схемы, определённые в заданной рабочей области с разной степенью точности, были адекватными её функционированию, определённому в этой области с максимально возможной степенью точности. Данному требованию можно удовлетворить, ограничивая области значений переменных схемы полурешётками состояний, адекватно моделирующими полурешётки всех возможных (в рабочей области) состояний соответствующих узлов схемы, и определяя функционирование схемы в заданной рабочей области с заданной точностью как функционирование с этой точностью соответствующего подавтомата её функционального автомата.

Именно, пусть $\mathcal{M}$ - произвольная переключательная схема ранга r , имеющая p узлов, n входных, k внутренних и m выходных полюсов, занумерованных некоторым образом натуральными числами от 1 до p , n , k и m соответственно. Пусть для каждого $i = 1, \dots, n$ задан интервал $I_i \in C^2$ такой, что $0^2 \in I_i$. Будем интересоваться поведением схемы $\mathcal{M}$ из отключённого начального состояния в предположении, что состояния i -го входа, задаваемые извне схемы, принадлежат полурешётке $\tilde{I}_i$, $i = 1, \dots, n$. Минимальный интервал в C^2 , в пределах которого находятся все возможные при данном функционировании схемы $\mathcal{M}$ состояния некоторого её узла, будем называть рабочей областью этого узла.

В реальных схемах для узлов, не являющихся входными, рабочая область определяется обычно типом схемы и задаётся вместе с последним. Так, в n -МОП схемах с двухполюсным питанием ею является интервал $\oplus X$, а в КМОП схемах с таким же питанием - интервал $\oplus \oplus$. Кроме того, для тех входных полюсов i схемы, полные проводимости до которых от других входных полюсов равны тождественно 0, рабочие области совпадают с заданными интервалами входных состояний I_i . В общем же случае для определения рабочей области узла схемы можно воспользоваться следующей формальной процедурой.

Для каждого $j = 1, \dots, p$ и для любого $t = 1, 2, \dots$ индукцией по t определим интервал $q_j(t) \in C^2$ следующим образом:

- 1) $q_j(1) = \begin{cases} 0^2, & \text{если } j\text{-й узел не является входом схемы } \mathcal{M}; \\ I_i, & \text{если } j\text{-й узел схемы } \mathcal{M} \text{ является её } i\text{-м входом;} \end{cases}$
- 2) $q_j(t+1) = \bigvee_{i=1}^n I_i \otimes \tilde{f}_{ij}(u(t))$, $t \geq 1$,

где $u(t)$ - набор значений внутренних переменных схемы, в котором переменная, приписанная s -му узлу, равна $q_s(t)$. Пусть

$q(t) = q_1(t) \dots q_p(t)$. Поскольку $0^2 \in I_i$ для всех $i = 1, \dots, n$,

то по определению операций $\vee$ и $\odot$ и ввиду $\tilde{f}_{ji} = 1$ имеет место $q(1) \subseteq q(2)$. Ввиду же монотонности функций $\vee$, $\odot$ и $\tilde{f}_{ji}$ выражение для $q_j(t+1)$ задаёт монотонную функцию состояний, поэтому $q(1) \subseteq q(2) \subseteq q(3) \subseteq \dots$ и, следовательно, $q(\ell) = q(\ell+1)$ для некоторого $\ell \geq 1$. Интервал $q_j(\ell)$ для такого ℓ и есть по определению рабочая область j -го узла схемы $\mathcal{N}$.

Рабочие области j -го внутреннего и ℓ -го выходного полюсов схемы $\mathcal{N}$ обозначаются далее через L_j и K_ℓ соответственно. Интересуясь функционированием схемы $\mathcal{N}$ с некоторой точностью, мы предполагаем, что на полурешётках $\tilde{I}_i$, $\tilde{L}_j$ и $\tilde{K}_\ell$ для всех возможных i , j и ℓ заданы некоторые конгруэнции λ_i , ρ_j и σ_ℓ соответственно. Пятёрка объектов $A = (X, Q, Y, \psi, \varphi)$, где $X = \tilde{I}_1 \times \dots \times \tilde{I}_n$, $Q = \tilde{L}_1 \times \dots \times \tilde{L}_k$, $Y = \tilde{K}_1 \times \dots \times \tilde{K}_m$, а ψ и φ — ограничения на области $X \times Q$ функций переходов и выходов функционального автомата схемы $\mathcal{N}$, является подавтоматом последнего. Интересуясь функционированием схемы $\mathcal{N}$ в заданной рабочей области, предполагаем, что в качестве таковой указано некоторое множество $W \subseteq X \times Q$. Функционирование автомата A с точностью до конгруэнций λ_i , ρ_j , σ_ℓ , ограниченное областью W , и называется функционированием схемы $\mathcal{N}$ в заданной рабочей области с заданной точностью.

Следует заметить, что во всех построениях, относящихся к этому понятию, способ функционирования (синхронный или асинхронный) значения не имеет, но предполагается одним и тем же. Таким образом, мы имеем определение как синхронного, так и асинхронного функционирования переключательной схемы в заданной рабочей области с заданной точностью.

Рассмотрим типичный для практики случай, когда конгруэнции λ_i , ρ_j и σ_ℓ , заданные на полурешётках $\tilde{I}_i$, $\tilde{L}_j$ и $\tilde{K}_\ell$, являются ограничениями на последних некоторой конгруэнции δ , заданной на основной полурешётке $\tilde{M}$. Конгруэнция δ , в частности, может иметь вид $\tilde{\lambda}$, где λ — эквивалентность на базисе $\tilde{M}$. Этот последний случай возникает, например, всякий раз, когда в схемах не делается различий между физически эквивалентными состояниями узлов. Так, если левые компоненты в парах OI и KI суть проводимости до некоторого узла схемы от земли, а правая компонента — проводимость до этого узла от полюса источника с высоким потенциалом и $1 \gg X$, то высокий потенциал в данном узле можно создавать, реализовав в схеме любую из этих пар проводимостей, которые тем самым можно считать неразличимыми и отнести к одному

классу эквивалентности R . Физическая эквивалентность состояний, отражаемая в отношении R , задаётся обычно специалистом - схемотехником на основе анализа соотношений между физическими параметрами компонент рассматриваемой схемы.

Делая ещё одно приближение к практике, предположим, что конгруенция δ сохраняется дизъюнкцией состояний и конъюнкцией состояния с проводимостью, т.е. если a_1, b_1, a_2, b_2 - состояния и c - проводимость, то из $a_1 \delta a_2$ следует $(c \wedge a_1) \delta (c \wedge a_2)$, а из $a_1 \delta a_2$ и $b_1 \delta b_2$ - $(a_1 \vee b_1) \delta (a_2 \vee b_2)$. В этом случае будем говорить, что схема N сохраняет конгруенцию δ (а вместе с ней и заданные конгруенции $\lambda_i, \rho_j, \sigma_e$), если собственная непосредственная проводимость любого её элемента стабильна относительно δ , т.е. на конгруэнтных (находящихся в отношении δ) наборах значений переменных принимает равные значения. Легко видеть, что компоненты функций переходов и выходов такой схемы сами сохраняют δ . Кроме того, ввиду теоремы 8 каноническая система уравнений адекватной модели по конгруенции δ функционального автомата схемы N , сохраняющей δ , может быть записана как

$$z_j = \bigvee_{i=1}^n x_i \wedge \tilde{f}_{ij}', \quad j = 1, \dots, k, \quad (2)$$

где $\vee^a$ и $\wedge^a$ - адекватные модели функций $\vee$ и $\wedge$ соответственно, определённые с точностью до δ , и $\tilde{f}_{ij}'$ - ограничение функции $\tilde{f}_{ij}$ на области, где каждый её аргумент z_t принимает значения из множества максимальных элементов фактор-полурешётки $\tilde{L}_t$ по ρ_t , $t = 1, \dots, k$.

Из теоремы 8 следует также, что в случае, когда схема N не сохраняет δ , вычисление состояний её узлов с помощью (2) вместо (1), может привести лишь к увеличению значений, что равносильно понижению точности модели.

Л и т е р а т у р а

1. Яблонский С.В. Введение в дискретную математику. - М.: Наука, 1979.-272 с.
2. Мальцев А.И. Алгебраические системы. - М.: Наука, 1970.- 392 с.
3. Скорняков Л.А. Элементы теории структур. - М.: Наука, 1970.- 148 с.
4. Агibalов Г.П. и др. Математическая модель схем из элементов с управляемой проводимостью. - Автоматика и телемеханика, 1982, № 9, с. 89-98.
5. Агibalов Г.П. и др. Логическое проектирование переключательных автоматов. - Томск: Изд-во Томского университета, 1983.- 154 с.
6. Хейес Дж.М. Обобщенная теория переключательных схем и ее применение для проектирования СВИС. - ТИИЭР, 1982, т. 70, № 10, с. 5-19.
7. Кроуфорд Р. Схемные применения МОП-транзисторов. - М.: Мир, 1970.- 192 с.
8. Степаненко И.П. Основы микроэлектроники. - М.: Советское радио, 1984 - 424 с.
9. Закревский А.Д. Алгоритмы синтеза дискретных автоматов. - М.: наука, 1971. - 512 с.
10. Агibalов Г.П. Беляев В.А. Технология решения комбинаторно-логических задач. - Томск: Изд-во Томского университета, 1981. - 124 с.
11. Лазарев В.Г., Пийль Е.И. Синтез управляющих автоматов. - М.: Энергия, 1978.- 408 с.
12. Ангер С. Асинхронные последовательностные схемы. - М.: Наука, 1977. - 400 с.
13. Cerny E., Marin M.A. An approach to unified methodology of combinational switching circuits. - IEEE Trans. on Computers, 1977, v. C-26, n8, p. 745-756.
14. Колдуэлл С. Логический синтез релейных устройств. -М.: ИЛ, 1962. - 740 с.

Ю.С.Драгин

НЕКОТОРЫЕ УСЛОВИЯ СУЩЕСТВОВАНИЯ ДЕКОМПОЗИЦИИ АВТОМАТОВ

1. Определения

Предлагаются новые необходимые и достаточные условия существования изоморфной и гомоморфной представимости автомата произвольной (параллельной, последовательной, параллельно-последовательной, каскадной и т.д. [1, 2]) сетью из автоматов, в том числе и из автоматов, каждый из которых содержит меньшее число состояний, чем исходный автомат.

Под автоматом далее понимается тройка объектов $\langle X, Q, \Psi \rangle$, где X и Q множества, называемые соответственно входным алфавитом и алфавитом состояний автомата, а Ψ — функция переходов автомата, отображающая декартово произведение $X \times Q$ в Q , т.е. $\Psi: X \times Q \rightarrow Q$.

Автомат $A = \langle X, Q_A, \Psi_A \rangle$ называется гомоморфным образом автомата $B = \langle X, Q_B, \Psi_B \rangle$, если существует сюръективное отображение $\alpha: Q_B \rightarrow Q_A$ такое, что для любой пары $(x, q) \in X \times Q_B$ имеет место $\alpha(\Psi_B(x, q)) = \Psi_A(x, \alpha(q))$. В частности, когда отображение α является взаимно однозначным, автомат A называется изоморфным образом автомата B . Здесь и всюду далее при рассмотрении гомоморфизма между автоматами предполагается, что их входные алфавиты совпадают.

Автомат $A = \langle X, Q_A, \Psi_A \rangle$ называется подавтоматом автомата $B = \langle X, Q_B, \Psi_B \rangle$, если $Q_A \subseteq Q_B$ и для любой пары $(x, q) \in X \times Q_A$ имеет место $\Psi_A(x, q) = \Psi_B(x, q)$.

Говорят, что автомат A гомоморфно (изоморфно) представим автоматом B , если A является гомоморфным (изоморфным) образом некоторого подавтомата автомата B .

В дальнейшем, если $\varphi: X_1 \times \dots \times X_r \rightarrow Y_1 \times \dots \times Y_r$ есть некоторое отображение, то через $\varphi^i: X_1 \times \dots \times X_r \rightarrow Y_i$ ($i=1, \dots, r$) обозначаем i -ю компоненту отображения φ и функцию, соответствующую этому отображению, обозначаем через $\varphi^i(x_1, \dots, x_r)$.

Автомат $\langle X, Q_1 \times \dots \times Q_r, \Psi \rangle$ ($r \geq 1$) называем сетью; множества $Q_1, \dots, Q_r$ называем компонентами его алфавита состояний, автоматы $A_1 = \langle X \times Q_2 \times \dots \times Q_r, Q_1, \Psi_{A_1} \rangle, \dots, A_r = \langle X \times Q_1 \times \dots \times Q_{r-1}, Q_r, \Psi_{A_r} \rangle$, где для любого элемента $(x, q_1, \dots, q_r) \in X \times Q_1 \times \dots \times Q_r$ имеет место

$$\psi^i(x, q_1, \dots, q_\ell) = \psi_{A_1}((x, q_\ell, \dots, q_2); q_1), \dots, \psi_{A_\ell}(q_1, \dots, q_{\ell-1}, q_\ell) =$$

$\psi_{A_\ell}((x, q_1, \dots, q_{\ell-1}), q_\ell)$, называем компонентами этой сети. Сеть с компонентами $A_1, \dots, A_\ell$ будем называть сетью из автоматов $A_1, \dots, A_\ell$.

Пусть $J_1, \dots, J_\ell \subseteq \{0, 1, \dots, \ell\}$. Сеть $A = \langle Q_0, Q_1, \dots, Q_\ell, \psi \rangle$, где $Q_0 = X$ (с компонентами $A_1 = \langle Q_0 \times Q_2 \times \dots \times Q_\ell, Q_1, \psi_{A_1} \rangle, \dots, A_\ell = \langle Q_0 \times Q_1 \times \dots \times Q_{\ell-1}, Q_\ell, \psi_{A_\ell} \rangle$), называем $(J_1, \dots, J_\ell)$ - сетью, если для каждого $j = 1, \dots, \ell$ функция $\psi^i(q_0, q_1, \dots, q_\ell)$ отображения ψ (или равная ей функция $\psi_{A_j}(q_0, \dots, q_{j-1}, q_{j+1}, \dots, q_\ell), q_j$) не зависит существенно от переменных с индексами из $\{0, 1, \dots, \ell\} \setminus J_j$.

Например, сеть $A = \langle Q_0, Q_1, Q_2, \psi \rangle$, где функции $\psi^1(q_0, q_1, q_2)$ и $\psi^2(q_0, q_1, q_2)$ соответственно не зависят существенно от переменных q_0 и q_1, q_2 , есть по определению $(\{1, 2\}, \{0\})$ - сеть.

В случае двухкомпонентных сетей всеми парами подмножеств $J_1, J_2 \subseteq \{0, 1, 2\}$ в общем случае задается $2^3 \times 2^3 = 64$ различных типа таких сетей. Эти сети схематически изображены на рис. I. Нетрудно видеть, что с точностью до перестановки местами подмножеств J_1 и J_2 в обозначениях этих сетей число различных типов двухкомпонентных сетей равно 36. Соответственно $(\{0, 1\}, \{0, 1, 2\})$ - сети, $(\{0, 1\}, \{1, 2\})$ - сети и $(\{0, 1\}, \{0, 2\})$ - сети называются параллельно-последовательными, последовательными и параллельными двухкомпонентными сетями.

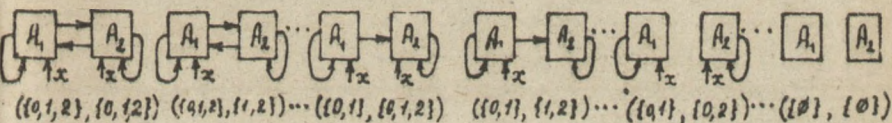


Рис. I

2. Необходимые и достаточные условия существования изоморфного представления автомата $(J_1, \dots, J_\ell)$ - сетью - при $0 \in J_1, \dots, J_\ell \subseteq \{0, 1, \dots, \ell\}$

Множество $\pi = \{\pi_1, \dots, \pi_\ell\}$ из необязательно различных, возможно в том числе и пустых, подмножеств множества Q , объединение которых есть Q , называется покрытием множества Q . Подмножества в покрытии называются блоками этого покрытия. Покрытие с непересекающимися блоками называется разбиением. Пусть $(\pi_1, \dots, \pi_\ell)$ есть система покрытий множества Q , элемент $(a_1, \dots, a_\ell) \in \pi_1 \times \dots \times \pi_\ell$ называем набором блоков этой системы. Пишут

$\pi_1 \times \dots \times \pi_e = 0$, если $a_1 \wedge \dots \wedge a_e \leq 1$ для любого набора блоков $(a_1, \dots, a_e) \in \pi_1 \times \dots \times \pi_e$. Пусть $\pi_1, \dots, \pi_e \subseteq \{a_1, \dots, a_e\}$ и $(\pi_1, \dots, \pi_e)$ есть система из e покрытий множества состояний Q_A автомата $A = \langle X, Q_A, \Psi_A \rangle$. Говорим, что в автомате A набор блоков $(a_1, \dots, a_e) \in \pi_1 \times \dots \times \pi_e$ ($\pi_1, \dots, \pi_e$) — сохраняется по x набором блоков $(b_1, \dots, b_e) \in \pi_1 \times \dots \times \pi_e$, если для каждого $j = 1, \dots, e$

$$\Psi_A(x, \bigwedge_{i \in J_j} a_i) \leq b_j.$$

Здесь и далее, если $R \subseteq Q_A$, то $\Psi_A(x, R) = \{\Psi_A(x, q) : q \in R\}$.

Говорим, что система покрытий $\pi_1, \dots, \pi_e$ ($\pi_1, \dots, \pi_e$) — сохраняема в автомате A , если для любого $x \in X$ и любого набора блоков $(a_1, \dots, a_e) \in \pi_1 \times \dots \times \pi_e$ существует $(\pi_1, \dots, \pi_e)$ — сохраняющий его по x набором блоков $(b_1, \dots, b_e) \in \pi_1 \times \dots \times \pi_e$.

Теорема I. Пусть $A = \langle X, Q_A, \Psi_A \rangle$ есть автомат с $n \geq 1$ состояниями; $n_1, \dots, n_e$ — натуральные числа; $0 \leq \pi_1, \dots, \pi_e \subseteq \{a_1, \dots, a_e\}$. Тогда для того чтобы автомат A был изоморфно представим $(\pi_1, \dots, \pi_e)$ — сетью из e автоматов с $n_1, \dots, n_e$ состояниями соответственно необходимо и достаточно существования $(\pi_1, \dots, \pi_e)$ — сохраняемой автоматом A системы из e разбиений $R_1, \dots, R_e$ множества Q_A соответственно с $n_1, \dots, n_e$ блоками, для которой $R_1 \times \dots \times R_e = 0$.

Доказательство теоремы I. Достаточность. Пусть $A = \langle X, Q_A, \Psi_A \rangle$ есть автомат с $n \geq 1$ состояниями; $0 \leq \pi_1, \dots, \pi_e \subseteq \{a_1, \dots, a_e\}$. Пусть $R_1 = \{z_{11}, \dots, z_{1n_1}\}, \dots, R_e = \{z_{e1}, \dots, z_{en_e}\}$ есть система разбиений множества Q_A соответственно с $n_1, \dots, n_e$ блоками, $(\pi_1, \dots, \pi_e)$ — сохраняемая автоматом A и для которой $R_1 \times \dots \times R_e = 0$; т.е. $R_1, \dots, R_e$ есть система разбиений множества Q_A , для которой выполняются условия:

1) для любого $x \in X$ и любого набора блоков $(a_1, \dots, a_e) \in R_1 \times \dots \times R_e$ существует $(\pi_1, \dots, \pi_e)$ — сохраняющий его по x набором блоков $(b_1, \dots, b_e) \in R_1 \times \dots \times R_e$, т.е. такой набором блоков, что для каждого $j = 1, \dots, e$ имеет место $\Psi_A(x, \bigwedge_{i \in J_j} a_i) \leq b_j$;

2) для любого набора блоков $(a_1, \dots, a_e) \in R_1 \times \dots \times R_e$ имеет место $a_1 \wedge \dots \wedge a_e \leq 1$.

Требуется показать, что тогда автомат $A = \langle X, Q_A, \Psi_A \rangle$ изоморфно представим некоторой $(\pi_1, \dots, \pi_e)$ — сетью из e автоматов с $n_1, \dots, n_e$ состояниями соответственно, т.е. требуется показать, что выполняются следующие условия:

а) найдется некоторая сеть $B = \langle X, Q, \dots, Q_e, \Psi_B \rangle$ из e ком-

понент $B_i = \langle X * Q_1 * \dots * Q_e, a_i, \Psi_{B_i} \rangle, \dots, B_e = \langle X * Q_1 * \dots * Q_{e-1}, Q_e, \Psi_{B_e} \rangle$
с $n_1, \dots, n_e$ состояниями соответственно такая, что для каждого $j = 1, \dots, e$ функция $\Psi_{B_j}^j(x_0, q_1, \dots, q_e)$ отображения Ψ_{B_j} (или равная ей функция переходов $\Psi_{B_j}^j((x_0, q_1, \dots, q_{j-1}, q_{j+1}, \dots, q_e), q_j)$ компоненты B_j) не зависит существенно от переменных с индексами из $\{1, \dots, e\} \setminus \{j\}$;

б) найдется некоторая подсеть (подавтомат) $\Delta = \langle X, Q_\Delta, \Psi_\Delta \rangle$ сети B (т.е. подмножество $Q_\Delta \subseteq Q_1 * \dots * Q_e$ и отображение $\Psi_\Delta: X * Q_\Delta \rightarrow Q_\Delta$) и взаимно однозначное отображение $\alpha: Q_\Delta \rightarrow Q_A$ такое, что для любых $x \in X, (a_1, \dots, a_e) \in Q_\Delta$ имеет место $\alpha(\Psi_\Delta(x, (a_1, \dots, a_e))) = \Psi_A(x, \alpha(a_1, \dots, a_e))$.

Покажем, что выполняется пункт а) доказываемого утверждения.

Построим произвольное отображение $\beta: R_0 * R_1 * \dots * R_e \rightarrow R_1 * \dots * R_e$, где $R_0 = X$ такое, что для любого $(a_0, a_1, \dots, a_e) \in R_0 * R_1 * \dots * R_e$ набор блоков $(a_1, \dots, a_e) \in R_1 * \dots * R_e$ ($\mathcal{I}_1, \dots, \mathcal{I}_e$) - сохраняется по a_0 набором блоков $\beta(a_0, a_1, \dots, a_e)$; согласно пункту I условия доказываемого утверждения такое отображение всегда можно построить. Преобразуем отображение β в отображение $\Psi_\beta: R_0 * R_1 * \dots * R_e \rightarrow R_1 * \dots * R_e$ следующим образом: для каждого $j = 1, \dots, e$ и каждого $(a_{i_1}, \dots, a_{i_j}) \in R_{i_1} * \dots * R_{i_j}$, где $\{i_1, \dots, i_j\} = \mathcal{I}_j \setminus \{0\}$, функция $\Psi_\beta^j(y_0, \dots, y_{j-1}, a_{i_1}, y_{j+1}, \dots, y_{j-1}, a_{i_j}, y_{j+1}, \dots, y_e)$ (функция, являющаяся j -й проекцией отображения Ψ_β , с фиксированными в ней значениями переменных $y_1, \dots, y_{i_1}, \dots, y_{i_j}, \dots, y_{i_{j-1}}, y_{i_{j+1}}, \dots, y_{i_e}$ соответственно равными $a_{i_1}, \dots, a_{i_j}$) есть константа, значение которой совпадает с одним (произвольным) из значений функции $\beta_j(y_0, \dots, y_{j-1}, a_{i_1}, y_{j+1}, \dots, y_{j-1}, a_{i_j}, y_{j+1}, \dots, y_e)$. Нетрудно видеть, что для каждого $j = 1, \dots, e$ определенная таким образом функция $\Psi_\beta^j(y_0, \dots, y_e)$ не зависит существенно от переменных с индексами из $\{0, 1, \dots, e\} \setminus \mathcal{I}_j$ и для любых $x \in X, (a_1, \dots, a_e) \in R_1 * \dots * R_e$ набор блоков $(a_1, \dots, a_e)$ является $(\mathcal{I}_1, \dots, \mathcal{I}_e)$ - сохраняемым по x набором блоков $\Psi_\beta(x, a_1, \dots, a_e)$.

Рассмотрим сеть $B = \langle X, R_1 * \dots * R_e, \Psi_\beta \rangle$ из e компонент $B_i = \langle X * R_1 * \dots * R_e, R_i, \Psi_{B_i} \rangle, \dots, B_e = \langle X * R_1 * \dots * R_{e-1}, R_e, \Psi_{B_e} \rangle$ (т.е. для каждого $j = 1, \dots, e$ алфавит состояний компоненты B_j сети B совпадает с множеством обозначений блоков разбиения R_j , имеющего n_j блоков) с $n_1, \dots, n_e$ состояниями соответственно. Нетрудно видеть, что сеть B по построению удовлетворяет пункту а) доказываемого утверждения.

Покажем, что выполняется пункт б) доказываемого утверждения.

Пусть Q_Δ есть множество всех тех наборов блоков $(a_1, \dots, a_e) \in R_1 * \dots * R_e$, для которых $(a_1, \dots, a_e) = t$, а Ψ_Δ есть часть

отображения Ψ_B с областью определения на множестве $X \times Q_A$.

Очевидно, $Q_A \subseteq R_1 \times \dots \times R_\ell$. Покажем, что $\Psi_B(X, Q_A) \subseteq Q_A$, т.е. покажем, что отображение Ψ_B есть отображение вида $\Psi_B: X \times Q_A \rightarrow Q_A$. Действительно, пусть $x \in X$ и $(a_1, \dots, a_\ell) \in Q_A$, тогда по способу построения отображения Ψ_B набор блоков $\Psi_B(x, (a_1, \dots, a_\ell)) = (b_1, \dots, b_\ell)$ является $(j_1, \dots, j_\ell)$ - сохраняющим по x набор блоков $(a_1, \dots, a_\ell)$, т.е. $\Psi_B(x, (a_1, \dots, a_\ell)) \subseteq b_j$ для каждого $j = 1, \dots, \ell$. Отсюда и из того, что $|a_1 \cap \dots \cap a_\ell| = 1$ следует, что $|b_1 \cap \dots \cap b_\ell| = 1$, а значит $(b_1, \dots, b_\ell) \in Q_A$, и следовательно, $\Psi_B(x, Q_A) \subseteq Q_A$.

Таким образом, в силу того, что $Q_A \subseteq R_1 \times \dots \times R_\ell$ и $\Psi_B: X \times Q_A \rightarrow Q_A$, автомат $\Delta = \langle X, Q_A, \Psi_B \rangle$ есть подсеть сети $B = \langle X, R_1 \times \dots \times R_\ell, \Psi_B \rangle$.

Зададим отображение $\alpha: Q_A \rightarrow Q_A$ так, что для каждого $(a_1, \dots, a_\ell) \in Q_A$ имеет место $\alpha(a_1, \dots, a_\ell) = a_1 \cap \dots \cap a_\ell$. Из определения множества Q_A и из того, что $R_1, \dots, R_\ell$ есть разбиения множества Q_A , следует, что отображение α является взаимно однозначным. Кроме этого, нетрудно видеть, из того факта, что для любых $x \in X$, $(a_1, \dots, a_\ell) \in Q_A$ набор блоков $(a_1, \dots, a_\ell)$ является $(j_1, \dots, j_\ell)$ - сохраняемым по x набором блоков $\Psi_B(x, (a_1, \dots, a_\ell)) = (b_1, \dots, b_\ell)$, а также из того, что $\alpha(a_1, \dots, a_\ell) = a_1 \cap \dots \cap a_\ell$ и $|a_1 \cap \dots \cap a_\ell| = 1$, а $\alpha(b_1, \dots, b_\ell) = b_1 \cap \dots \cap b_\ell$ и $|b_1 \cap \dots \cap b_\ell| = 1$, следует, что $\alpha(\Psi_B(x, (a_1, \dots, a_\ell))) = \Psi_B(x, \alpha(a_1, \dots, a_\ell))$. Следовательно, подсеть $\Delta = \langle X, Q_A, \Psi_B \rangle$ удовлетворяет пункту б) доказываемого утверждения. Достаточность теоремы I доказана.

Необходимость. Пусть $A = \langle X, Q_A, \Psi_A \rangle$ есть автомат с $n \geq 1$ состояниями $j_1, \dots, j_\ell \in \{1, \dots, \ell\}$. Пусть автомат A изоморфно представим некоторой $(j_1, \dots, j_\ell)$ - сетью из ℓ автоматов соответственно с $n_1, \dots, n_\ell$ состояниями, т.е. пусть выполняются пункты а) и б), сформулированные ранее при доказательстве достаточных условий доказываемой теоремы.

Требуется показать, что существует $(j_1, \dots, j_\ell)$ - сохраняемая автоматом A система из ℓ разбиений $R_1, \dots, R_\ell$ множества Q_A соответственно с $n_1, \dots, n_\ell$ блоками, для которой $R_1 \times \dots \times R_\ell = O$, т.е. требуется показать, что существует система разбиений $R_1, \dots, R_\ell$ множества Q_A соответственно с $n_1, \dots, n_\ell$ блоками такая, что для нее пункты 1 и 2, сформулированные ранее, при доказательстве достаточных условий доказываемой теоремы выполняются.

Покажем, что выполняется пункт 2 доказываемого утверждения.

По множествам состояний $Q_1 = \{q_{11}, \dots, q_{1n_1}\}, \dots, Q_\ell = \{q_{\ell 1}, \dots, q_{\ell n_\ell}\}$ компонент $V_1 = \langle X \times Q_1 \times \dots \times Q_\ell, Q_1, \Psi_{q_1} \rangle, \dots, V_\ell = \langle X \times Q_1 \times \dots \times Q_\ell, Q_\ell, \Psi_{q_\ell} \rangle$ и по подсети $\Delta = \langle X, Q_A, \Psi_B \rangle$ сети $B = \langle X, Q_1 \times \dots \times Q_\ell, \Psi_B \rangle$ постро-

им разбиения $R_1 = \{z_{11}, \dots, z_{1n_1}\}, \dots, R_\ell = \{z_{\ell 1}, \dots, z_{\ell n_\ell}\}$ (блок $z_{i\ell}$ разбиения R_i для каждого $i=1, \dots, \ell$ и $\ell=1, \dots, n_i$ считаем соответствующим состоянию $q_{i\ell}$ множества состояний Q_i и обозначаем через $\mathbb{F}(q_{i\ell})$, а состояние $q_{i\ell}$, как обратно соответствующее блоку $z_{i\ell}$ обозначаем через $\mathbb{F}'(z_{i\ell})$, т.е. $z_{i\ell} = \mathbb{F}(q_{i\ell})$ и $q_{i\ell} = \mathbb{F}'(z_{i\ell})$, возможно, в том числе, и с пустыми блоками, следующим образом: для каждого $q \in Q_A$ рассмотрим набор состояний $(z_1, \dots, z_\ell) \in Q_A \subseteq Q_1 \times \dots \times Q_\ell$ такой, что для отображения $\lambda: Q_A \rightarrow Q_A$ имеет место $\lambda(z_1, \dots, z_\ell) = q$, и рассмотрим соответствующий ему набор блоков $(a_1, \dots, a_\ell) = (\mathbb{F}(z_1), \dots, \mathbb{F}(z_\ell))$ и отнесем состояние q ко всем блокам $a_1, \dots, a_\ell$. Очевидно, что блок какого-либо разбиения, к которому не отнесется по указанной процедуре ни одно состояние из Q_A , будет пустым.

Из способа построения разбиений $R_1, \dots, R_\ell$ нетрудно видеть, что для любого набора блоков $(a_1, \dots, a_\ell) \in R_1 \times \dots \times R_\ell$ имеет место $|a_1 \wedge \dots \wedge a_\ell| \leq 1$; при этом $|a_1 \wedge \dots \wedge a_\ell| = 1$, если $(\mathbb{F}'(a_1), \dots, \mathbb{F}'(a_\ell)) \in Q_A$ и $|a_1 \wedge \dots \wedge a_\ell| = 0$ (т.е. $a_1 \wedge \dots \wedge a_\ell = \emptyset$), если $\mathbb{F}'(a_1), \dots, \mathbb{F}'(a_\ell) \in Q_1 \times \dots \times Q_\ell \setminus Q_A$. Таким образом, система разбиений $R_1, \dots, R_\ell$ удовлетворяет пункту 2 доказываемого утверждения.

Покажем, что выполняется пункт I доказываемого утверждения.

Зададим отображение $\varphi: X \times R_1 \times \dots \times R_\ell \rightarrow R_1 \times \dots \times R_\ell$, соответствующее функции переходов $\psi_\theta: X \times Q_1 \times \dots \times Q_\ell \rightarrow Q_1 \times \dots \times Q_\ell$ сети $B = \langle X, Q_1 \times \dots \times Q_\ell, \psi_\theta \rangle$ следующим образом: для каждого $x \in X$ и $(a_1, \dots, a_\ell) \in R_1 \times \dots \times R_\ell$ имеет место $\varphi(x, a_1, \dots, a_\ell) = (b_1, \dots, b_\ell)$ тогда и только тогда, когда $\psi_\theta(x, (\mathbb{F}'(a_1), \dots, \mathbb{F}'(a_\ell))) = (\mathbb{F}'(b_1), \dots, \mathbb{F}'(b_\ell))$.

Укажем последующие свойства б) и 2) отображения φ :

в) согласно условию пункта а) доказываемого утверждения для каждого $j=1, \dots, \ell$ проекция $\psi_\theta^j(x, a_1, \dots, a_\ell)$ отображения $\psi_\theta: X \times Q_1 \times \dots \times Q_\ell \rightarrow Q_1 \times \dots \times Q_\ell$ не зависит существенно от переменных с индексами из множества $\{1, \dots, \ell\} \setminus j$; отсюда и из способа построения отображения $\varphi: X \times R_1 \times \dots \times R_\ell \rightarrow R_1 \times \dots \times R_\ell$ следует, что для каждого $j=1, \dots, \ell$ и проекция $\varphi^j(x, a_1, \dots, a_\ell)$ отображения φ не зависит существенно от переменных с индексами из множества $\{1, \dots, \ell\} \setminus j$.

г) пусть R_A есть множество всех тех наборов блоков из $R_1 \times \dots \times R_\ell$, которые соответствуют наборам состояний из Q_A . Согласно условию пункта б) доказываемого утверждения для каждого $x \in X$ и $(a_1, \dots, a_\ell) \in R_A$ имеет место $\lambda(\psi_\theta(x, (a_1, \dots, a_\ell))) = \psi_\theta(x, \lambda(a_1, \dots, a_\ell))$, где $\lambda: R_A \rightarrow R_A$ есть взаимно однозначное отображение; отсюда и из способа построения отображения $\varphi: X \times R_1 \times \dots \times R_\ell \rightarrow R_1 \times \dots \times R_\ell$ следует, что для каждого $x \in X$ и $(a_1, \dots, a_\ell) \in R_A$ имеет место; если

$\varphi(x, (a_1, \dots, a_\ell)) = (b_1, \dots, b_\ell)$, то $\varphi_A(x, (a_1 \wedge \dots \wedge a_\ell)) = b_1 \wedge \dots \wedge b_\ell$.

Далее покажем, что для любых $x \in X$ и $(a_1, \dots, a_\ell) \in R_1 \times \dots \times R_\ell$ набор блоков $(b_1, \dots, b_\ell) \in \varphi(x, (a_1, \dots, a_\ell))$ таков, что для каждого $j = 1, \dots, \ell$ $\varphi_A(x, \bigwedge_{i \in J_j} a_i) \subseteq b_j$.

Действительно, пусть $x \in X$, $(a_1, \dots, a_\ell) \in R_1 \times \dots \times R_\ell$, $\varphi(x, (a_1, \dots, a_\ell)) = (b_1, \dots, b_\ell)$, $j \in \{1, \dots, \ell\}$, $J_j = \{i_1, \dots, i_s\}$. Рассмотрим функцию $\varphi'(x, y_1, \dots, y_{i_1}, a_{i_1}, y_{i_2}, \dots, y_{i_s}, a_{i_s}, y_{i_{s+1}}, \dots, y_\ell) = b_j$ (функцию, являющуюся j -й проекцией отображения φ , с фиксированными в ней значениями переменных $y_{i_1}, \dots, y_{i_s}$, равными соответственно $a_{i_1}, \dots, a_{i_s}$), которая есть функции-константа, так как по построению (свойство δ) не зависит существенно от переменных с индексами из множества $\{i_1, \dots, i_s\} \setminus J_j$.

Пусть $q = a_{i_1} \wedge \dots \wedge a_{i_s}$. Нетрудно видеть, что найдется набор блоков $c = (c_1, \dots, c_{i_1}, a_{i_1}, c_{i_1+1}, \dots, c_{i_2}, a_{i_2}, c_{i_2+1}, \dots, c_\ell) \in R_1 \times \dots \times R_\ell$ такой, что $\{q\} = c_1 \wedge \dots \wedge c_{i_1} \wedge a_{i_1} \wedge c_{i_1+1} \wedge \dots \wedge c_\ell$. Очевидно, что $\varphi'(x, c) = b_j$ и $\varphi(x, c) = (d_1, \dots, d_{i_1}, b_j, d_{i_1+1}, \dots, d_\ell) \in R_1 \times \dots \times R_\ell$ и согласно условию γ) имеет место $\varphi_A(x, c_1 \wedge \dots \wedge c_{i_1} \wedge a_{i_1} \wedge c_{i_1+1} \wedge \dots \wedge c_\ell) = d_1 \wedge \dots \wedge d_{i_1} \wedge b_j \wedge d_{i_1+1} \wedge \dots \wedge d_\ell$.

т.е. $\varphi_A(x, q) = d_1 \wedge \dots \wedge d_{i_1} \wedge b_j \wedge d_{i_1+1} \wedge \dots \wedge d_\ell$, и следовательно, $\varphi_A(x, q) \subseteq b_j$. Этим и показано, что $\varphi_A(x, \bigwedge_{i \in J_j} a_i) \subseteq b_j$, т.е. показали, что выполняется пункт I доказываемого утверждения. Необходимость теоремы I доказана. Теорема I доказана.

3. Достаточные условия существования гомоморфного представления автомата $(J_1, \dots, J_\ell)$ - сетью при $0 \in J_1, \dots, J_\ell \subseteq \{a_1, \dots, a_\ell\}$

Теорема Ia. Пусть $A = \langle X, Q_A, \varphi_A \rangle$ есть автомат с $n \geq 1$ состояниями; $n_1, \dots, n_\ell$ - натуральные числа; $a_1, \dots, a_\ell \in \{0, 1, \dots, \ell\}$. Тогда для того, чтобы автомат A был гомоморфно представим $(J_1, \dots, J_\ell)$ - сетью из ℓ автоматов с $n_1, \dots, n_\ell$ состояниями соответственно, достаточно существования $(J_1, \dots, J_\ell)$, - сохраняемой автоматом A системы из ℓ покрытий $\varphi_1, \dots, \varphi_\ell$ множества Q_A соответственно с $n_1, \dots, n_\ell$ блоками, для которой $\varphi_1 \times \dots \times \varphi_\ell = 0$.

Доказательство теоремы Ia. Если всюду в тексте доказательства достаточных условий теоремы I термин "изоморфно представим" заменить на термин "гомоморфно представим", термин "взаимно однозначное отображение" - на термин "отображение", "систему разбиений" - на "систему покрытий", а всё остальное в этом тексте оставить без изменения, то полученный таким образом текст будет являться доказательством теоремы Ia.

4. Необходимые и достаточные условия существования гомоморфного представления автомата $(\mathcal{A}, \dots, \mathcal{A}_\ell)$ -сеть при $\mathcal{A}_1, \dots, \mathcal{A}_\ell \subseteq \{0, 1, \dots, \ell\}$

Говорим, что система $(\mathcal{A}_1, \dots, \mathcal{A}_\ell)$ покрытий множества состояний Q_A автомата $A = \langle X, Q_A, \psi_A \rangle$ отмечена отображением $\alpha: \mathcal{A}_1 \times \dots \times \mathcal{A}_\ell \rightarrow Q_A \cup \{\emptyset\}$, если для любого $(a_1, \dots, a_\ell) \in \mathcal{A}_1 \times \dots \times \mathcal{A}_\ell$, либо $\alpha(a_1, \dots, a_\ell) \in a_1 \cap \dots \cap a_\ell$, либо $\alpha(a_1, \dots, a_\ell) = \emptyset$ и отображение α в Q_A является сюръективным. Говорим, что система покрытий $\mathcal{A}_1, \dots, \mathcal{A}_\ell$ множества Q_A состояний автомата $A = \langle X, Q_A, \psi_A \rangle$, отмеченная отображением α , $(\mathcal{A}_1, \dots, \mathcal{A}_\ell)_\alpha$ - сохраняема ($\mathcal{A}_1, \dots, \mathcal{A}_\ell \subseteq \{0, 1, \dots, \ell\}$) автоматом A , если существует отображение $\beta: X \times \mathcal{A}_1 \times \dots \times \mathcal{A}_\ell \rightarrow \mathcal{A}_1 \times \dots \times \mathcal{A}_\ell$ такое, что для каждого $j = 1, \dots, \ell$ проекция $\beta^j(x, a_1, \dots, a_\ell)$ отображения β не зависит существенно от переменных с индексами из множества $\{0, 1, \dots, \ell\} \setminus \mathcal{A}_j$ и для любых $x \in X, (a_1, \dots, a_\ell) \in \mathcal{A}_1 \times \dots \times \mathcal{A}_\ell$ имеет место $\psi(x, \alpha(a_1, \dots, a_\ell)) = \alpha(\beta(a_1, \dots, a_\ell))$.

Теорема 2. Пусть $A = \langle X, Q_A, \psi_A \rangle$ есть автомат с $n \geq 1$ состояниями; $n_1, \dots, n_\ell$ - натуральные числа; $\mathcal{A}_1, \dots, \mathcal{A}_\ell \subseteq \{0, 1, \dots, \ell\}$. Тогда для того, чтобы автомат A был гомоморфно представим $(\mathcal{A}_1, \dots, \mathcal{A}_\ell)$ - сетью из ℓ автоматов с $n_1, \dots, n_\ell$ состояниями соответственно, необходимо и достаточно существования $(\mathcal{A}_1, \dots, \mathcal{A}_\ell)_\alpha$ - сохраняемой автоматом A системы из ℓ покрытий $\mathcal{A}_1, \dots, \mathcal{A}_\ell$ множества Q_A соответственно с $n_1, \dots, n_\ell$ блоками.

Доказательство теоремы 2. Достаточность. Пусть $A = \langle X, Q_A, \psi_A \rangle$ есть автомат с $n \geq 1$ состояниями; $\mathcal{A}_1, \dots, \mathcal{A}_\ell \subseteq \{0, 1, \dots, \ell\}$. Пусть $\mathcal{A}_j = \{\mathcal{A}_{j1}, \dots, \mathcal{A}_{jn_j}\}$, $\dots$, $\mathcal{A}_\ell = \{\mathcal{A}_{\ell 1}, \dots, \mathcal{A}_{\ell n_\ell}\}$ - система покрытий множества Q_A соответственно с $n_1, \dots, n_\ell$ блоками $(\mathcal{A}_1, \dots, \mathcal{A}_\ell)_\alpha$ - сохраняемая автоматом A , т.е. пусть $\mathcal{A}_1, \dots, \mathcal{A}_\ell$ есть система покрытий множества Q_A , для которой следующие условия выполняются:

1) найдется отображение $\beta: X \times \mathcal{A}_1 \times \dots \times \mathcal{A}_\ell \rightarrow \mathcal{A}_1 \times \dots \times \mathcal{A}_\ell$ такое, что для каждого $j = 1, \dots, \ell$ проекция $\beta^j(x, a_1, \dots, a_\ell)$ отображения β не зависит существенно от переменных с индексами из множества $\{0, 1, \dots, \ell\} \setminus \mathcal{A}_j$;

2) найдется отображение $\alpha: \mathcal{A}_1 \times \dots \times \mathcal{A}_\ell \rightarrow Q_A \cup \{\emptyset\}$ такое, что для любого $(a_1, \dots, a_\ell) \in \mathcal{A}_1 \times \dots \times \mathcal{A}_\ell$ либо $\alpha(a_1, \dots, a_\ell) \in a_1 \cap \dots \cap a_\ell$, либо $\alpha(a_1, \dots, a_\ell) = \emptyset$ и отображение α в Q_A является сюръективным;

3) для любых $x \in X, (a_1, \dots, a_\ell) \in \mathcal{A}_1 \times \dots \times \mathcal{A}_\ell$ имеет место $\psi_A(x, \alpha(a_1, \dots, a_\ell)) = \alpha(\beta(x, a_1, \dots, a_\ell))$.

Требуется показать, что тогда автомат $A = \langle X, Q_A, \varphi_A \rangle$ гомоморфно представим некоторой $(\mathcal{I}_1, \dots, \mathcal{I}_\ell)$ - сетью из ℓ автоматов с $n_1, \dots, n_\ell$ состояниями соответственно, т.е. требуется показать, что выполняются следующие условия:

а) найдется некоторая сеть $B = \langle X, Q_1 \times \dots \times Q_\ell, \varphi_B \rangle$ из ℓ компонент $B_1 = \langle X \times Q_1 \times \dots \times Q_\ell, Q_1, \varphi_{B_1} \rangle, \dots, B_\ell = \langle X \times Q_1 \times \dots \times Q_{\ell-1}, Q_\ell, \varphi_{B_\ell} \rangle$ с $n_1, \dots, n_\ell$ состояниями соответственно такая, что для каждого $j=1, \dots, \ell$ функция $\varphi_{B_j}^j(x_0, q_1, \dots, q_\ell)$ отображения φ_{B_j} (или равная ей функция переходов $\varphi_{B_j}(x_0, (q_1, \dots, q_{j-1}, q_{j+1}, \dots, q_\ell), q_j)$ компоненты B_j) не зависит существенно от переменных с индексами из $\{0, 1, \dots, \ell\} \setminus \{j\}$;

б) найдется некоторая подсеть (подавтомат) $\Delta = \langle X, Q_\Delta, \varphi_\Delta \rangle$ сети B (т.е. подмножество $Q_\Delta \subseteq Q_1 \times \dots \times Q_\ell$ и отображение $\varphi_\Delta: X \times Q_\Delta \rightarrow Q_\Delta$) и сюръективное отображение $\mathcal{L}: Q_\Delta \rightarrow Q_A$ такое, что для любых $x \in X, (q_1, \dots, q_\ell) \in Q_\Delta$ имеет место $\varphi_A(x, \mathcal{L}(q_1, \dots, q_\ell)) = \mathcal{L}(\varphi_\Delta(x, (q_1, \dots, q_\ell)))$.

Покажем, что выполняется пункт а) доказываемого утверждения.

Рассмотрим сеть $B = \langle X, Q_1 \times \dots \times Q_\ell, \varphi_B \rangle$ из ℓ компонент $B_1 = \langle X \times Q_1 \times \dots \times Q_\ell, Q_1, \varphi_{B_1} \rangle, \dots, B_\ell = \langle X \times Q_1 \times \dots \times Q_{\ell-1}, Q_\ell, \varphi_{B_\ell} \rangle$ с $n_1, \dots, n_\ell$ состояниями соответственно, где $Q_1 = \mathcal{T}_{11}, \dots, Q_\ell = \mathcal{T}_{1\ell}$ (т.е. положим, что для каждого $j=1, \dots, \ell$ алфавит состояний компоненты B_j сети B совпадает с множеством обозначений блоков покрытия $\mathcal{T}_{1j}$ с n_j блоками) и $\varphi_{B_1} = \beta^1, \dots, \varphi_{B_\ell} = \beta^\ell$ (т.е. $B = \langle X, Q_1 \times \dots \times Q_\ell, \varphi_B \rangle = \langle X, \mathcal{T}_{11} \times \dots \times \mathcal{T}_{1\ell}, \beta \rangle$). Тогда нетрудно видеть, что согласно пункту I условия выполняется пункт а) доказываемого утверждения.

Покажем, что выполняется пункт б) доказываемого утверждения.

Пусть Q_Δ есть множество всех тех наборов состояний $(q_1, \dots, q_\ell) \in Q_1 \times \dots \times Q_\ell$ (наборов блоков $(q_1, \dots, q_\ell) \in \mathcal{T}_{11} \times \dots \times \mathcal{T}_{1\ell}$), для которых $\mathcal{L}(q_1, \dots, q_\ell) \neq \emptyset$, т.е. $\mathcal{L}(q_1, \dots, q_\ell) \in q_1 \cap \dots \cap q_\ell$, а φ_Δ есть часть отображения $\beta = \varphi_B$ с областью определения Q_Δ .

Очевидно, $Q_\Delta \subseteq Q_1 \times \dots \times Q_\ell$. Покажем, что $\varphi_\Delta(x, Q_\Delta) \subseteq Q_\Delta$, т.е. покажем, что отображение φ_Δ есть отображение вида $\varphi_\Delta: X \times Q_\Delta \rightarrow Q_\Delta$. Действительно, пусть $x \in X, (q_1, \dots, q_\ell) \in Q_\Delta$ и $(b_1, \dots, b_\ell) = \beta(x, (q_1, \dots, q_\ell)) = \varphi_B(x, (q_1, \dots, q_\ell)) = \varphi_\Delta(x, (q_1, \dots, q_\ell))$. Покажем, что $(b_1, \dots, b_\ell) \in Q_\Delta$. В самом деле, согласно пункту 3 условия доказываемой теоремы имеем $\varphi_A(x, \mathcal{L}(q_1, \dots, q_\ell)) = \mathcal{L}(\beta(x, q_1, \dots, q_\ell)) = \mathcal{L}(b_1, \dots, b_\ell)$. Далее, так как $\mathcal{L}(q_1, \dots, q_\ell) \neq \emptyset$, то $\varphi_A(x, \mathcal{L}(q_1, \dots, q_\ell)) \neq \emptyset$ и $\mathcal{L}(b_1, \dots, b_\ell) \neq \emptyset$,

т.е. $(q_1, \dots, q_\ell) \in Q_A$.

Таким образом, $\Delta = \langle X, Q_A, \psi_A \rangle$ есть подсеть сети $\mathcal{B} = \langle X, Q_1 \times \dots \times Q_\ell, \psi_B \rangle$.

Зададим сюръективное отображение $\mathcal{L}': Q_A \rightarrow Q_A$ такое, что для каждого $(q_1, \dots, q_\ell) \in Q_A$ $\mathcal{L}'(q_1, \dots, q_\ell) = \mathcal{L}(q_1, \dots, q_\ell)$ тогда согласно условию пункта 3 доказываемого утверждения для каждого $x \in X$, $(q_1, \dots, q_\ell) \in Q_A$ имеет место $\psi_A(x, \mathcal{L}'(q_1, \dots, q_\ell)) = \psi_A(x, \mathcal{L}(q_1, \dots, q_\ell)) = \mathcal{L}(\beta(x, (q_1, \dots, q_\ell))) = \mathcal{L}'(\psi_B(x, (q_1, \dots, q_\ell))) = \mathcal{L}'(\psi_A(x, (q_1, \dots, q_\ell)))$, т.е. показано, что выполняется пункт б) доказываемого утверждения. Достаточность доказана.

Необходимость. Пусть $A = \langle X, Q_A, \psi_A \rangle$ есть автомат с $n \geq 1$ состояниями; $n_1, \dots, n_\ell$ - натуральные числа; $\mathcal{I}_1, \dots, \mathcal{I}_\ell \subseteq \{0, 1, \dots, \ell\}$. Пусть A гомоморфно представим некоторой $(\mathcal{I}_1, \dots, \mathcal{I}_\ell)$ -сетью из ℓ автоматов с $n_1, \dots, n_\ell$ состояниями соответственно, т.е. пусть выполняются условия пунктов а) и б), сформулированные ранее при доказательстве достаточных условий доказываемой теоремы.

Требуется показать, что тогда существует $(\mathcal{I}_1, \dots, \mathcal{I}_\ell)$ - сохраняемая автоматом $A = \langle X, Q_A, \psi_A \rangle$ система из ℓ покрытий $\pi_1, \dots, \pi_\ell$ множества Q_A соответственно с $n_1, \dots, n_\ell$ блоками, т.е. требуется показать, что существует система из ℓ покрытий $\pi_1 = \{\pi_{11}, \dots, \pi_{1n_1}\}, \dots, \pi_\ell = \{\pi_{\ell 1}, \dots, \pi_{\ell n_\ell}\}$ множества Q_A соответственно с $n_1, \dots, n_\ell$ блоками, для которой выполняются условия пунктов 1, 2 и 3, сформулированные ранее, при доказательстве достаточных условий доказываемой теоремы.

Покажем, что выполняется пункт 1 доказываемого утверждения.

По множествам состояний $Q_1 = \{q_{11}, \dots, q_{1n_1}\}, \dots, Q_\ell = \{q_{\ell 1}, \dots, q_{\ell n_\ell}\}$ компонент $\mathcal{B}_1 = \langle X, Q_1 \times \dots \times Q_\ell, \psi_{B_1} \rangle, \dots, \mathcal{B}_\ell = \langle X, Q_1 \times \dots \times Q_\ell, \psi_{B_\ell} \rangle$ и по подсети $\Delta = \langle X, Q_A, \psi_A \rangle$ сети $\mathcal{B} = \langle X, Q_1 \times \dots \times Q_\ell, \psi_B \rangle$ построим покрытия $\pi_1 = \{\pi_{11}, \dots, \pi_{1n_1}\}, \dots, \pi_\ell = \{\pi_{\ell 1}, \dots, \pi_{\ell n_\ell}\}$ (блок π_{it} для каждого $i = 1, \dots, \ell$ и $t = 1, \dots, n_i$ считаем соответствующим состоянию q_{it} множества состояний Q_i и обозначаем через $\mathcal{F}(q_{it})$, и, наоборот, состояние q_{it} считаем соответствующим блоку π_{it} и обозначаем через $\mathcal{F}'(\pi_{it})$, возможно, в том числе и с пустыми блоками следующим образом: для каждого состояния $q \in Q_A$ рассмотрим все те наборы состояний $(s_1, \dots, s_\ell) \in Q_A \subseteq Q_1 \times \dots \times Q_\ell$, для которых $\mathcal{L}'(s_1, \dots, s_\ell) = q$, и соответствующие им наборы блоков $(\mathcal{F}(s_1), \dots, \mathcal{F}(s_\ell))$ и отнесем состояние q ко всем блокам $\mathcal{F}(s_1), \dots, \mathcal{F}(s_\ell)$.

Очевидно, что блок какого-либо покрытия, к которому не отнесется по указанной процедуре ни одно состояние из Q_A , будет пустым.

Зададим отображение $\beta: X \times T_1 \times \dots \times T_\ell \rightarrow T_1 \times \dots \times T_\ell$, соответствующее функции переходов $\psi_\beta: X \times Q_1 \times \dots \times Q_\ell \rightarrow Q_1 \times \dots \times Q_\ell$ сети $B = \langle X, Q_1 \times \dots \times Q_\ell, \psi_\beta \rangle$ следующим образом: для каждого $x \in X$ и $(a_1, \dots, a_\ell) \in T_1 \times \dots \times T_\ell$ имеет место $\beta(x, a_1, \dots, a_\ell) = (b_1, \dots, b_\ell) \in T_1 \times \dots \times T_\ell$ тогда и только тогда, когда $\psi_\beta(x, (F^{-1}(a_1), \dots, F^{-1}(a_\ell))) = (F^{-1}(b_1), \dots, F^{-1}(b_\ell))$.

Нетрудно видеть, что согласно пункту а) доказываемого утверждения и способа построения отображения β следует, что для каждого $j = 1, \dots, \ell$ проекция $\beta^j(x, y_1, \dots, y_\ell)$ отображения β не зависит существенно от переменных с индексами из множества $\{a_1, \dots, a_\ell\} \setminus y_j$.

Таким образом, показано, что выполняется пункт I доказываемого утверждения.

Покажем, что выполняется пункт 2 доказываемого утверждения.

Зададим отображение $\alpha: T_1 \times \dots \times T_\ell \rightarrow Q_A \cup \{\emptyset\}$ следующим образом: для каждого $(a_1, \dots, a_\ell) \in T_1 \times \dots \times T_\ell$ имеет место $\alpha(a_1, \dots, a_\ell) = q \in Q_A$ тогда и только тогда, когда $(F^{-1}(a_1), \dots, F^{-1}(a_\ell)) \in Q_A$, $\alpha'(F^{-1}(a_1), \dots, F^{-1}(a_\ell)) = q$ и $\alpha(a_1, \dots, a_\ell) = \emptyset$ — в противном случае.

Из способа построения отображения α нетрудно видеть, что для каждого $(a_1, \dots, a_\ell) \in T_1 \times \dots \times T_\ell$, либо $\alpha(a_1, \dots, a_\ell) \in a_1 \cap \dots \cap a_\ell$, либо $\alpha(a_1, \dots, a_\ell) = \emptyset$, т.е. выполняется пункт 2 доказываемого утверждения.

Покажем, что выполняется пункт 3 доказываемого утверждения.

Пусть $x \in X$ и $(a_1, \dots, a_\ell) \in T_1 \times \dots \times T_\ell$.

Рассмотрим случай, когда $(F^{-1}(a_1), \dots, F^{-1}(a_\ell)) \in Q_A$. Согласно пункту б) доказываемого утверждения имеем $\psi_A(x, \alpha'(F^{-1}(a_1), \dots, F^{-1}(a_\ell))) = \alpha'(\psi_A(x, (F^{-1}(a_1), \dots, F^{-1}(a_\ell))))$. Рассмотрим $(b_1, \dots, b_\ell) = \beta(x, (a_1, \dots, a_\ell))$. Согласно определению отображения $\beta: X \times T_1 \times \dots \times T_\ell \rightarrow T_1 \times \dots \times T_\ell$ имеем $(F^{-1}(b_1), \dots, F^{-1}(b_\ell)) = \psi_\beta(x, (F^{-1}(a_1), \dots, F^{-1}(a_\ell)))$. Согласно определению отображения $\alpha: T_1 \times \dots \times T_\ell \rightarrow Q_A \cup \{\emptyset\}$ имеем $\alpha'(F^{-1}(a_1), \dots, F^{-1}(a_\ell)) = \alpha(a_1, \dots, a_\ell)$ и $\alpha'(F^{-1}(b_1), \dots, F^{-1}(b_\ell)) = \alpha(b_1, \dots, b_\ell)$. Тогда $\psi_A(x, \alpha(a_1, \dots, a_\ell)) = \psi_A(x, \alpha'(F^{-1}(a_1), \dots, F^{-1}(a_\ell))) = \alpha'(\psi_A(x, (F^{-1}(a_1), \dots, F^{-1}(a_\ell)))) = \alpha'(F^{-1}(b_1), \dots, F^{-1}(b_\ell)) = \alpha(b_1, \dots, b_\ell) = \alpha(\beta(x, (a_1, \dots, a_\ell)))$, что и требовалось доказать.

Рассмотрим случай, когда $(F^{-1}(a_1), \dots, F^{-1}(a_\ell)) \notin Q_A$. Согласно определению отображения $\alpha: T_1 \times \dots \times T_\ell \rightarrow Q_A \cup \{\emptyset\}$ имеем $\alpha(a_1, \dots, a_\ell) = \emptyset$, приняв, что для любых $q \in Q_A \cup \{\emptyset\}$, $x \in X$ имеет место $q = \psi(x, \emptyset)$, получим $\alpha(\beta(x, (a_1, \dots, a_\ell))) = \psi(x, \emptyset) = \psi_A(x, \alpha(a_1, \dots, a_\ell))$, что и

требовалось доказать. Таким образом, показано, что выполняется пункт 3 доказываемого утверждения. Необходимость доказана. Теорема 2 доказана.

5. Некоторые следствия

В заключение, в качестве иллюстрации полученных результатов, рассмотрим некоторые частные случаи приведенных теорем.

Например, рассмотрим следствия (частные случаи) теорем 1 и 2 для случая представимости исходного автомата двухкомпонентной параллельной сетью.

Следствие 1. $(\{0,1\}, \{0,2\})$. Пусть $A = \langle X, Q_A, \psi_A \rangle$ есть автомат с $n > 1$ состояниями; n_1, n_2 — натуральные числа. Тогда для того чтобы автомат A был изоморфно представим параллельной сетью (т.е. $(\{q_1\}, \{q_2\})$ — сетью) из двух автоматов с n_1 и n_2 состояниями соответственно, необходимо и достаточно существования $(\{0,1\}, \{0,2\})_1$ — сохраняемой автоматом A системы из двух разбиений R_1 и R_2 множества Q_A соответственно с n_1 и n_2 блоками, для которой $R_1 \times R_2 = 0$.

Следствие 2. $(\{0,1\}, \{0,2\})$. Пусть $A = \langle X, Q_A, \psi_A \rangle$ есть автомат с $n > 1$ состояниями; n_1, n_2 — натуральные числа. Тогда для того чтобы автомат A был гомоморфно представим параллельной сетью (т.е. $(\{q_1\}, \{q_2\})$ — сетью) из двух автоматов с n_1 и n_2 состояниями соответственно, необходимо и достаточно существования $(\{0,1\}, \{0,2\})_L$ — сохраняемой автоматом A системы из двух покрытий π_1 и π_2 множества Q_A соответственно с n_1 и n_2 блоками.

Далее, например, условия гомоморфной представимости исходного автомата параллельной сетью из двух автоматов, каждый из которых содержит меньшее число состояний, чем исходный автомат, дает следующее следствие (частный случай) теоремы 2.

Следствие 2'. $(\{0,1\}, \{0,2\})$. Пусть $A = \langle X, Q_A, \psi_A \rangle$ есть автомат с $n \geq 2$ состояниями; n_1, n_2 — натуральные числа и $n_1 < n, n_2 < n$. Тогда для того чтобы автомат A был гомоморфно представим параллельной сетью (т.е. $(\{q_1\}, \{q_2\})$ — сетью) из двух автоматов с n_1 и n_2 состояниями необходимо и достаточно существования $(\{0,1\}, \{0,2\})_2$ — сохраняемой автоматом A системы из двух покрытий π_1, π_2 множества Q_A соответственно с n_1 и n_2 блоками.

Л и т е р а т у р а

1. Hartmanis J., Stearns R.E. *The Algebraic Structure Theory of Sequential Machines*, Prentice-Hall, Inc. Englewood Cliff, New York, 1966-210 p.
2. Агисалов Г.П., Евтушенко Н.В. *Декомпозиция конечных автоматов*. - Томск: Изд-во Томского ун-та, 1985. - 128 с.

В.Л.Павлов

СИНТЕЗ КОМБИНАЦИОННЫХ СХЕМ В ПРОИЗВОЛЬНОМ БАЗИСЕ

Излагается декомпозиционный метод синтеза комбинационных схем, свободных от статических составаний, с параметрическим описанием компонент разложения. Развиваемый подход является обобщением метода, описанного в [1,2], и может быть использован при синтезе управляющих схем на базе матричных БИС.

Пусть логические элементы произвольного базисного набора имеют собственные булевы функции $g_i \in G$, возможно, недоопределенные, зависящие от переменных $Z = \{z_1, z_2, \dots, z_k\}$ и заданные на интервалах из множеств $U_{g_i}^0, U_{g_i}^1$, $i = \overline{1, \ell}$. Условимся, что при входных изменениях, лежащих внутри интервалов из этих множеств, конструкция элементов обеспечивает отсутствие статических составаний на выходе.

Систему $F = \{f_1, f_2, \dots, f_m\}$ булевых функций от переменных $X = \{x_1, x_2, \dots, x_n\}$ требуется реализовать комбинационной схемой из элементов набора G так, чтобы каждая функция $f_i \in F$ была реализована без статических составаний при любых входных изменениях, лежащих внутри интервалов ее задания $U_{f_i}^0, U_{f_i}^1 \subseteq U_F$.

Метод предусматривает поочередную реализацию функций системы F на базе построенной части схемы. К моменту реализации $f(X) \in F$ на полюсах построенной части схемы может быть доступно множество Φ булевых функций $\varphi(X)$, реализованных без статических составаний на интервалах из $U_\varphi^0, U_\varphi^1 \subseteq U_F$. Если $|U_\varphi^0| = n^\sigma$, то семейство $\mathcal{H}_f$ функций, реализующих $f(X)$, можно задать параметрически: для любой $h \in \mathcal{H}_f$ $|U_h^0 \cap U_\varphi^0| \geq n^\sigma$, $\sigma = 0, 1$. Если $\Phi \cap \mathcal{H}_f \neq \emptyset$, то $f(X)$ реализована на одном из полюсов уже имеющейся схемы, в противном случае выбирается функция $g \in G$ и функция $f(X)$ подвергается разложению, свойство которого уточняет

Тесрема. Разложение $g(h_1, h_2, \dots, h_k)$ реализует $f(X)$ без составаний на интервалах из U_f^0, U_f^1 , если значения функций $h_1, h_2, \dots, h_k$ на любом интервале $u \in U_f^0$ образуют триничный набор, принадлежащий какому-либо интервалу из U_g^0, U_g^1 , $\sigma = 0, 1$, и каждая компонента $h_i(X)$ реализована без составаний на интервалах из $U_{h_i}^0, U_{h_i}^1 \subseteq U_f$.

Таким образом, правильно выбранная система $\psi = \{h_1, h_2, \dots, h_k\}$

компонент разложения сопоставляет каждому интервалу $u \in U_j^\sigma$ интервал $v \in V_j^\sigma$, $\sigma = 0, 1$, т.е. определяет отображение $U_j^\sigma \rightarrow V_j^\sigma$, $\sigma = 0, 1$. Разумное ограничение свойств этого отображения, в свою очередь, повлияет на местоположение и мощность областей определения компонент разложения, причем можно ввести систему параметров, описывающих семейства возможных компонент. Параметрическое описание позволяет получать и разлагать далее не конкретные компоненты разложения, а их параметры, что создает предпосылки для оптимизации синтезируемых схем.

Например, пусть $|U_j^\sigma| = m^\sigma$, $|U_j| = n^\sigma$, и в множестве трюичных векторов, соответствующих интервалам из V_j^σ содержится $m_{i\sigma}$ векторов, имеющих значение ν в i -й позиции, $\nu, \sigma = 0, 1; i = \overline{1, k}$. Пусть также отображение ψ таково, что для всякого $v \in V_j^\sigma$ множества $\psi^{-1}(v)$ примерно равноможны для каждого $\sigma = 0, 1$. В таком случае естественно потребовать, чтобы компонента h_i принимала значение ν на части множества U_j^σ , пропорциональной $m_{i\sigma}/m^\sigma$. В итоге, семейство $\mathcal{H}_i$ функций, пригодных в качестве h_i , задается описанием: для любой $h \in \mathcal{H}_i$

$$|U_h \cap U_j^\sigma| \geq \alpha_{i\sigma}^{\nu}, \quad |U_h \cap U_j^\sigma| \geq \alpha_{i1}^{\nu}, \quad (1)$$

где $\alpha_{i\sigma}^{\nu} = n^\sigma \frac{m_{i\sigma}}{m^\sigma}$, $\nu, \sigma = 0, 1, i = \overline{1, k}$.

Имея описание компонент разложения, можно попытаться конкретизировать их путем подстановки функций, реализованных на полюсах построенной части схемы и образующих множество Φ . Конкретизацию следует начинать с компоненты h_i , заданной наиболее узкими границами (сумма коэффициентов α из (1) наибольшая), и при условии $\Phi \cap \mathcal{H}_i \neq \emptyset$. Вместо h_i подставляется наилучшая по некоторому критерию функция из $\Phi \cap \mathcal{H}_i$, соответствующий ей полюс присоединяется к i -му полюсу элемента g , после чего описания остальных компонент уточняются. Если таким образом удастся последовательно доопределить все компоненты, то $f(x)$ реализована на выходном полюсе элемента g , в противном случае встретится компонента, которую нельзя конкретизировать на данном ярусе, и ее, или вернее ее описание, следует разлагать дальше.

Пусть, например, с самого начала не доопределяется компонента h_l , заданная границами $\alpha_{l\nu}^{\nu}$, вычисленными в (1) для $i = l$. Если разлагать h_l по той же функции g , то описание компоненты h_{lj} , $j = \overline{1, k}$, этого разложения будет задано неравенствами (1), с новыми границами $\beta_{j\nu}^{\nu}$, выраженными через прежние границы $\alpha_{l\nu}^{\nu}$ сле-

дующим образом:

$$\beta_{j0}^{\sigma} = \alpha_{l_0}^{\sigma} \frac{m_{j0}^1}{m^0} + \alpha_{l_1}^{\sigma} \frac{m_{j0}^1}{m^1}; \quad \beta_{j1}^{\sigma} = \alpha_{l_0}^{\sigma} \frac{m_{j1}^0}{m^0} + \alpha_{l_1}^{\sigma} \frac{m_{j1}^1}{m^1} \quad (2)$$

Допустим, что в соответствии с ограничениями (1) конкретизированы компоненты $h_1, h_2, \dots, h_i$, установим ограничения на область определения компоненты h_{i+1} . Множества U_f^{σ} распадаются на непересекающиеся подмножества $U_f^{\sigma c}$: на каждом интервале из $U_f^{\sigma c}$ компоненты $h_1, h_2, \dots, h_i$ принимают значение $c = c_1, c_2, \dots, c_i$, где $c_k \in \{0, 1\}$. При конкретизации остальных компонент интервалам из $U_f^{\sigma c}$ могут быть сопоставлены лишь те интервалы из U_g^{σ} , чьи наборы в первых i компонентах содержат набор c . Пусть $m_g^{\sigma/c}$ есть множество таких наборов. Через $m_g^{\sigma/c, v} \subseteq m_g^{\sigma/c}$ обозначим множество наборов, которые имеют, кроме того, значение v в $i+1$ -й компоненте, $v \in \{0, 1\}$. Ограничения для семейства U_{i+1} компонент h_{i+1} будут такими:

$$U_{h_{i+1}}^{\delta} = U_f^{\sigma c} = U_f^{\sigma} \cap \bigcap_{j=1}^i U_{h_j}^{c_j}, \quad \delta = \sigma c,$$

$$\alpha_{i+1, v}^{\delta} = \alpha_{i+1, v}^{\sigma c} = n^{\sigma} \frac{|U_f^{\sigma c}|}{|U_f^{\sigma}|} \cdot \frac{|m_g^{\sigma/c, v}|}{|m_g^{\sigma/c}|},$$

$$h \in U_{i+1}, \text{ если } |U_h^v \cap U_{h_{i+1}}^{\delta}| \geq \alpha_{i+1, v}^{\delta}.$$

Теперь могут быть легко поняты выкладки, соответствующие произвольному шагу синтеза схемы. Границы для области определения функции $f(X)$ задаются совокупностью попарно непересекающихся множеств U_f^{σ} , $\sigma \in I_f$, и неотрицательными числами α_v^{σ} , причем для выходного полюса схемы $U_f^{\sigma} = U_f^{\sigma}$, $\sigma = I_f = \{0, 1\}$,

$$\alpha_v^{\sigma} = \begin{cases} |U_f^{\sigma}|, & \text{если } \sigma = v, \\ 0, & \text{в противном случае.} \end{cases}$$

Считается, что функция h реализует $f(X)$, если

$$|U_h^v \cap U_f^{\sigma}| \geq \alpha_v^{\sigma}; \quad \sigma \in I_f, \quad v \in \{0, 1\}.$$

Границы для области определения $i+1$ -й компоненты разложения функции f по функции g вычисляются следующим образом:

$$U_{h_{i+1}}^{(\delta)} = U_f^{(\sigma)} \cap \bigcap_{j=1}^n U_{h_j}^{c_j}; \quad \delta = \sigma c = \sigma c_1 c_2 \dots c_i \in I_{h_{i+1}},$$

$$\alpha_{i+1, \nu}^{\delta} = \sum_{d \in \{0,1\}} \alpha_d^{\sigma} \frac{|U_{h_{i+1}}^{(\delta)}|}{|U_f^{(\sigma)}|} \cdot \frac{|m_d^{d/c \nu}|}{|m_g^{d/c}|}.$$

Метод легко обобщается на случай многозначных переключательных функций, что показано в [3].

В качестве примера реализуем функцию десяти переменных из работы [4], где она реализована суперпозицией

$$f(x_1, x_2, \dots, x_{10}) = \varphi(\varphi(\bar{x}_1, x_2, 1, \varphi(1, x_1, x_5, x_{10})), \bar{x}_3, 1, x_9).$$

Здесь $\varphi(a, b, c, d)$ - функция, реализуемая элементом схемы, представленная в табл.1.

Таблица 1

		a	b	c	d
φ^1	1	0	-	-	-
	2	-	1	0	1
	3	-	1	1	0
φ^0	1	1	0	-	-
	2	1	-	0	0
	3	1	-	1	1

Функция $f(x_1, x_2, \dots, x_{10})$ задана в левой части табл.2.

Таблица 2

	x_{10}	9	8	7	6	5	4	3	2	1	$f(\varphi, \bar{x}_1, x_2, \bar{x}_3)$	$\varphi_1(\bar{x}_1, x_2, 1, \bar{x}_3)$
U_f^1	1	0	0	0	1	0	0	0	0	1	0	0 1 0
	2	0	0	0	1	1	1	1	0	1	1	0 1 1 1 1
	3	0	0	1	0	1	1	1	0	1	0	0 1 1 1 1
	4	0	0	1	0	0	1	1	0	0	1	0 1 0
	5	0	1	0	1	1	1	0	0	1	0	0 1 0
	6	0	1	1	0	1	1	1	0	1	0	1 0 0
	7	0	1	1	1	1	0	0	0	1	0	0 1 0
	8	1	0	0	1	-	0	0	1	0	1	1 0 0
	9	1	0	1	0	1	0	1	0	0	0	0 1 1 1 1
	10	1	0	1	1	0	0	0	0	0	0	0 1 0
	11	1	0	0	1	1	0	0	0	0	0	0 1 0
U_f^0	1	0	1	0	1	1	0	0	1	1	0	1 0
	2	0	1	0	1	1	1	1	0	0	0	1 1 1 1 0
	3	0	1	1	0	0	0	0	1	1	1	1 0
	4	0	1	1	0	1	0	1	0	0	0	1 1 1 1 0
	5	0	1	1	1	1	1	1	1	1	1	1 0
	6	1	0	0	0	1	0	1	1	1	1	1 0
	7	1	0	0	1	1	1	0	1	0	1	1 0
	8	1	1	1	0	0	0	0	1	1	0	1 0

Непосредственно из табл.1 видно, что первая компонента разложения $f(x)$ по φ должна принимать единичное значение на всех интервалах из U_f^0 (во всех наборах из V_φ^0 первая компонента равна 1). Среди функций $x_1, x_2, \dots, x_{10}$, представленных столбцами в левой части табл.2, подходящих нет, поэтому первая компонента разложения $f(x)$ по φ может быть получена как результат более глубокого разложения. В нашем случае в качестве этой компоненты будет принята функция φ_1 из табл.2, которая должна принимать значение 1 на всех интервалах из U_f^1 и значение 0 не менее чем на трети интервалов из U_f^1 . Первая компонента разложения функции φ_1 должна принять в соответствии с соотношениями (2) значение 1 не менее чем на трети интервалов из U_f^1 и значение 0 не менее чем на трети интервалов из U_f^0 . В качестве этой компоненты лучше других подходит функция $\bar{x}_3$. Теперь на всех интервалах из U_f^1 , кроме интервалов u_2, u_4 , обеспечено значение $\varphi_1 = 1$, интервалам же u_2, u_4 при конкретизации оставшихся компонент должны быть сопоставлены наборы $v_2, v_3 \in V_\varphi^1$, имеющие в качестве второй компоненты 1. Таким образом, вторая компонента разложения функции φ_1 должна принять значение 1 на интервалах $u_2, u_4 \in U_f^0$ и значение 0 на возможно большем числе интервалов из U_f^1 , на которых компонента $\bar{x}_3$ приняла значение 1. В качестве второй компоненты лучше всего подходят функции x_6, x_9 , мы выберем x_6 . Для конкретизации третьей и четвертой компонент разложения функции φ_1 есть несколько возможностей. Легко видеть, что на место этих компонент годятся соответственно константа 1 и функция $\bar{x}_9$. В результате φ_1 примет вид, показанный в правой части табл.2.

В качестве второй компоненты разложения $f(x)$ должна быть выбрана функция, принимающая значение 1 на интервалах $u_6, u_8 \in U_f^1$ и значение 0 не менее чем на трети интервалов из U_f^1 . Лучше других для этого подходит функция $\bar{x}_2$.

Третья и четвертая компоненты конкретизируются соответственно как x_8 и $\bar{x}_9$.

В итоге $f(x)$ реализована на двух элементах вместо трех:

$$f(x_1, x_2, \dots, x_{10}) = \varphi(\varphi(\bar{x}_3, x_6, 1, \bar{x}_9), \bar{x}_2, x_8, \bar{x}_9).$$

В качестве второго примера приведем реализацию функции из табл.3 схемой из элементов, реализующих функцию $g(z_1, z_2, z_3) = z_1 z_2 \vee \bar{z}_2 \bar{z}_3$. Этот пример взят из работы [5], где получена схема из восьми элементов.

Таблица 3

	x	1	2	3	4
u_j		0	0	0	0
		0	0	0	1
		0	0	1	1
		0	1	1	0
		1	0	1	0
		1	1	0	0
		1	1	1	1

	x	1	2	3	4
u_j		0	0	1	0
		0	1	0	0
		0	1	1	1
		1	0	0	0
		1	0	1	1
		1	1	0	1
		1	1	1	0

Предлагаемым методом получена суперпозиция, соответствующая схеме из четырех элементов:

$$f(x_1, x_2, x_3, x_4) = g(x_1, g(\bar{x}_2, g(x_4, x_3, \bar{x}_1)), g(x_2, \bar{x}_3, x_1)), \bar{x}_1).$$

Литература

1. Павлов В.Л. О синтезе логических схем из элементов „ИЛИ-НЕ“ с ограниченным числом входов. - В кн.: Вычислительная техника. - Каунас: Каунасский политехнический ин-т, 1971, т.2, с. 219-223.
2. Агибалов Г.П., Комаров Ю.М., Липский В.Б. Синтез комбинационных схем, свободных от статических состязаний. - Автоматика и вычислительная техника, 1979, № 3, с. 1-6.
3. Агибалов Г.П., Бузанов В.А., Липский В.Б., Румянцев Б.Ф. Логическое проектирование переключательных автоматов. - Томск: Изд-во Томского ун-та, 1983. - 154 с.
4. Гаврилов М.А., Копыленко В.М. Метод „переходных таблиц“ синтеза многовыходных комбинационных структур на произвольных элементах. - В кн.: Абстрактная и структурная теория релейных устройств. - М.: Наука, 1972, с. 7-32.
5. Бивол Л.Г. О реализации булевых функций на произвольных элементах. - В кн.: Абстрактная и структурная теория релейных устройств. - М.: Наука, 1972, с. 32-38.

И.А.Панкратова

СИНТЕЗ КОМБИНАЦИОННЫХ ФУНКЦИОНАЛЬНЫХ СХЕМ В МНОГОЗНАЧНОЙ ЛОГИКЕ

1. Основные понятия и определения. Метод синтеза

Функциональные схемы, рассматриваемые в данной работе, являются дискретными моделями интегральных схем, выполненных по КМОП-технологии, введенными в [1]. Каждому узлу КМОП-схемы ставится в соответствие пара проводимостей ab , где a и b - проводимости схемы от данного узла до полюсов "Земля" и "Питание" соответственно. Значениями a и b могут быть 0 (цепь разомкнута), 1 (цепь замкнута) и Φ (неопределенное значение). Таким образом, $ab \in M = \{00, 01, 0\Phi, 10, 11, 1\Phi, \Phi 0, \Phi 1, \Phi\Phi\}$. Однако на практике часто нет необходимости различать все пары проводимостей из M , так как разные пары проводимостей могут интерпретироваться одинаковыми физическими состояниями. В этой связи на множестве M задается некоторое разбиение R . Все пары проводимостей, входящие в один блок, считаются эквивалентными и представляющими некоторое одно и то же состояние.

Пусть S - множество состояний, задаваемое разбиением R . Функцией состояний от n переменных назовем отображение $f: S^n \rightarrow S$, а функциональным элементом - $(n, 1)$ -полюсник, входам которого сопоставлены переменные $x_1, \dots, x_n$ со значениями из S , а на выходе реализуется некоторая функция состояний $f(x_1, \dots, x_n)$. Функциональные схемы строятся из функциональных элементов путем отождествления некоторых их полюсов. Функциональная схема называется комбинационной, если для всех ее элементов в ней нет путей передачи информации от выхода элемента к его входам. Функционирование такой схемы заключается в реализации соответствующей ей функции состояний, которая получается в результате суперпозиции функций элементов схемы.

Задаваясь различными разбиениями множества M , можно получать модели реальных схем с равной степенью точности и с равной интерпретацией состояний. Количество блоков разбиения R определяет значимость используемой в операциях анализа и синтеза схем логики.

Задача синтеза комбинационной функциональной схемы состоит в построении из элементов заданного базиса B схемы S , на выходах

которой реализуются функции состояний $f_i \in F, i = \overline{1, m}$. При этом желательно получение как можно более простых схем, т.е. схем, состоящих из возможно меньшего числа элементов. На упрощение получаемых схем ориентирован декомпозиционный метод В.Л.Павлова, разработанный для реализации функциональными схемами сначала частичных булевых функций [2], а затем 7 -значных функций состояний [1].

По методу Павлова схема строится последовательно, от входов ее к входам, путем упрощения задаваемых для реализации функций. На очередном шаге синтеза функция состояний $f \in F$ отождествляется, если это возможно, с одной из входных переменных схемы или с уже реализованной в построенной части схемы функцией. В противном случае выбирается элемент $k \in B$, по функции которого производится разложение функции f . В отличие от традиционного декомпозиционного метода, в методе Павлова в момент разложения не определяются сразу все компоненты разложения $f_1, \dots, f_m$, а вычисляются ограничения, которым должна удовлетворять f_1 для обеспечения реализации функции f функцией элемента k . Из числа входных переменных и имеющихся в схеме функций выбирается функция g , удовлетворяющая этим ограничениям, и отождествляется с f_1 . После этого вычисляются ограничения на f_2 , и так далее, пока не будут получены все компоненты разложения. Если не найдется удовлетворяющей ограничениям на f_i функции g , то f_i сама подвергается разложению по функции некоторого базисного элемента. Таким образом, на каждом шаге синтеза генерируется не один, а целое семейство вариантов разложения, выбор из которых осуществляется вместе с построением подсхем, реализующих компоненты разложения.

Далее описывается программная реализация метода Павлова для синтеза функциональных схем с различной значностью применяемой логики.

2. Синтез схем из элементов „ИНЕ” в 3-значной логике

Пусть классами разбиения R являются: $0 = \{10\}$, $1 = \{0, 1\}$, $\Phi = \{00, 0\Phi, 11, 1\Phi, \Phi 0, \Phi 1, \Phi \Phi\}$. Тогда $C = \{0, 1, \Phi\}$. В этом случае функции состояний представляют собой частичные булевы функции, а функциональные элементы — обычные логические элементы.

Обозначим через $\tilde{f}$ множество всех доопределений частичной

булевой функции f и рассмотрим случай, когда базис B состоит из элемента „ИНЕ" с числом входов p и из его частного случая - инвертора. Введем в рассмотрение совокупность $\Phi = \{\varphi_i\}$ функций, реализованных в построенной части схемы к очередному шагу синтеза. Перед начальным шагом $\Phi = \{x_1, \dots, x_n, \bar{x}_1, \dots, \bar{x}_n, 0, 1\}$. Алгоритм синтеза комбинационной функции начальной схемы методом Павлова выглядит в этом случае следующим образом.

1. Выбираем очередную функцию $f \in F$. Она задана множествами интервалов U_f^0, U_f^1 . Ищем среди функций Φ функцию φ_i , являющуюся доопределением f . В случае успеха отождествляем узлы схемы, соответствующие f и φ_i ; переход к 8. Если не существует $\varphi_i \in \Phi$, то 2.

2. Ищем разложение $\tilde{f} \geq \varphi_1 \wedge \varphi_2 \wedge \dots \wedge \varphi_k, k \leq p$, так что $\tilde{f} \geq (\varphi_1 \wedge \tilde{f}_1), \tilde{f}_1 \geq (\varphi_2 \wedge \tilde{f}_2), \dots, \tilde{f}_{i-1} \geq (\varphi_i \wedge \tilde{f}_i), \dots$. С этой целью положим $f_0 = f, n_0^0 = |U_f^0|, n_0^1 = |U_f^1|$ и выполним пп. 3-5.

3. Положим $i = 1$.

4. Разложение $\tilde{f}_{i-1} \geq (\varphi_i \wedge \tilde{f}_i)$ осуществимо, если существует $\varphi_i \in \Phi$, удовлетворяющая следующим условиям:

$$|U_{f_{i-1}}^0 \cap M_{\varphi_i}^1| \geq \alpha_i; |U_{f_{i-1}}^1 \cap M_{\varphi_i}^0| \geq \beta_i, \quad (1)$$

$$\text{где } \alpha_i = n_{i-1}^0; \beta_i = \frac{n_{i-1}^1}{p-i+1}.$$

Если не существует $\varphi_i \in \Phi$, удовлетворяющей (1), то переход к 7. Из всех функций в Φ , удовлетворяющих (1), выбираем φ_i , дающую наибольшую разность $n_i^0 - n_i^1$ (так как в процессе последовательного разложения стремимся к возможности доопределения функции f_k до константы 0), и переходим к 5.

5. Вычисляется область определения функции f_i :

$$U_{f_i}^0 = U_{f_{i-1}}^0 \cap M_{\varphi_i}^1; U_{f_i}^1 = U_{f_{i-1}}^1 \setminus (U_{f_{i-1}}^1 \cap M_{\varphi_i}^0);$$

$$n_i^0 = n_{i-1}^0; n_i^1 = n_{i-1}^1 - |U_{f_{i-1}}^1 \cap M_{\varphi_i}^0|.$$

Если $n_i^1 > 0$, то $i := i + 1$, переход к 4. Иначе 6.

6. Определена последняя компонента разложения $\tilde{f} \geq \varphi_1 \wedge \varphi_2 \wedge \dots \wedge \varphi_k = \varphi, k \in p$. Это разложение может иметь избыточные компоненты, которые устраняются следующим образом.

Построим матрицу значений функций $\varphi_i, i = \overline{1, k}$, на интервалах из $U_f^1 \cap M_{\varphi}^1$. Строки матрицы соответствуют интервалам, столбцы - функциям. Найдем кратчайшее столбцовое покрытие $\{i_1, \dots, i_z\}$ отрицания этой матрицы. Здесь $z \leq k$. Построим функцию $\varphi' = \varphi_{i_1} \wedge \dots \wedge \varphi_{i_z}$.

Очевидно, что $U_j^1 \cap M_{\varphi^1} = U_j^1 \cap M_{\varphi^2}$ и $(U_j^0 \cap M_{\varphi^1}) \supseteq (U_j^0 \cap M_{\varphi^2})$, т.е. φ^1 (при $\kappa < k$) — более простая, чем φ , реализация f , при этом, возможно, и более полная.

Подключаем ветвь, реализующую φ^1 , к схеме. Функции φ^1 и $\bar{\varphi}^1$ включаем в множество Φ . Если $f \in F$, переход к 8. Иначе извлекаем значение i из стека номеров компонент; если $i \neq p$, переход к 5, иначе 6.

7. На некотором шаге не нашли $\varphi_i \in \Phi$, удовлетворяющую условию реализации (1). Тогда подвергаем разложению по функции р-входового элемента "ИНЕ" функцию $\bar{f}_{i-1}$. Для этого положим $U_{j_0}^0 = U_{j_{i-1}}^1$; $U_{j_0}^1 = U_{j_{i-1}}^0$; $\pi_0^0 = \beta_i$; $\pi_0^1 = \alpha_i$. Записываем значение i в стек номеров компонент; переход к 3.

8. Если совокупность F не исчерпана, переход к 1. Иначе конец.

Описанный алгоритм реализован программно на языке ЛЯПАС-М. Программа состоит из 30 модулей, расположенных на 7 уровнях иерархии, и использует М-сегмент поиска кратчайшего покрытия булевой матрицы [3]. Головная макрооперация — АЛГОРИТМ(). Она вводит с перфокарт, печатает и использует следующие данные:

- 1) Тройчная матрица $A_{n \times k}$ значений переменных $x_1, \dots, x_n$ на заданных интервалах $u_1, \dots, u_k$;
- 2) Тройчная матрица $B_{m \times k}$ значений функций $f_1, \dots, f_m$ на интервалах $u_1, \dots, u_k$;
- 3) натуральное число $a = 1 \text{ к } 521$;
- 4) натуральное число p — количество входов базового элемента "ИНЕ".

Результатом работы макрооперации является печать описания полученной схемы S на языке СЛОГ [1]. Базис B является декомпозиционно полным, поэтому процесс всегда сходится.

Заметим, что эту же программу можно использовать для синтеза схем из элементов "ЛИНЕ", учитывая формулу двойственности:

$$f^*(x_1, \dots, x_n) = \bar{f}(\bar{x}_1, \dots, \bar{x}_n).$$

3. Синтез схем в 6-значной логике

Пусть разбиение R имеет следующие классы: $1 = \{01\}$, $0 = \{10\}$, $z = \{00\}$, $\bar{z} = \{0\bar{0}\}$, $\odot = \{0\bar{0}\}$, $A = \{11, 1\bar{0}, \bar{0}1, \bar{0}\bar{0}\}$.

Тогда алфавит состояний есть $C = \{1, 0, z, \bar{z}, \odot, A\}$. Пусть базис B является частью следующего множества элементов [1]:

Таблица 1

	1111100000ZZZZZZZZZZ
	10ZZZZA0ZZZZAZZZAZZZAA
Проводное ИЛИ КМОП ИЛИ КМОП ИЛИ	1A11AA00A0AZZZAZAAZZAA 0000001ZZZZZZZZZZZZAA 01ZZZZA11111ZZZZZZZZAA

Символы алфавита С в матрицах А и В кодируются парами символов

лов согласно табл.2

Таблица 2

с	1	0	z	x	⊙	А
ког	1 ₁	0 ₁	x ₁	0 ₁	1 ₁	А ₁

Программа ищет в памяти ЭВМ описания элементов базиса и удаляет из списка базисных те элементы, описание которых отсутствует.

Выдается информация об этом: печатаются списки элементов "заданный базис" и "допустимый базис". Результатом работы макрооперации является печать описания полученной схемы на языке СЛОГ, либо сообщение: "Синтез не окончен".

Литература

1. Агibalов Г.П. и др. Логическое проектирование переключательных автоматов. - Томск: Изд-во Томского ун-та, 1983. - 152 с.
2. Павлов В.Л. О синтезе логических схем из элементов "ЛИИ-НЕ" с ограниченным числом входов - В кн.: Вычислительная техника. - Каунас: Каунасский политехнический ин-т, 1971, т.2, с.219-223.
3. Черкашин Ю.Д. Алгоритм поиска кратчайшего покрытия булевой матрицы на основе поиска покрытий заданной мощности. - В кн.: Автоматизация технической подготовки производства. Минск: ИИТ АН БССР, 1983, вып.3.

В.Е.Липский

СИСТЕМА ЛОГИЧЕСКОГО СИНТЕЗА ИНТЕРВАЛЬНЫХ АВТОМАТОВ - СИНТА

1. Введение

Система логического синтеза СИНТА предназначена для построения схем из элементов И, ИНЕ, ИЛИНЕ, реализующих заданные интервальные автоматы [1]. Программы, входящие в систему, решают следующие задачи:

- анализ интервального автомата на квазимонотонность,
- кодирование внутренних состояний,
- построение структурных функций,
- синтез комбинационной схемы из элементов заданного базиса,
- организация памяти схемы,
- вывод результатов синтеза на внешние носители.

Функционирование синтезируемой схемы задаётся с помощью интервального автомата, являющегося более общим понятием, чем конечный автомат. Интервальный автомат даёт возможность задавать различные способы функционирования схем: синхронный, асинхронный в статике (при неизменяющемся состоянии входных полюсов), и асинхронный в динамике (с учётом изменения состояния входных полюсов).

Программы написаны на языке ЛЯПАС и реализованы на ЭВМ типа ЕС и СМ-4. Область применения системы СИНТА ограничивается объёмом памяти ЭВМ и характеризуется следующими границами для параметров реализуемого автомата:

- число различных интервалов входных, внутренних и выходных состояний ≤ 256 ,
- число переходов ≤ 1000 .

Среднее время синтеза на ЭВМ ЕС-1022 комбинационной схемы из 100 элементов не превосходит 8 мин, а последовательностной схемы такой же сложности - 20 мин.

2. Задание функционирования логических схем

Различные способы функционирования логических схем задаются интервальным автоматом, представляющим собой, как и конечный автомат, пятёрку $\Omega = (X, Q, Y, \psi, \varphi)$, в которой X, Q, Y , как обычно, конечные алфавиты, называемые соответственно множествами входных, внутренних и выходных состояний, а ψ и φ - интервальные функции переходов и выходов, определённые и принимающие значения на интервалах - векторах, компонентами которых являются подмножества неко-

торых множеств. Пусть M есть декартово произведение некоторых множеств. Обозначим через $\tilde{M}$ множество всех непустых интервалов из M . Тогда функции переходов и выходов интервального автомата суть отображения $\psi: \tilde{X} \times \tilde{Q} \rightarrow \tilde{Q}$, $\varphi: \tilde{X} \times \tilde{Q} \rightarrow \tilde{Y}$.

Логическая схема представляет собой структуру, показанную на рис. I, где $x_1, \dots, x_n, y_1, \dots, y_m, z_1, \dots, z_r$, переменные соответственно входные, выходные, внутренние, ЛП - логический преобразователь (комбинационная схема), БП - блок памяти, представляющий собой набор задержек, триггеров или вентилях, управляемых синхроимпульсами. Вид блока памяти зависит от способа функционирования. Логической схеме сопоставляется интервальный автомат $\alpha = (E^n, E^r, E^m, \psi, \dots, \varphi_r, \varphi_1, \dots, \varphi_m)$, где $E = \{0, 1\}$, а $\psi, \dots, \varphi_r, \varphi_1, \dots, \varphi_m$ - интервальные функции, реализуемые логическим преобразователем на выходах $y_1, \dots, y_m, z_1, \dots, z_r$ соответственно.

Рис. I

Функционированием логической схемы называется множество переходов вида $(\alpha q, \beta s, \gamma)$, понимаемых следующим образом: значения входных переменных (интервал α) схемы, находящейся в состоянии q , определяемом набором значений её внутренних переменных, меняются на β ; при этом схема переходит в состояние s , а всевозможные векторы значений выходных переменных, возникающие в течение перехода, содержатся в γ .

Пусть $T = \{(\alpha q, \beta s, \gamma): \alpha, \beta \in E^n, \gamma \in E^m, q, s \in \{1, \dots, r\}, r \geq 2\}$ есть множество переходов. Тогда функционирование логической схемы задаётся интервальным автоматом $\alpha = (E^n, \{1, \dots, r\}, E^m, \psi, \varphi_1, \dots, \varphi_m)$, где функции $\psi, \varphi_1, \dots, \varphi_m$ зависят от способа функционирования. Для синхронного способа функционирования

$$\psi(\beta, q) = s, \quad \varphi_1(\beta, q) \dots \varphi_m(\beta, q) = \gamma$$

для всех $(\alpha q, \beta s, \gamma) \in T$. Для асинхронного в статике способа функционирования

$$\psi(\beta, q \cup s) = s, \quad \varphi_1(\beta, q \cup s) \dots \varphi_m(\beta, q \cup s) = \gamma$$

для всех $(\alpha q, \beta s, \gamma) \in T$. Для асинхронного в динамике способа функционирования

$$\psi(\alpha \cup \beta, q) = \langle t \rangle, \quad \psi(\beta, \langle t \rangle) = s, \\ \varphi_1(\alpha \cup \beta, \langle t \rangle) \dots \varphi_m(\alpha \cup \beta, \langle t \rangle) = \gamma$$

для всех $t = (\alpha q, \beta s, \gamma) \in T$, где $\langle t \rangle \subseteq \{1, \dots, r\}$ вычисляется по следующей итерационной процедуре: $\langle t \rangle_1 = q$,
 $\langle t \rangle_{i+1} = \langle t \rangle_i \cup \{s' : (\alpha q', \beta s', \gamma) \in T \wedge \beta s' \subseteq \alpha \cup \beta \wedge q' \subseteq \langle t \rangle_i\}$, $i = 1, \dots, p$,
 где p берётся из условия $\langle t \rangle_{p-1} = \langle t \rangle_p$; полагаем по определению
 $\langle t \rangle = \langle t \rangle_p$.

3. Метод синтеза

Определение. Интервальный автомат $\alpha = (X, Q, Y, \psi, \varphi)$ реализуется интервальным автоматом $\beta = (X, P, Y, \psi', \varphi')$, если существует отображение $\rho: Q \rightarrow P$ такое, что

$$\psi'(\alpha, \rho(q)) \subseteq \rho(\psi(\alpha, q)),$$

$$\varphi'(\alpha, \rho(q)) \subseteq \varphi(\alpha, q)$$

для всех $\alpha q \subseteq X \times Q$.

Задача синтеза ставится следующим образом: задан интервальный автомат $\alpha = (E^n, Q, E^m, \psi, \varphi_1 \dots \varphi_m)$, представляющий один из способов функционирования. Требуется из элементов функционально полного базиса построить логическую схему, автомат которой реализует α , и, кроме того, удовлетворяющую ограничению на число полюсов, подсоединённых к выходу одного элемента (ограничение на нагрузочную способность элементов).

Задача синтеза, как показано в [1], имеет решение, если функции автомата α квазимонотонны, т.е. удовлетворяют условию

$$\alpha \cap \beta \neq \emptyset \rightarrow f(\alpha) \cap f(\beta) \neq \emptyset.$$

В процессе синтеза решаются следующие задачи:

1) проверка функций переходов и выходов автомата α на квазимонотонность;

2) кодирование внутренних состояний автомата α , сохраняющее квазимонотонность функций переходов и выходов;

3) построение при выбранном кодировании интервальных функций, подлежащих реализации логическим преобразователем;

4) синтез логического преобразователя;

5) получение требуемой логической схемы путём замыкания соответствующих полюсов логического преобразователя через блок памяти;

6) выдача логической схемы и таблицы кодирования внутренних состояний автомата на печать, экран или в архив.

4. Представление входной и выходной информации

Языком представления интервального автомата и логической схемы является подмножество языка СЛОГ [1], расширенное средствами описания функционирования схем. Любое описание в этом языке есть цепочка символов, состоящая из заголовка - имени описываемого объекта, совокупности разделов, выделяемых ключевыми словами, и признака конца описания - ** (две звёздочки).

Описание логической схемы содержит пять разделов, открываемых ключевыми словами 'ВХОДНЫЕ', 'ВЫХОДНЫЕ', 'ВНУТРЕННИЕ', 'БЛОКИ' и 'СВЯЗИ'. Первые три раздела описывают соответственно входные, выходные и внутренние переменные схемы и имеют одинаковый вид P/n , где P есть символ "X", "Y" или "Z", а n - цифровая запись числа входных, выходных или внутренних переменных. Раздел 'ВНУТРЕННИЕ' отсутствует, если схема комбинационная.

Раздел 'БЛОКИ' состоит из фрагментов, разделённых символом ";". В каждом фрагменте перечислены элементы схемы, имеющие одинаковый тип: И, НЕИ, НЕИЛИ. Фрагмент открывается именем типа, далее в круглых скобках указывается число входов в элемент, затем следует символ ":", после чего через запятую перечисляются имена элементов в схеме, имеющих данный тип и число входов. Имя элемента состоит из символа "A" и цифровой записи его номера в схеме. Например: БЛОКИ И(2): A2, A4; НЕИЛИ(3): A1, A5; НЕИ(1): A3

Раздел 'СВЯЗИ' состоит из уравнений связи, задающих узлы схемы - списки отождествлённых полюсов. Уравнения связи упорядочены: сначала идут связи, содержащие входные полюсы схемы, далее - выходные, затем - внутренние и, наконец, прочие связи. В каждом уравнении связи через символ "=" перечислены имена полюсов, входящих в описываемый узел. Имя полюса элемента схемы состоит из имени этого элемента и, через точку, символа "Y", если это выходной полюс, или числа i , если это i -й вход элемента.

Описание интервального автомата состоит из разделов 'ВХОДНЫЕ', 'ВЫХОДНЫЕ', аналогичных соответствующим разделам в описании логической схемы, и раздела 'БАЗИСКА', задающего функции переходов ψ и выходов φ интервального автомата. Раздел 'БАЗИСКА' представляет собой совокупность строк, разделённых символом ";". Каждая строка имеет вид

$$\beta_1, \dots, \beta_n (N_1, \dots, N_q) / (M_1, \dots, M_s) \delta_1, \dots, \delta_m,$$

где $N_1, \dots, N_q, M_1, \dots, M_s$ - натуральные числа; n - число входных переменных; m - число выходных переменных; $\beta_1, \dots, \beta_n, \delta_1, \dots, \delta_m \in \{0, 1, -1\}$. Векторы $\beta_1, \dots, \beta_n, \delta_1, \dots, \delta_m$ задают соответственно интервалы α и γ

входных и выходных состояний автомата, в обозначениях $O = \{0\}$, $I = \{1\}$, $- = \{0, 1\}$, а множества $(N_1, \dots, N_q)(M_1, \dots, M_s)$ соответственно интервалы q и s внутренних состояний автомата такие, что

$$s = \psi(\alpha, q), \quad \gamma = \varphi(\alpha, q).$$

Описания схем и автоматов могут содержать пробелы и управляющие символы типа "возврат каретки" и "перевод строки" с целью их вывода на печать и экран в удобной для зрительного восприятия форме.

5. Работа с системой

Работа с системой СИНТА ведётся в диалоговом режиме через видеотерминал. Вызов системы осуществляется командой

*RUN LSOOI ,

в результате чего на экране появляется надпись

ЛОГИЧЕСКИЙ СИНТЕЗ ИНТЕРВАЛЬНЫХ АВТОМАТОВ

и 'СХЕМА', приглашающая ввести имя синтезируемой схемы. Предполагается, что под этим именем в архиве I2-программирующей системы ЛЯПАС находится СЛОГ-описание интервального автомата, задающего функционирование этой схемы. Если в архиве нет автомата с требуемым именем, то выводится надпись:

В АРХИВЕ I2 НЕТ АВТОМАТА С ЭТИМ ИМЕНЕМ

и 'СХЕМА', предлагая синтезировать другую схему, функционирование которой представлено в архиве I2.

В процессе синтеза возможны аварийные ситуации, сопровождаемые надписями:

НЕХВАТКА ПАМЯТИ ЭВМ,

ЗАДАННАЯ ФУНКЦИЯ НЕКВАЗИМОНОТОННА В СЛЕДУЮЩИХ ИНТЕРВАЛАХ:

i	И	j
k	И	l

где i, j, k, l , - натуральные числа, указывающие строки в разделе 'ФУНКЦИЯ', для которых не выполняется условие квазимонотонности,

НЕ ХВАТАЕТ ПАМЯТИ ДЛЯ ТАБЛИЦЫ УСЛОВИЙ,

ЧИСЛО ВНУТРЕННИХ ПЕРЕМЕННЫХ > 32 ,

НЕХВАТКА ПАМЯТИ ДЛЯ СТРУКТУРНЫХ ФУНКЦИЙ АВТОМАТА.

В таких случаях синтез прекращается, и система переходит к синте-

зу следующей схемы.

Если аварийных ситуаций не возникло, то пользователь должен выбрать режим синтеза, в ответ на вопросы

БАЗИС НЕИ?

ЧИСЛО ВХОДОВ В ЭЛЕМЕНТ?

НАГРУЗОЧНАЯ СПОСОБНОСТЬ?

вводя с экрана соответствующие сообщения: ДА, если базис синтеза содержит единственный элемент НЕИ; НЕТ, если синтез ведётся в расширенном базисе: И, НЕИ, НЕИЛИ, и конкретные числа входов и нагрузочной способности.

По окончании процесса синтеза выдаётся сообщение 'РЕЗУЛЬТАТ', позволяющий результат синтеза: таблицу кодирования состояний и схему вывести на экран или печать. Схему, кроме того, можно поместить в архив. Для этого надо набрать сообщение:

$TK > t. \text{СХЕМА} > t'$,

где $t, t' \in \{ЭК, ПЧ, АР\}$.

Для выхода из системы СИНТА в ответ на сообщение 'СХЕМА' следует набрать пустое сообщение.

6. Пример

В качестве примера рассмотрим синтез D-триггера - устройства, работа которого описывается вход-выходными последовательностями, приведёнными на рис.2. Устройство имеет два входа и один выход. Первый вход носит название информационного, второй - тактового.

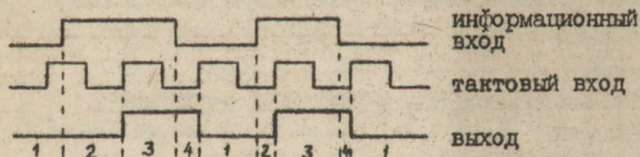


Рис.2

Автомат, описывающий работу D-триггера, имеет 4 состояния: 1(3) - состояние устойчивого 0 (соответственно 1): выход принимает значение 0(1) и следующий тактовый импульс его не меняет; 2(4) - состояние неустойчивого 0(1): выход принимает значение 0(1), но с приходом тактового импульса может измениться. Функционирование D-триггера состоит из множества T переходов, заданных таблицей

N	α	q	β	s	γ	$\langle t \rangle$
1	0 -	1	0 -	1	0	1
2	0 1	1	1 1	1	0	1
3	1 1	1	1 0	2	0	1,2,3,4
4	1 0	2	1 1	3	-	2,3
5	1 1	3	1 0	3	1	3
6	1 0	3	0 0	4	1	3,4
7	0 0	4	0 1	1	-	1,4
8	0 0	1	1 0	2	0	1,2,3,4
9	1 0	3	1 1	3	1	3
10	1 0	2	0 0	1	0	1,2,3,4
11	0 0	3	0 1	3	1	1,3,4
12	0 0	4	1 0	3	1	3,4

Интервальные автоматы ТСИНХР, ТАСТАТ и ТАДИН, задающие функционирование соответственно синхронное, асинхронное в статике и асинхронное в динамике, имеют следующие СЛОГ-описания.

ТСИНХР

'ВХОДНЫЕ' $X/2$

'ВЫХОДНЫЕ' $Y/1$

'ФУНКЦИЯ'

$0-(1)/(1)0;$	$11(1)/(1)0;$	$10(1)/(2)0;$
$11(2)/(3)-;$	$10(3)/(3)1;$	$00(3)/(4)1;$
$01(4)/(1)-;$	$11(3)/(3)1;$	$00(2)/(1)0;$
$01(3)/(3)1;$	$10(4)/(3)1 **$	

ТАСТАТ

'ВХОДНЫЕ' $X/2$

'ВЫХОДНЫЕ' $Y/1$

'ФУНКЦИЯ'

$0-(1)/(1)0;$	$11(1)/(1)0;$	$10(1,2)/(2)0;$
$11(2,3)/(3)-;$	$10(3)/(3)1;$	$00(3,4)/(4)1;$
$01(1,4)/(1)-;$	$11(3)/(3)1;$	$00(1,2)/(1)0;$
$01(3)/(3)1;$	$10(3,4)/(3)1 **$	

ТАДИН

'ВХОДНЫЕ' $X/2$

'ВЫХОДНЫЕ' $Y/1$

'ФУНКЦИЯ'

0-(1)/(1)0;	-1(1)/(1)0;	1-(1)/(1,2,3,4);
10(1,2,3,4)/(2)0;	1-(2)/(2,3)-;	11(2,3)/(3)-;
1-(3)/(3)1;	-0(3)/(3,4)1;	00(3,4)/(4)1;
0-(4)/(1,4)-;	01(1,4)/(1)-;	-0(1)/(1,2,3,4)0;
10(1,2,3,4)/(2)0;	-0(2)/(1,2,3,4)0;	00(1,2,3,4)/(1)0;
0-(3)/(1,3,4)1;	01(1,3,4)/(3)1;	-0(4)/(3,4)1;
10(3,4)/(3)1 **		

Автомат ТАДИН неквазимонотонен. Таблица кодирования автомата ТАСТАТ и СЛОГ-описание реализующей его схемы следующие.

ТАБЛИЦА КОДИРОВАНИЯ

1: 0 0

2: 0 1

3: 1 1

4: 1 0

ТАСТАТ

'ВХОДНЫЕ' X/2

'ВЫХОДНЫЕ' Y/1

'ВНУТРЕННИЕ' Z/2

'БЛОКИ' НЕИ(4): А1, А7;

НЕИ(2): А2, А3, А4, А8, А9;

НЕИ(1): А5; НЕИ(3): А6

'СВЯЗИ' У1=А1.У, Z1=А1.У, Z2=А7.У,

X1=А2.2=А6.1=А8.1, X2=А5.1=А6.2=А9.2,

А1.У=А2.1=А3.1=А4.1=А9.1, А2.У=А1.1=А7.1,

А3.У=А1.2, А4.У=А1.3, А5.У=А4.2=А8.2,

А6.У=А1.4=А7.2, А7.У=А3.2=А6.3,

А8.У=А7.3, А9.У=А7.4 **

ЛИТЕРАТУРА

1. Агибалов .П., Бузанов В.А., Липский В.Б., Румянцев Б.Ф.
Логическое проектирование переключательных автоматов. -Томск:
Изд-во Том. ун-та, 1983. -154 с.

Н. Г. Юрит

СИНТЕЗ МИКРОПРОГРАММНЫХ АВТОМАТОВ НА БАЗЕ МИКРОПРОЦЕССОРОВ

Пусть некоторое дискретное управляющее устройство описано на языке МИКРОПРОГРАММА [1] и описание отлажено, например, с помощью пакета программ [2]. Требуется реализовать (синтезировать) устройство на базе произвольного микропроцессорного (ММП) набора [3]. Процесс реализации устройств на ММП базе существенно отличается от традиционных методов синтеза аппаратуры, например, на элементах "Логика-2", так как появилась возможность реализовать часть устройства (иногда и полностью) программно, в то время как традиционные методы направлены на разработку схемы.

Микропроцессорные наборы (ММПН), выпускаемые и используемые современной промышленностью, различаются по структуре и своим функциональным возможностям. К структурным характеристикам относятся разрядность данных, обрабатываемых ММПН, объем оперативного запоминающего устройства (ОЗУ), наличие и количество регистров в арифметико — логическом устройстве, наличие и объем стека (память магазинного типа) и т.д. К функциональным возможностям относится система команд ММПН. Понятно, что система команд позволяет более или менее удобно программировать те или иные алгоритмы. В настоящее время известны ММПН 4, 8, 12, 16 и 32 — разрядные [3]. Так как 4 — разрядные ММПН допускают возможность объединения их в 8 — и 16 — разрядные конфигурации и система команд ММПН любой разрядности (кроме 12 — разрядного ММПН) содержит команды, оперирующие с байтами, то результат, сформулированный для 8 — разрядного ММПН будет справедлив для ММПН произвольной разрядности.

Для реализации произвольного микропрограммного автомата (ММПА), удовлетворяющего ограничениям, приведенным в [2], на базе произвольного микропроцессорного набора (ММПН) предлагается использовать двухэтапный процесс. На первом этапе составляется программа, реализующая работу ММПА на мнемокоде некоторого гипотетического микропроцессорного набора (ГМПН). На втором этапе вместо команд ГМПН подставляются команды на языке реального ММПН. Первый этап реализуется "вручную" и только один раз. Второй —

автоматически, для чего достаточно проектировщику МКПА описать каждую команду ГМН эквивалентной командой или командами конкретного МКПН. Алгоритм функционирования МКПА для представления в памяти МКПН подготавливается автоматически, с помощью соответствующей программы. Подготовка всей информации, необходимой для записи в память МКПН, подготавливается на "большой" вычислительной машине с помощью программ, представленных в языке ЛЯПАС-М [4].

Гипотетический микропроцессорный набор

Гипотетический микропроцессорный набор (ГМН) подобен рассматриваемому в [3]. Приведем здесь лишь характеристики, существенные в данной работе. Объем оперативного запоминающего устройства (ОЗУ) ГМН не менее чем 2^{16} (64к, $k=1024$) восьмибитовых слов (байт). Имеются три восьмибитовых регистра общего назначения (РОН), обозначаемые буквами А, В, С. Причем регистр А является аккумулятором, то есть регистром, в котором находится один из аргументов, участвующих в операции, и куда посылается результат операции, если особо не оговорено что-нибудь другое. Шестнадцатиразрядный регистр адреса, обозначаемый Н, позволяющий адресовать непосредственно любое из слов ОЗУ, восьмизнарядный регистр команд. Двухразрядный регистр признаков результата выполнения операции (флажков). Один признак Z отмечает равенство (неравенство) результата операции нулю, второй - Р - перенос из старшего разряда (или заём из предстаршего разряда). Стек, состоящий из пяти шестнадцатиразрядных слов.

Система команд ГМН выбиралась с учетом двух требований:

1. Обеспечить удобство программирования алгоритма интерпретации микропрограммного автомата;

2. Близость к командам реальных микропроцессорных наборов (РМН), чтобы облегчить составление заготовок из команд РМН для замены команд ГМН.

Перечислим команды ГМН, используя следующие обозначения:

(X) - содержимое X;

В - регистр общего назначения В или С;

Z - один из регистров общего назначения А, В или С;

А - регистр общего назначения с особым свойством - аккумулятор;

М - разряд аккумулятора с номером Z. Разряды нумеруются от младших к старшим (справа - налево), $Z=0, 1, \dots, 7$;

Р - признак результата (перенос или заём);

Z - признак нулевого результата;

H - адресный регистр (длина - 16 разрядов);

M - слово ОЗУ, адрес которого находится в регистре H;

$\langle B_2 \rangle$ - второй байт команды;

$\langle B_3 \rangle$ - третий байт команды;

PC - программный счетчик;

$St_i - 2$ - е слово стека (слово стека 16 - разрядное).

Набор команд гипотетического микропроцессорного набора

Регистровые пересылки

1. *MOV* z, z_2 содержимое PОН z_2 передается в PОН z , $/(z_2) \rightarrow (z)$
2. *MVM* z содержимое байта ОЗУ, адрес которого содержится в адресном регистре H, передается в PОН $z/(H) \rightarrow (z)$
3. *LRI* $z I$ загрузка регистра непосредственная содержимым второго слова команды $/I \rightarrow (z)$
4. *LDR* $z B_2 B_3$ содержимое байта ОЗУ с адресом $B_2 B_3$ помещается в PОН $z/(B_2 B_3) \rightarrow (z)$
5. *LHI* $B_2 B_3$ загрузка адресного регистра непосредственная содержимым второго и третьего слов команды $/\langle B_2 \rangle \langle B_3 \rangle \rightarrow (H)$
6. *LH* загрузка адресного регистра содержимым двух рядом расположенных байтов ОЗУ, адрес левого байта рассматриваемой пары задан в адресном регистре $/(H)((H)+1) \rightarrow (H)$
7. *LDA* $B_2 B_3$ загрузка A (аккумулятора) содержимым байта ОЗУ с адресом $B_2 B_3/(B_2 B_3) \rightarrow (A)$
8. *LAI* I в аккумулятор помещается содержимое второго байта команды $/I \rightarrow (A)$
9. *LAN* содержимое байта ОЗУ по адресу, указанному в адресном регистре H, помещается в A $/(H) \rightarrow (A)$

Команды передачи управления

10. *JAZ* $B_2 B_3$ при нулевом содержимом аккумулятора управление передается на байт с адресом, указанным во втором и третьем байтах команды
11. *JAN* $B_2 B_3$ при ненулевом содержимом аккумулятора передача управления, как в команде # 10

12. *JAP B₂B₃* передача управления при положительном содержимом аккумулятора
13. *JAM B₂B₃* передача управления при отрицательном содержимом аккумулятора
14. *JCZ R B₂B₃* передача управления по адресу $\langle B_2 \rangle \langle B_3 \rangle$ при нулевом содержимом РОН R
15. *JCN R B₂B₃* при (R) $\neq 0$ передача управления по адресу $\langle B_2 \rangle \langle B_3 \rangle$
16. *JMP B₂B₃* безусловная передача управления по адресу $\langle B_2 \rangle \langle B_3 \rangle$
17. *JHL* безусловная передача управления по адресу, содержащемуся в адресном регистре H
18. *JMS B₂B₃* переход на подпрограмму, которая начинается с адреса $\langle B_2 \rangle \langle B_3 \rangle$, при этом
 $(St_i) \rightarrow (St_{i+1}) \quad i=1, 2, \dots, 5$
 $(PC) \rightarrow (St_1)$
 $\langle B_2 \rangle \langle B_3 \rangle \rightarrow (PC)$, PC - программный счетчик
 (St_5) — пропадает
19. *RET* возврат из подпрограммы, при этом
 $(St_1) \rightarrow (PC)$
 $(St_i) \rightarrow (St_{i-1})$, $i=2, 3, \dots, 5$
 $(St_5) \rightarrow (St_5)$

Арифметические и логические операции,
команды записи в память и сдвига

20. *ADD R* $(A) + (R) \rightarrow (A)$, при этом возникающий при сложении перенос P передается в регистр признака результата
21. *ADC R* $(A) + (R) + (P) \rightarrow (A)$, возникающий перенос P передается в регистр признака результата
22. *SUB R* $(A) - (R) \rightarrow (A)$, возникающий заем P из предстаршего разряда передается в регистр признака результата
23. *SBC R* $(A) - (R) - (P) \rightarrow (A)$, возникающий заем P из предстаршего разряда передается в регистр признака результата
24. *INR R* $(R) + 1 \rightarrow (R)$ инкремент (единичное приращение) регистра данных, возникающий перенос P передается в регистр признака результата

25. *DCR* $(R) - 1 \rightarrow (R)$ декремент (отрицательное единичное приращение) регистра данных, возникающий заём/из предстаршего разряда P передается в регистр признака результата
26. *INH* $(H) + 1 = (H)$ инкремент содержимого регистра адреса, (P) - не изменяется
27. *DSH* $(H) - 1 = (H)$ декремент содержимого адресного регистра, (P) - не изменяется
28. *AND R* $(A) \wedge (R) \rightarrow (A)$ логическое умножение содержимых аккумулятора и одного из РОН В или С
29. *ANI I* $(A) \wedge I \rightarrow (A)$, I - второе слово (байт) команды, логическое умножение содержимого аккумулятора с непосредственным операндом
30. *ANH* $(A) \wedge ((H)) \rightarrow (A)$ логическое умножение содержимого аккумулятора с байтом OSV , адрес которого находится в адресном регистре
31. *OR R* $(A) \vee (R) \rightarrow (A)$ логическое сложение содержимого аккумулятора с содержимым одного из РОН В или С
32. *OI I* $(A) \vee I \rightarrow (A)$ логическое сложение содержимого аккумулятора с непосредственным операндом, заданным вторым байтом команды
33. *OH* $(A) \vee ((H)) \rightarrow (A)$ логическое сложение содержимого аккумулятора с байтом OSV , адрес которого задан в адресном регистре
34. *XOR R* $(A) \oplus (R) \rightarrow (A)$ сложение по $mod\ 2$ (дизъюнкция с исключением) содержимого аккумулятора с содержимым РОН В или С
35. *XI I* $(A) \oplus I$ сравнение содержимого аккумулятора с константой, заданной вторым байтом команды; в результате выполнения команды вырабатывается признак $Z = \begin{cases} 0, & (A) \neq I \\ 1, & (A) = I \end{cases}$
36. *XOH* $(A) \oplus ((H)) \rightarrow (A)$ сложение по $mod\ 2$ содержимого аккумулятора с байтом OSV , адрес которого задан в адресном регистре.
37. *SMA* $\neg(A) \rightarrow A$ инверсия содержимого аккумулятора как логического вектора

Команды с № 28-37 не изменяют содержимого разряда P регистра признаков результата, но вырабатывают признак Z , $Z = 0$, если

- (A) $\neq 0$ и $\chi = 1$, если (A) = 0, где (A) - содержимое аккумулятора, полученное в результате выполнения операции.
38. *СМС* $1P = P$ инверсия одного разряда регистра признака результата
39. *STC* $1 = P$ установка единичного значения разряда переноса в регистре признака результата
40. *STR* $r B_2 B_3$ $(r) \rightarrow (B_2, B_3)$ содержимое одного из общих регистров (A, B, C) записывается в ОЗУ по адресу, указанному вторым и третьим байтами команды
41. *STA* $(A) \rightarrow (H)$, содержимое аккумулятора помещается в ОЗУ по адресу, указанному в адресном регистре
42. *STH* $B_2 B_3$ $(H) \rightarrow (B_2 B_3) ((B_2 B_3) + 1)$ содержимое адресного регистра запоминается в двух последовательных байтах ОЗУ с адресами $B_2 B_3$ и $B_2 B_3 + 1$
43. *RTL* $(P) \rightarrow (A_0), (A_i) \rightarrow (A_{i+1}), i = 0, \dots, 6, (A_7) = (P)$ сдвиг содержимого аккумулятора и признака переноса влево на один разряд
44. *RTR* $(P) \rightarrow (A_7), (A_i) \rightarrow (A_{i-1}), i = 1, \dots, 7, (A_0) \rightarrow P$ сдвиг содержимого аккумулятора и признака переноса вправо на один разряд

Остальные команды

45. *INP* B_2 один байт данных с устройства ввода с номером B_2 (второе слово команды) передается в аккумулятор
46. *OUT* B_2 содержимое аккумулятора передается на устройство вывода с номером B_2
47. *PIB* опустить данные в стек
 $(St_i) \rightarrow (St_{i+1}), i = 1, 2, \dots, 5$
 $0000000(P)(A) \rightarrow (St_2)$
 $(H) \rightarrow (St_1)$
48. *POP* поднять данные из стека
 $(St_1) \rightarrow (H)$
 $(St_2 L) \rightarrow (A)$
 $(St_2 H_0) \rightarrow P$
 $(St_i) \rightarrow (St_{i-2}), i = 3, \dots, 5$
 $(St_j) \rightarrow (St_j), j = 4, 5$
 Первое слово стека передается в адресный регистр, младший байт второго слова стека

49. *NOP*

50. *NLT*

передается в аккумулятор, младший разряд старшего байта второго слова стека передается признаку результата (перенос)
пустая операция
останов

Внутренняя форма представления микропрограммного автомата

Ограничения и форма представления микропрограммного автомата (МПА) для задач отладки (синтаксиса и семантики), описания и моделирования алгоритма работы МПА описаны в [2]. Учитывая структурную организацию памяти микропроцессорных наборов (МПН) и возможности команд преобразования данных в них, предлагается внутренняя форма представления МПА, отличная от описанной в [2]. Списки входных, выходных и установочных переменных опускаются. Отмечается лишь число входных переменных. Метки из фрагментов описания установочных слов образуют отдельный массив. Одна метка занимает одно восьмиразрядное слово (или 12-разрядное, исключительно для 12 — разрядных МПН). Каждая переменная в микрокомандах заменяется своим порядковым номером в соответствующем списке. Знаки ":", "Λ", ":", ":", ">", "<" и метки (номера) микрокоманд всюду опускаются, а знаки "(", ")", "!", ">", "<", "ж" и условия вида "1" кодируются восьмиразрядными двоичными числами, не совпадавшими с номерами переменных. Длительные сигналы нумеруются в порядке их вхождения в микропрограмму, выделяются из нее и содержащаяся в них информация разносится по двум массивам, в одном из которых (обозначим его А) записываются условия и последовательности выходных переменных, а в другом (массив В) — записываются номера состояний (микрокоманд), к которым соответствующие длительные сигналы относятся. Разделителем между номерами, относящимися к различным длительным сигналам, служит символ "0" (ноль). Метки микрокоманд опускаются и создается массив координат (начало и конец) микрокоманд и массив начал длительных сигналов (условия и последовательности выходных переменных). Кроме того, формируется таблица, в которой отмечаются начала всех перечисленных частей описания МПА.

Перекодирование программы интерпретации микропрограммного автомата в языке гипотетического микропроцессорного набора на язык ассемблера реального микропроцессорного набора

Процесс перекодировки осуществляется по следующей схеме:

1. Просматриваются подряд команды в программе интерпретации, записанной на мнемокоде гипотетического микропроцессорного набора (ГМН) и заменяются на заготовки, соответствующие заменяемым командам ГМН. Заготовка есть последовательность команд реального микропроцессорного набора (РМН), выполненная на соответствующем ассемблере с учетом правил, которые будут сформулированы ниже. Все заготовки представляются единым массивом и отделяются друг от друга символом Λ , определяемым пользователем.

2. Массив заготовок готовится закавчиком - проектировщиком. Порядок следования заготовок жестко задается порядком описания команд ГМН.

3. Предполагается, что заготовки свободны от ошибок.

4. Программа на мнемокоде ГМН представлена команда за командой без разделителей. Каждая команда имеет двухбайтное поле меток, в котором пишется метка или два "пробела".

5. Метка имеет вид $A I$, где $A \in \{M, P, K, X, T\}$, $I = 0, 1, \dots, 9$.

6. Позиции заготовок, в которые необходимо подставить данные из команды программы ГМН, отмечаются символами "+", "-", " ".

Правила оформления заготовок

1. В командах, где есть непосредственный адрес (он может быть задан меткой), "+" отмечает место (байт) старшей части адреса, "-" отмечает место (байт) младшей части адреса;

2. Если, кроме непосредственного адреса, используется регистр, то позиция (ба.), где должно быть записано имя (адрес) регистра, отмечается знаком "X";

3. В двухадресных командах (формат RR - регистр-регистр или RI - регистр - непосредственный операнд) позиция приемника результата определяется знаком "+", регистр источник или непосредственный операнд (I) отмечается (его местоположение) знаком " - ";

4. В одноадресных командах, где в качестве адреса используется регистр или непосредственный операнд, его местоположение

задаётся знаком "-".

Инструкция пользователя

1. Описать проектируемое устройство на языке МИКРОПРОГРАММА;
2. Задать информацию для семантического анализа и моделирования ММПА;
3. Проверить, совпадают ли результаты моделирования с ожидаемыми и
 - а) если всё в порядке, то перейти к п.4;
 - б) изменить описание устройства и перейти к п.2;
4. Составить заготовки команд РМН для замены каждой команды ГМН с учётом правил оформления;
5. Составить массив-последовательность заготовок в порядке, задаваемом списком команд ГМН.

Литература

1. Фрит Я.Г. К моделированию микропрограмм на ЭВМ.-Известия АН СССР. Техническая кибернетика, 1976, №4, с.123-126.
2. Фрит Я.Г. Анализ и моделирование микропрограмм.-В кн.: Алгоритмы решения задач дискретной математики. Томск: Изд-во Том.ун-та, 1979, вып.1, с. 93-99.
3. Гивоне Д., Россер В. Микропроцессоры и микрокомпьютеры. Вводный курс.-М.: Мир, 1983.- 464с.
4. Закревский А.Д., Торопов Н.Р. Система программирования ЛЯПАС-М. -Минск: Наука и техника, 1978.- 240с.

А.А.Левашников

СИНТАКСИЧЕСКИЙ КОНТРОЛЬ СХЕМ НА ЯЗЫКЕ СЛОГ

Пусть X — последовательность символов из алфавита языка СЛОГ [1]. Требуется выяснить, является ли X схемой. Если X — не схема, то указать синтаксические ошибки, обнаруженные в этой последовательности.

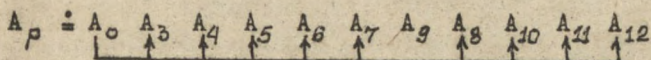
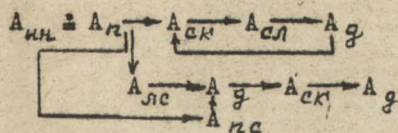
Для решения поставленной задачи необходимо по порождающей грамматике G языка СЛОГ (см. приложение) построить синтаксический анализатор A . В данном случае G — контекстно-свободная грамматика. Для таких грамматик описаны методы проектирования анализаторов [3]. Нас эти методы не устраивают по следующим причинам. Во-первых, они ориентированы на грамматический разбор и построение деревьев вывода. Во-вторых, строятся анализаторы как МП-автоматы с большими таблицами, описывающими их поведение. В-третьих, такие анализаторы трудно контролировать, т.е. трудно построить контрольную последовательность, доказывающую правильность функционирования анализатора. Согласно [2] МП-автомат можно спроектировать как сеть конечных автоматов.

Синтаксический контроль выполняется в два этапа. Сначала в X находятся лексемы: комментарии, ключевые слова, целые числа, идентификаторы и имена. Комментарии исключаются из последовательности, а целые числа, идентификаторы и имена заменяются буквами H , I и J соответственно. Разделители остаются без изменения. Это преобразование выполняет анализатор A_1 . В результате X преобразуется в X_{HJ} . На этом этапе может обнаружиться ошибка в комментарии, а именно: отсутствие пары символов $*$ $/$. Ошибка эта устраняется методом «переполоха» [4], т.е. удаляются все символы от $/$ $*$ до $\cdot$. Здесь $\cdot$ — концевой маркер последовательности; A_1 — сеть автоматов-анализаторов, включающая A_K — анализатор комментариев, A_{IH} — анализатор имен, идентификаторов и целых чисел.

На втором этапе выполняется синтаксический анализ X_{HJ} анализатором A_2 . При этом сначала распознаются и свертываются индексы, а затем — разделы.

Структуру A изобразим так:

$$\begin{aligned} A &\doteq A_1 \rightarrow A_2, \quad A_1 \doteq A_K \rightarrow A_{IH} \rightarrow A_{KJ} \\ A_2 &\doteq A_{IH} \rightarrow A_P, \end{aligned}$$



Для примера приведем таблицы переходов и выходов анализаторов A_n , $A_{ск}$, $A_{сл}$ и A_d .

A _n :	x	s	
		0	1
	ИИ	0	0
	+ - X :	0	0
	9	0	1
	(	0	0
	)	0	0
	+	0	0

X	S	
	0	1
	D	D
+ - xx: ;	+	D+, E
?	e	e, E
(	(	(
)	)	D, E
-	-	.E

A_{CK}	x_n	S							
		0	1	2	3	4	5	6	7
	(	1	1	1	3	0	0	0	0
	)	2	3	2	3	0	0	0	0
	$\Delta\{4\}$	0	1	2	3	0	0	0	0
	-	5	6	7	4	0	0	0	0

x_n	S							
	0	1	2	3	4	5	6	7
(	(	(	(	(	e	e	e	e
)	)	)	)	)	e	e	e	e
Δ	Δ	Δ	Δ	Δ	e	e	e	e
-	-	-	-	-	e	e	e	e

$A_{сг}$:

$x_{сг}$	S	
	4	0
(	0	0
)	0	4
Δ	4	0
-	8	8

$x_{сг}$	S	
	4	0
(	e	(
)	)	e
Δ	Δ	Δ
-	-	-

A_g :

$x_{сг}$	S				
	8	1	2	3	4
Д	1	1	1	1	1
+	2	2	2	2	2
(	3	3	3	3	3
)	4	4	4	4	4
-	0	0	0	0	0

$x_{сг}$	S				
	8	1	2	3	4
Д	Д	Е	e	Д + Д, Е	
+	Е	e	e	Д, Е	+
(	(	+(, Е	+	(	+(, Е
)	)	)	), Е	)	)
-	-	-	-, Е	-	-

В таблицах используются обозначения:

- $x_{сг}$ - входной символ,
- S - состояние анализатора,
- Е - ошибка,
- e - пустой символ.

Вышеприведенные анализаторы построены на основе синтаксических правил 39-42 из приложения. К этим правилам добавлены правила замены синтаксических ошибок. Покажем на примере анализатора A_g , как формируются правила замены.

Правило 39 порождает следующие цепочки Д+Д+Д, (Д+Д+Д), (Д+Д)+Д+(Д+Д). Из первой цепочки можно получить три неправильных: +Д+Д, Д++Д, Д+Д+. Вторая добавит еще две цепочки: (Д+Д), (Д+Д+), а третья - (Д+Д)Д+(Д+Д), (Д+Д)+Д(Д+Д). Неправильные цепочки типа ДД в последовательности $x_{сг}$ встречаться не могут, так как они заменились бы на Д. Проанализировав неправильные цепочки, приходим к выводу, что к правилу 39 надо добавить следующие: $++ \rightarrow ++$, $Д \rightarrow ++Д$, $Д \rightarrow Д+$, $Д+(2Д(,)+Д \rightarrow Д$. Здесь $A \rightarrow B$ означает, что А заменяет В.

В заключение отметим, что вышеописанный метод синтаксического контроля схем реализован в виде пакета программ СИН на языке ЛЯПАС-И в системе ТОМЛЕС [5].

Приложение

СПИСОК ПРАВИЛ ПОРОЖДАЮЩЕЙ ГРАММАТИКИ ЯЗЫКА СЛОГ

1. <алфавит> ::= <буква> | <цифра> | <разделитель> | <ключевое_слово>
2. <буква> ::= А | Б | В | Г | Д | Е | Ж | З | И | К | Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Э | Ю | Я | D | F | G | I | J | L | N | R | S | U | V | W | Z
3. <цифра> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
4. <разделитель> ::= + | - | x | : | ; | . | (|) | [|] | / | , | ' | 7 | < | = | * | ! | ? | # |
5. <ключевое_слово> ::= 'слово'
6. <слово> ::= <параметры> | <полосы> | <входные> | <выходные> | <блоки> | <связи> | <питание> | <земля> | <настройка> | <программируемые> | <алгоритм> | <функция> | <таблица>
7. <параметры> ::= ПА | ПАР | ПАРА | ПАРАМ | ПАРАМЕ | ПАРАМЕТ | ПАРАМЕТР | ПАРАМЕТРЫ
8. <полосы> ::= ПО | ПОЛ | ПОЛУ | ПОЛЮС | ПОЛЮСЫ
9. <входные> ::= ВХ | ВХО | ВХОД | ВХОДН | ВХОДНЫ | ВХОДНЫЕ
10. <выходные> ::= ВЫ | ВЫХ | ВЫХО | ВЫХОД | ВЫХОДН | ВЫХОДНЫ | ВЫХОДНЫЕ
11. <блоки> ::= Б | БЛ | БЛО | БЛОК | БЛОКИ
12. <связи> ::= СВ | СВЯ | СВЯЗ | СВЯЗИ
13. <питание> ::= ПИ | ПИТ | ПИТА | ПИТАН | ПИТАНИ | ПИТАНИЕ
14. <земля> ::= З | ЗЕ | ЗЕМ | ЗЕМЛ | ЗЕМЛЯ
15. <настройка> ::= Н | НА | НАС | НАСТ | НАСТР | НАСТРО | НАСТРОЙ | НАСТРОЙК | НАСТРОЙКА
16. <программируемые> ::= ПР | ПРО | ПРОГ | ПРОГ | ПРОГР | ПРОГРА | ПРОГРАМ | ПРОГРАММА | ПРОГРАММИ | ПРОГРАММИР | ПРОГРАММИРУ | ПРОГРАММИРУЕ | ПРОГРАММИРУЕМ | ПРОГРАММИРУЕМЫ | ПРОГРАММИРУЕМЫЕ
17. <алгоритм> ::= А | АЛ | АЛГ | АЛГО | АЛГОР | АЛГОР | АЛГОРИТ | АЛГОРИТМ
18. <функция> ::= Ф | ФУ | ФУН | ФУНК | ФУНКЦ | ФУНКЦИ | ФУНКЦИЯ
19. <таблица> ::= Т | ТА | ТАБ | ТАБЛ | ТАБЛИ | ТАБЛИЦ | ТАБЛИЦА
20. <идентификатор> ::= <буква> | <идентификатор> <бц>
21. <бц> ::= <буква> | <цифра>
22. <имя> ::= <идентификатор> | <целое> <идентификатор>

23. $\langle \text{целое} \rangle ::= \langle \text{цифра} \rangle \mid \langle \text{целое} \rangle \langle \text{цифра} \rangle$
24. $\langle \text{комментарий} \rangle ::= \langle \text{текст} \rangle \langle \text{текст} \rangle$
25. $\langle \text{текст} \rangle ::= \langle \text{символ} \rangle \mid \langle \text{текст} \rangle \langle \text{символ} \rangle$
26. $\langle \text{символ} \rangle ::= \langle \text{бц} \rangle \mid \langle \text{разделитель} \rangle \mid \langle \text{произвольный_символ} \rangle$
27. $\langle \text{параметр} \rangle ::= \langle \text{имя} \rangle$
28. $\langle \text{схема} \rangle ::= \langle \text{с_схема} \rangle \mid \langle \text{ф_схема} \rangle$
29. $\langle \text{с_схема} \rangle ::= \langle \text{описание_с_схемы} \rangle \langle \text{с_схема} \rangle$
30. $\langle \text{ф_схема} \rangle ::= \langle \text{описание_ф_схемы} \rangle \langle \text{ф_схема} \rangle$
31. $\langle \text{описание_с_схемы} \rangle ::= \langle \text{заголовок} \rangle \langle \text{тело_с_схемы} \rangle$
32. $\langle \text{описание_ф_схемы} \rangle ::= \langle \text{заголовок} \rangle \langle \text{тело_ф_схемы} \rangle$
33. $\langle \text{тело_с_схемы} \rangle ::= \langle \text{описание_с_полюсов} \rangle \mid$
 $\langle \text{описание_параметров} \rangle \langle \text{описание_с_полюсов} \rangle$
 $\mid \langle \text{описание_с_полюсов} \rangle \langle \text{описание_блоков} \rangle$
 $\langle \text{описание_связей} \rangle \mid \langle \text{описание_параметров} \rangle$
 $\langle \text{описание_с_полюсов} \rangle \langle \text{описание_блоков} \rangle$
 $\langle \text{описание_связей} \rangle$
34. $\langle \text{тело_ф_схемы} \rangle ::= \langle \text{описание_ф_полюсов} \rangle \mid$
 $\langle \text{описание_параметров} \rangle \langle \text{описание_ф_полюсов} \rangle$
 $\mid \langle \text{описание_ф_полюсов} \rangle \langle \text{алгоритм_поведения} \rangle$
 $\mid \langle \text{описание_параметров} \rangle \langle \text{описание_ф_полюсов} \rangle$
 $\langle \text{алгоритм_поведения} \rangle$
35. $\langle \text{имя_полюса} \rangle ::= \langle \text{v1} \rangle \mid \langle \text{v1} \rangle$
36. $\langle \text{v1} \rangle ::= \langle \text{имя} \rangle \mid \langle \text{целое} \rangle \mid \langle \text{целое} \rangle \mid \langle \text{имя} \rangle \langle \text{целое} \rangle$
 $\mid \langle \text{целое} \rangle \langle \text{целое} \rangle$
37. $\langle \text{имя_группы_полюсов} \rangle ::= \langle \text{ин_имя} \rangle \mid$
 $\langle \text{индексированное_имя} \rangle \mid \langle \text{имя_с_границами} \rangle$
38. $\langle \text{ин_имя} \rangle ::= \langle \text{индекс} \rangle \mid \langle \text{имя_полюса} \rangle \mid \langle \text{индекс} \rangle \mid$
 $\langle \text{ин_имя} \rangle \mid \langle \text{индекс} \rangle \mid \langle \text{ин_имя} \rangle \langle \text{v1} \rangle$
39. $\langle \text{индекс} \rangle ::= \langle \text{пц} \rangle \mid \langle 0 \rangle \langle \text{индекс} \rangle \mid (\langle \text{индекс} \rangle)$
 $\mid \langle \text{индекс} \rangle \langle 01 \rangle \langle \text{индекс} \rangle$
40. $\langle \text{пц} \rangle ::= \langle \text{параметр} \rangle \mid \langle \text{целое} \rangle$
41. $\langle 0 \rangle ::= ? \mid \#$
42. $\langle 01 \rangle ::= + \mid - \mid x \mid : \mid ;$
43. $\langle \text{индексированное_имя} \rangle ::= (\langle \text{индексное_выражение} \rangle) \mid$
 $\langle \text{имя_полюса} \rangle (\langle \text{индексное_выражение} \rangle) \mid$
 $\langle \text{ин_имя} \rangle (\langle \text{индексное_выражение} \rangle)$
44. $\langle \text{Индексное_выражение} \rangle ::= \langle \text{индекс} \rangle \mid$
 $\langle \text{индексное_выражение} \rangle, \langle \text{индекс} \rangle$
45. $\langle \text{имя_с_границами} \rangle ::= \langle \text{v5} \rangle \mid \langle \text{имя_полюса} \rangle \langle \text{v5} \rangle \mid$
 $\langle \text{ин_имя} \rangle \langle \text{v5} \rangle \mid \langle \text{индексированное_имя} \rangle \langle \text{v5} \rangle$

46. $\langle \text{в5} \rangle ::= / \langle \text{границы} \rangle \mid . / \langle \text{границы} \rangle$
47. $\langle \text{границы} \rangle ::= \langle \text{индекс} \rangle \mid \langle \text{индекс} \rangle * \langle \text{индекс} \rangle$
48. $\langle \text{имя_блока} \rangle ::= \langle \text{имя_полюса} \rangle$
49. $\langle \text{имя_группы_блоков} \rangle ::= \langle \text{имя_группы_полюсов} \rangle$
50. $\langle \text{имя_полюса_блока} \rangle ::= \langle \text{имя_блока} \rangle . \langle \text{имя_полюса} \rangle$
51. $\langle \text{имя_группы_полюсов} \rangle ::= \langle \text{имя_группы_блока} \rangle .$
 $\quad \langle \text{имя_группы_полюсов} \rangle / \langle \text{имя_группы_блоков} \rangle .$
 $\quad \langle \text{имя_полюса} \rangle \mid \langle \text{имя_группы_блоков} \rangle .$
 $\quad \langle \text{имя_группы_полюсов} \rangle$
52. $\langle \text{описание_с_полюсов} \rangle ::=$
 $\quad \langle \text{описание_входных_с_полюсов} \rangle \mid$
 $\quad \langle \text{описание_выходных_с_полюсов} \rangle \mid$
 $\quad \langle \text{описание_двунаправленных_с_полюсов} \rangle \mid$
 $\quad \langle \text{описание_с_полюсов} \rangle \langle \text{описание_входных_с_полюсов} \rangle$
 $\quad \mid \langle \text{описание_с_полюсов} \rangle \langle \text{описание_выходных}$
 $\quad \quad \text{с_полюсов} \rangle$
 $\quad \mid \langle \text{описание_с_полюсов} \rangle$
 $\quad \quad \langle \text{описание_двунаправленных_с_полюсов} \rangle$
53. $\langle \text{описание_выходных_с_полюсов} \rangle ::= \langle \text{'входные'} \rangle$
 $\quad \langle \text{список_с_полюсов} \rangle$
54. $\langle \text{описание_выходных_с_полюсов} \rangle ::= \langle \text{'выходные'} \rangle$
 $\quad \langle \text{список_с_полюсов} \rangle$
55. $\langle \text{описание_двунаправленных_с_полюсов} \rangle ::= \langle \text{'полюсы'} \rangle$
 $\quad \langle \text{список_с_полюсов} \rangle$
56. $\langle \text{список_с_полюсов} \rangle ::= \langle \text{в6} \rangle \mid \langle \text{список_с_полюсов} \rangle , \langle \text{в6} \rangle$
57. $\langle \text{в6} \rangle ::= \langle \text{имя_полюса} \rangle \mid \langle \text{имя_группы_полюсов} \rangle$
 $\quad \mid \langle \text{имя_полюса} \rangle [\langle \text{описание_границ} \rangle]$
 $\quad \mid \langle \text{имя_группы_полюсов} \rangle [\langle \text{описание_границ} \rangle]$
58. $\langle \text{описание_границ} \rangle ::= \langle \text{интервал} \rangle \mid$
 $\quad \langle \text{описание_границ} \rangle , \langle \text{интервал} \rangle$
59. $\langle \text{интервал} \rangle ::= \langle \text{точка} \rangle \mid \langle \text{отрезок} \rangle$
60. $\langle \text{точка} \rangle ::= \langle \text{параметр} \rangle = \langle \text{индекс} \rangle \mid \langle \text{индекс} \rangle = \langle \text{параметр} \rangle$
61. $\langle \text{отрезок} \rangle ::= \langle \text{параметр} \rangle \langle \text{02} \rangle \langle \text{индекс} \rangle \mid \langle \text{индекс} \rangle \langle \text{02} \rangle$
 $\quad \langle \text{параметр} \rangle \langle \text{02} \rangle \langle \text{индекс} \rangle$
62. $\langle \text{02} \rangle ::= \langle \text{'|'} \rangle \mid \langle \text{'='} \rangle$
63. $\langle \text{описание_ф_полюсов} \rangle ::= \langle \text{описание_с_полюсов} \rangle \mid$
 $\quad \langle \text{описание_переменных} \rangle$
64. $\langle \text{описание_переменных} \rangle ::=$
 $\quad \langle \text{описание_входных_переменных} \rangle \mid$
 $\quad \langle \text{описание_выходных_переменных} \rangle$

65. $\langle \text{описание_входных_переменных} \rangle :: = \langle \text{входные} \rangle$
 $\langle \text{список_переменных} \rangle$
66. $\langle \text{описание_выходных_переменных} \rangle :: = \langle \text{выходные} \rangle$
 $\langle \text{список_переменных} \rangle$
67. $\langle \text{список_переменных} \rangle :: = \langle \text{в7} \rangle | \langle \text{список_переменных} \rangle, \langle \text{в7} \rangle$
68. $\langle \text{в7} \rangle :: = \langle \text{имя} \rangle = \langle \text{в6} \rangle | \langle \text{имя} \rangle = (\langle \text{список_с_полюсов} \rangle)$
69. $\langle \text{описание_блоков} \rangle :: = \langle \text{блоки} \rangle$
 $\langle \text{список_фрагментов_блоков} \rangle$
70. $\langle \text{список_фрагментов_блоков} \rangle :: =$
 $\langle \text{фрагмент_описания_блоков} \rangle$
 $| \langle \text{список_фрагментов_блоков} \rangle ;$
 $\langle \text{фрагмент_описания_блоков} \rangle$
71. $\langle \text{фрагмент_описания_блоков} \rangle :: = \langle \text{тип_блока} \rangle :$
 $\langle \text{список_имен_блоков} \rangle$
72. $\langle \text{тип_блока} \rangle :: = \langle \text{идентификатор} \rangle |$
 $\langle \text{идентификатор} \rangle (\langle \text{индексное_выражение} \rangle)$
 $| \langle \text{идентификатор} \rangle \langle \text{описание_настройки} \rangle$
 $| \langle \text{идентификатор} \rangle (\langle \text{индексное_выражение} \rangle)$
 $\langle \text{описание_настройки} \rangle$
73. $\langle \text{список_имен_блоков} \rangle :: = \langle \text{список_с_полюсов} \rangle$
74. $\langle \text{описание_настройки} \rangle :: = \langle \text{настройка} \rangle$
 $\langle \text{список_уравнений_связи} \rangle$
75. $\langle \text{заголовок} \rangle :: = \langle \text{идентификатор} \rangle | \langle \text{идентификатор} \rangle$
 $(\langle \text{список_параметров} \rangle)$
76. $\langle \text{список_параметров} \rangle :: = \langle \text{параметр} \rangle |$
 $\langle \text{список_параметров} \rangle, \langle \text{параметр} \rangle$
77. $\langle \text{описание_связей} \rangle :: = \langle \text{описание_фиксированных_связей} \rangle$
 $| \langle \text{описание_программируемых_связей} \rangle$
 $| \langle \text{описание_фиксированных_связей} \rangle$
 $\langle \text{описание_программируемых} (\text{x связей}) \rangle$
78. $\langle \text{описание_фиксируемых_связей} \rangle :: = \langle \text{связи} \rangle$
 $\langle \text{список_уравнений_связи} \rangle$
79. $\langle \text{описание_программируемых_связей} \rangle :: = \langle \text{программируемые} \rangle$
 $\langle \text{список_уравнений_связи} \rangle$
80. $\langle \text{список_уравнений_связи} \rangle :: = \langle \text{уравнение_связи} \rangle |$
 $\langle \text{список_уравнений_связи} \rangle, \langle \text{уравнение_связи} \rangle$
81. $\langle \text{уравнение_связи} \rangle :: = \langle \text{терминал} \rangle = \langle \text{терминал} \rangle |$
 $\langle \text{уравнение_связи} \rangle = \langle \text{терминал} \rangle$
82. $\langle \text{терминал} \rangle :: = \langle \text{питание} \rangle | \langle \text{земля} \rangle | \langle \text{имя_полюса} \rangle$

- | < имя_группы_полюсов > | < имя_полюса_блока >
| < имя_группы_полюсов_блока >
83. < описание_параметров > :: = < параметры >
 < программа_вычислений >
84. < программа_вычислений > :: = < фрагмент > |
 < программа_вычислений > , < фрагмент >
85. < фрагмент > :: = < а_строка > | * < макро >
86. < а_строка > :: = < а-выражение > | < а_строка > < 0 >
 < а-выражение > | < а_строка > < 01 > < а_выражение >
87. < а-выражение > :: = < индекс > = < параметр > |
 < а-выражение > = < параметр >
88. < макро > :: = < макрооперация > | < макро > < макроопераций >
89. < макрооперация > :: = * < идентификатор >
 (< входные_операнды > / < список_параметров >)
90. < входные_операнды > :: = < целое > | < параметр >
 | < входные_операнды > , < целое >
 | < входные_операнды > ,
 < параметр >

Литература

1. Агibalов Г.П., Буванов В.А., Липский В.Б., Румянцев Б.Ф. Логическое проектирование переключательных автоматов. Томск: Изд-во Томского университета, 1983. - 155 с.
2. Олленгрен А. Определение языков программирования интерпретирующими автоматами. - М.: Мир, 1977. - 288 с.
3. Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции. т.1, М.: Мир, 1978 - 613 с., т.2, М.: Мир: 1978 - 488 с.
4. Хантер Р. Проектирование и конструирование компиляторов. - М.: Финансы и статистика, 1984. - 232 с.
5. Быкова С.В., Головчинер М.Н., Жуковский О.И., Николаева Л.А., Питосин В.Т. ТОМЛЕС - система программирования с базовым языком Ляпас-М для ЕС ЭВМ. - Программирование, 1984, № 5, с. 56-63.

И.Е.Веревкина

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ СХЕМ ИЗ ФУНКЦИОНАЛЬНЫХ ЭЛЕМЕНТОВ

1. Основные понятия

Под функциональным элементом здесь понимается (M, P) – полюсник, реализующий P монотонных функций состояний от M переменных. Функция состояний – это любая функция, которая вместе со своими переменными принимает значения из 7-значного алфавита $\underline{C} = \{0, I, X, X', I', O', A\}$, представляющего собой множество всевозможных непустых подмножеств трехэлементного множества $\underline{E} = \{0, I, X\}$. В нем $O = \{0\}$, $I = \{I\}$, $X = \{X\}$, $X' = \{0, I\}$, $I' = \{0, X\}$, $O' = \{I, X\}$, $A = \{0, I, X\}$. Одну из возможных интерпретаций символов алфавита $\underline{C}$ можно найти в [1], где O, I и X трактуются как состояния логических $0, I$ и высокого импеданса соответственно. Функция f называется монотонной, если для любых двух наборов α и β значений ее переменных из $\alpha \leq \beta$ следует $f(\alpha) \leq f(\beta)$.

Схемы из функциональных элементов строятся как схемы с иерархической структурой, т.е. как схемы из блоков, каждый из которых может быть функциональным элементом или опять же схемой из блоков. Блоки в схеме соединяются путем отождествления их полюсов с образованием узлов схемы, причем в один узел может быть объединено несколько выходных полюсов блоков, тогда этот узел считается функциональным элементом, реализующим функцию проводного ИЛИ. Полным состоянием схемы называется состояние всех ее узлов, входным состоянием – состояние ее входных узлов, выходным – состояние выходных узлов.

Имитационное моделирование схемы заключается в определении состояний выходов блоков схемы через их функции или схемы в зависимости от состояний входов этих блоков до тех пор, пока выходы не перестанут изменяться, т.е. пока не будет достигнуто устойчивое состояние схемы. Чтобы это происходило всегда, предполагается, что начальное состояние схемы устойчиво, а моделирование ведется по методу Эйхельбергера [2], т.е. через промежуточное входное состояние, равное объединению прежнего и текущего входных состояний. Особо выделяются синхронизирующие входы схемы. Для них промежуточное входное состояние совпадает с текущим.

2. Комплекс программ имитационного моделирования

Программный комплекс, составленный на языке ЛЯПАС-М, содержит около 30 макроопераций и использует рекурсивное обращение. Программа-имитатор позволяет моделировать схемы, стандартно представленные и хранящиеся в памяти ЭВМ. Для каждого функционального элемента существует и хранится в памяти ЭВМ программа, вычисляющая его функцию.

Внешне схемы описываются в канонической форме языка СЛОГ [1]. Для моделирования схемы надо последовательно выполнить две макрооперации: ПЕРЕВОД и АНАЛ2. Макрооперация ПЕРЕВОД () осуществляет автоматическое преобразование внешней формы описания в стандартную и помещает ее в архив. Все необходимые данные макрооперация вводит с перфокарт в следующем порядке:

- 1) описание схемы в канонической форме языка СЛОГ;
- 2) список синхровходов схемы через запятую, возможно пустой;
- 3) номер архива (S), в котором хранятся стандартные представления схем;
- 4) номер архива (Q), сохраняющего списки имен внешних полюсов схем.

Результатом работы макрооперации является пополнение архивов и выдача на печать стандартного представления схемы или диагностическое сообщение об ошибке.

К моменту обращения к макрооперации ПЕРЕВОД в архиве S должны находиться стандартные представления всех схем, которые встречаются в данной схеме в качестве блоков на любом уровне иерархии, и всех функциональных элементов, использованных в схеме, а в архиве Q - все соответствующие таблицы имен внешних полюсов.

Моделирование схемы осуществляется макрооперацией АНАЛ2 (), которая вводит следующие массивы перфокарт:

- 1 - имя схемы, представленное в символьном виде (на одной карте);
- 2 - номер архива, в котором хранятся стандартные представления схем;
- 3 - номер архива, в котором хранятся М-сегменты, вычисляющие функции элементов;
- 4 - номер рабочего архива;
- 5 - исходное полное устойчивое состояние схемы;
- 6, 7... - последовательность входных состояний схемы.

Массивы отделяются друг от друга картами конца файла (кроме 2, 3, 4). Последний конец файла дублируется, так как признаком завершения работы макрооперации является пустой набор значений входных состояний. Подробно способ задания информации описан в п.4.

Результатом работы макрооперации является печать таблиц моделирования схемы. Столбцы таблиц соответствуют состояниям схемы: *С* – предыдущее полное устойчивое состояние; *С + В* – промежуточное полное устойчивое; *В* – последующее полное устойчивое состояние схемы. Стрски таблиц соответствуют узлам схемы.

3. Алгоритм моделирования

Суть алгоритма моделирования, реализованного в макрооперации, состоит в следующем. Из последовательности входных состояний выбирается очередное входное состояние и вычисляется соответствующее промежуточное с учетом особенностей синхровходов. Выделяются блоки схемы, значения входных полюсов которых при подаче на входы схемы промежуточного состояния стали отличаться от их значений в исходном состоянии. Последовательно рассматривается каждый такой блок. Если он – функциональный элемент, то его функция вычисляется по хранящейся в памяти ЭВМ программе, а если нет, то структура блока раскрывается, т.е. начинается моделирование его схемы по тому же принципу, что и основной схемы. Полученное выходное состояние определяет, у каких еще блоков схемы изменились входные состояния; эти блоки также подлежат дальнейшему рассмотрению. Это продолжается до тех пор, пока схема не приходит в устойчивое состояние. Далее на ее входы подается выбранное входное состояние, и всё повторяется. Затем выбирается новое входное состояние из последовательности и т.д.

Ниже приводится подробная структура данного алгоритма.

1. Получение нового входного состояния.
2. Составление списка моделируемых блоков.
3. Объявление списка рабочим списком.
4. Выбор очередного блока из рабочего списка.
5. Выделение входного состояния этого блока.
6. Проверка: является ли данный блок функциональным элементом? Если нет, то переход к 1, иначе 7.

7. Вычисление функции элемента по соответствующей программе.
8. Сравнение полученного выходного состояния с соответствующим старым состоянием. Если они равны, то переход к II, иначе 9.
9. Фиксирование нового выходного состояния.
10. Формирование нового списка моделируемых блоков.
- II. Если просмотр рабочего списка не закончен, то переход к 4, иначе I2.
- I2. Если новый список моделируемых блоков не пуст, то переход к 3, иначе I3.
- I3. Если промоделирована схема, то переход к I6, если же закончено моделирование блока, то I4.
- I4. Выделение выходного состояния блока.
- I5. Переход к п.8.
- I6. Если последовательность входных состояний не исчерпана, то переход к I, иначе I7.
- I7. Конец.

4. Внешнее представление информации

Каноническая форма языка СЛОГ ориентирована на представление конкретной схемы и включает в себя разделы "СХЕМА", "ВХОДНЫЕ", "ВЫХОДНЫЕ", "ПОЛЮСЫ", "БЛОКИ" и "СВЯЗИ", указывающие соответственно имя схемы (не более 8 символов), имена ее входных, выходных и двунаправленных полюсов (не более 50 символов на имя), имена и типы блоков (тип блока - не более 8 символов, имя блока - не более 50), а также способ разбиения множества полюсов блоков на классы, каждый из которых является узлом схемы. Уравнения связи тем самым содержат только одиночные полюсы и максимальны, т.е. в каждом уравнении указываются все полюсы блоков, отождествленные в один узел. При указании имени полюса блока сначала указывается имя самого блока, затем, после символа точки, имя полюса. Описание схемы заканчивается двумя звездочками (жж).

В описании схем, являющихся функциональными элементами, отсутствуют разделы "БЛОКИ" и "СВЯЗИ".

Исходные полные устойчивые состояния схемы и последовательности входных состояний для моделирования задаются следующим образом. Одновременно можно моделировать до тридцати двух после-

довательностей, каждую из своего исходного состояния. По умолчанию исходные состояния принимаются равными полностью неопределенным состояниям. При другом наборе исходных состояний он задается в виде матрицы, элементы которой принимают значения из алфавита $\mathcal{C}$. Строки матрицы соответствуют узлам схемы, столбцы - моделируемым последовательностям. Наборы входных состояний задаются аналогичным образом: там строки соответствуют входным узлам схемы.

5. П р и м е р

Приводится пример моделирования схемы JK-триггера [1] на четырех входных последовательностях длины 3 из полностью неопределенных состояний.

Для макрооперации ПЕРЕВОД задаются:

1) описание схемы в канонической форме языка СЛОГ:

"СХЕМА" JK ТРИГГЕР

"ВХОДНЫЕ" 1, 2, 3, 4, 5, 6

"ВЫХОДНЫЕ" 7, 8

"БЛОКИ" ИНЕКМ2:1,2,3,4; ИЛИНЕКМ2:5;

ОСИНЕКМ:6; КЛЮЧ:7; КЛЮЧ2:8

"СВЯЗИ" 1 = 5.2, 2 = 6.1, 3 = 1.2 = 4.1, 4 = 2.2 = 3.1,

5 = 8.1 = 7.1, 6 = 8.2 = 7.2, 6.4 = 7.3,

3.3=7.4, 1.3=3.2 = 8.3, 4.3=8.4,

7 = 2.3=4.2=5.1=6.2, 7.5 = 1.1,

8 = 8.5=2.1, 5.3 = 6.3 КХ ;

(здесь входы: 1=J, 2=K, 3= $\bar{A}$, 4= $\bar{B}$, 5=C, 6= $\bar{C}$; блоки: 1,2,3,4=A, 5=B, 6=D, 7=C, 8= $\bar{C}$; выходы - 7=Q, 8= $\bar{Q}$),

2) список синхровходов: 5,6 ;

3), 4) - номера архивов: например, 28, 29.

Для макрооперации АНАЛ2 задаются:

1) имя схемы: JKТРИГГЕР ;

2), 3), 4) - номера архивов: например, 28, 31, 30 ;

5) набор исходных состояний - в нашем примере не задается (принимается по умолчанию),

6), 7), 8) - четыре последовательности входных состояний, каждая длины 3:

	6	7	8
1	0000	0000	0000
2	1101	1101	1101
3	1011	1111	1111
4	0100	1111	1111
5	0011	0011	1100
6	1100	1100	0011

(Здесь столбец - это состояние; последовательность k -х столбцов трех данных массивов - это k -я последовательность состояний, $k = 1, 2, 3, 4$).

Результат работы макрооперации - печать таблиц моделирования:

T_1	С	С+В	В	С	С+В	В	С	С+В	В	С	С+В	В
	А	А	О	А	А	О	А	А	О	А	А	О
	А	А	І	А	А	І	А	А	О	А	А	І
	А	А	І	А	А	О	А	А	І	А	А	І
	А	А	О	А	А	І	А	А	О	А	А	О
	А	О	О	А	О	О	А	І	І	А	І	І
	А	І	І	А	І	І	А	О	О	А	О	О
	А	А	О	А	А	А	А	А	І	А	А	О
	А	А	І	А	А	О	А	А	І	А	А	І
	А	А	О	А	А	І	А	А	А	А	А	А
	А	А	О	А	А	І	А	А	О	А	А	О
(*)	А	А	І	А	А	О	А	А	І	А	А	І
	А	А	І	А	А	О	А	А	І	А	А	О
	А	А	О	А	А	І	А	А	О	А	А	О
	А	А	О	А	А	А	А	А	О	А	А	О
T_2	О	О	О	О	О	О	О	О	О	О	О	О
	І	І	І	І	І	І	О	О	О	І	І	І
	І	І	І	О	Х	І	І	І	І	І	І	І
	О	Х	І	І	І	І	О	Х	І	О	Х	І
	О	О	О	О	О	О	І	І	І	І	І	І
	І	І	І	І	І	І	О	О	С	О	О	О
	О	О	О	А	О	О	І	І	І	О	О	О

$T_2:$	C	C+B	B	C	C+B	B	C	C+B	B	C	C+B	B
	I	I	I	0	0	0	I	I	I	I	A	0
	0	0	0	I	I	I	A	0	0	A	I	I
	0	0	0	I	I	I	0	0	0	0	0	0
(*)	I	I	I	0	0	0	I	I	I	I	I	I
	I	I	I	0	0	0	I	I	I	0	0	0
	0	0	0	I	I	I	0	0	0	0	0	0
	0	0	0	A	I	I	0	0	0	0	0	0

$T_3:$	C	C+B	B	C	C+B	B	C	C+B	B	C	C+B	B
	0	0	0	0	0	0	0	0	0	0	0	0
	I	I	I	I	I	I	0	0	0	I	I	I
	I	I	I	I	I	I	I	I	I	I	I	I
	I	I	I	I	I	I	I	I	I	I	I	I
	0	I	I	0	I	I	I	0	0	I	0	0
	I	0	0	I	0	0	0	I	I	0	I	I
	0	0	0	0	0	0	I	I	I	0	0	0
	I	0	0	0	0	0	I	I	I	0	0	0
	0	I	I	I	I	I	0	0	0	I	I	I
	0	0	0	I	I	I	0	0	0	0	I	I
(*)	I	I	I	0	0	0	I	I	I	I	0	0
	I	0	0	0	0	0	I	I	I	0	0	0
	0	0	0	I	I	I	0	0	0	0	I	I
	0	0	0	I	I	I	0	0	0	0	I	I

Таблицы T_1 , T_2 и T_3 отражают реакцию схемы соответственно на первые, вторые и третьи входные состояния четырех последовательностей одновременно. Строки таблиц соответствуют узлам схемы в порядке их следования в СЛОГ-описании схемы. Так, например, строка, отмеченная (*), соответствует выходу схемы 7, т.е. Q . В ответ на первую входную последовательность состояния этого узла принимает значения I, I, I; в ответ на вторую - 0, 0, 0; на третью - I, I, I и на четвертую входную последовательность - I, I, 0.

Л и т е р а т у р а

1. Агибалов Г.П., Бузанов В.А. и др. Логическое проектирование переключательных автоматов. - Томск: Изд-во Томского ун-та, 1983. - 152 с.
2. Eichelberger E.B. Hazard detection in combinational and sequential switching circuits. - *IBM Journal of Research and Development*, 1965, v. 9, N2, p. 90-99.

Л. А. Николаева

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ МИКРОПРОЦЕССОРНЫХ СИСТЕМ

1. Введение

В настоящее время микропроцессорные системы (МПС) становятся массовым средством вычислительной техники [1]. В процессе проектирования таких систем возникает необходимость в их имитационном моделировании на ЭВМ, что объясняется следующими причинами:

- малый объем оперативной памяти МПС;
- ограниченность состава внешних устройств МПС;
- необходимость параллельной разработки МПС и программного обеспечения (ПО);
- возможность комплексной отладки ПО с имитацией внешней среды.

Для имитации МПС необходимо иметь ее описание на функциональном уровне, определяющем список команд, структуру данных, временные соотношения, управляющие сигналы. Типовая МПС включает пять функциональных блоков: устройство ввода, память, арифметическое устройство, устройство управления, устройство вывода. Данные могут быть в прямом, дополнительном и обратном коде. Списки команд конкретных МПС и алгоритмы их функционирования могут существенно различаться, так как на выбор их влияет проблемная ориентация разрабатываемых МПС.

В данной статье сообщается об автоматизированной системе разработки имитаторов МПС на базе алгоритмического языка ЛАПАС-М [2]. Система представляет пользователю следующие средства:

- 1) язык описания МПС на уровне систем команд;
- 2) расширяющийся набор программных моделей операций, выполняемых в МПС;
- 3) средства автоматической генерации имитатора конкретной МПС по ее программной модели функционирования (ПМФ).

Для разработки имитатора конкретной МПС пользователь данной системы должен описать МПС массивами, приведенными в п.2, подготовить ее ПМФ, если она не охватывается существующей в системе, и недостающие программы выполнения операций и с помощью генератора имитаторов (ГИМ) настроить ПМФ на имитацию данной МПС. Син-

тактический контроль описания приведен в п.3. ПМФ, реализованная в системе, описана в п.4. Генератор имитаторов и требования, предъявляемые к ПМФ, изложены в п.5. Все программные средства системы записаны на ЛЯПАС -М и могут эксплуатироваться в рамках операционных систем ЕС и СМ ЭВМ.

2. Язык описания микропроцессорных систем

Описание МПС в системе выполняется с помощью трех массивов: ПАМЯТЬ, ОПЕРАЦИЯ и ОПЕРАНД. Массив ПАМЯТЬ представляет собой последовательность, составленную из слов (ячеек) ОЗУ, ПЗУ, ППЗУ. Разрядность слов может быть до 32 бит. В ПАМЯТИ содержатся данные и команды. ПАМЯТЬ можно разделить на участки двух типов, один из которых допускает только чтение (ПЗУ, ППЗУ), другой - чтение и запись в ОЗУ. Описание массива ПАМЯТЬ имеет следующую структуру:

ПАМЯТЬ П1П2... П_i ... П_Р ** ,

где П_i - описание *i*-й группы массива. Полное описание *i*-й группы имеет следующий вид:

СЛОВО (С) ЧТЕНИЕ (И) ЗАПИСЬ (И) * ИМЯ (И),

где С указывает разрядность (количество бит) слова или ячейки памяти, а И - интервал или последовательность интервалов адресов слов памяти, допускающих чтение или запись и содержащих данные или программы с некоторым именем ИМЯ. Интервал может быть представлен одним числом или двумя, разделенными знаком " - ". Интервалы в последовательности разделяются запятыми. Слово СЛОВО, ЧТЕНИЕ, ЗАПИСЬ могут быть сокращены до одной буквы, перед ними и перед символом * могут стоять пробелы и в описании любое из них может быть опущено. Например, описание

ПАМЯТЬ С (8) Ч(1-100) Ч(120-300)

* Д (550-600) * П(200-300) З(550-600)

означает, что задана память, состоящая из последовательности 8-разрядных слов, при этом интервалы слов с номерами от 1 до 100 и от 120 до 300 открыты для чтения и интервал слов с номерами от 550 до 600 открыт для чтения и записи; в памяти в ячейках, имеющих номера от 550 до 600, находятся данные с именем Д, а в ячейках от 200 до 300 - программа с именем П.

Массив ОПЕРАЦИЯ - это список операций МП-системы (мнемокоды, десятичные коды) и их характеристик, указывающих, например, на возможность прерывания после выполнения операций, на необходимость записи в ячейку или чтения из ячейки памяти, на длительность

выполнения, длину команд. Описание массива ОПЕРАЦИЯ имеет следующую структуру:

ОПЕРАЦИЯ 01 02 ... 0i ... 0m **,

где 0i - описание i-й группы характеристик операций. Описание конкретной группы имеет следующий вид:

Н / П / (C_i [N_i] , ..., C_i [N_i] , ..., C_n [N_n] ,

где Н - это общее название группы операций или характеристик, П - это некоторое пояснение характеристики, например, единица измерения; C_i - i-й код операции или значение i-й характеристики; N_i - количество соседних одинаковых значений характеристик. Компоненты [N_i] или / П / могут быть опущены, если N_i = 1 или нет пояснения П, например, описание

ОПЕРАЦИЯ

МЯМОКОД / БЦ / (ADD , SUB , MOV1 , JMP)

КОД / 10 / (101 , 102 , 103 , 104)

ОПЕРАНД / 01 * p , 10 - П / (3 [2] , 1 , 0)

ПРЕРЫВАНИЕ / 0 - нет / (1 [3] , 0)

ВРЕМЯ / МКСЕК / (1 , 2 , 3 [2])

ДЛИНА / БИТ / (8 [2] , 16 , 8)

означает, что в МПС выполняются операции ADD, SUB, MOV1, JMP, записанные в буквенно-цифровом коде. Их десятичный код: 101, 102, 103, 104. По характеристике, имеющей имя ОПЕРАНД, определяется тип операнда, обрабатываемого операцией. В пояснении указано, что двоичное число 01 означает регистровый операнд, 10 - операнд из памяти или регистра и число 00 - отсутствие операндов. В данном случае операции ADD и SUB работают с операндами из памяти и регистровыми, операция MOV1 обрабатывает регистровый операнд и операция JMP безоперандная или операнд содержится непосредственно в команде. Характеристика с именем ПРЕРЫВАНИЕ указывает, возможно ли прерывание после выполнения операции. В пояснении отмечено, что нулевое значение характеристики означает невозможность прерывания после выполнения операции, то есть прерывание не допускается после команды JMP. Последующие характеристики указывают время выполнения этих операций в микросекундах и длину соответствующих команд в битах.

Массив ОПЕРАНД включает регистры, триггеры, признаки, счетчики, то есть те элементы и параметры ЭВМ, которые используются при реализации операций. Описание массива ОПЕРАНД имеет следующую структуру:

ОПЕРАНД 01 , 02 , ..., 0i , ... , 0m **,

где O_i - описание i -го операнда массива. Полная структура описания конкретного операнда имеет следующий вид:

$N - I (M) [P] < C >$,

где N - номер операнда в массиве, I - его имя, M - число операндов с одним общим именем I , P - разрядность операнда, C - его начальное значение (состояние). Любая из компонент (M) , $[P]$ или $<C>$ может быть опущена из описания операнда, если $M = 1$, $P = 32$, или $C = 0$ соответственно.

Например, описание

ОПЕРАНД

1 - регистр адреса команд (16) $< 100 >$,

2 - регистр признаков (4),

3 - регистр ввода (8) $[5] **$

означает, что при реализации операций данной МПС используется 16-разрядный регистр адреса команд, имеющий исходное значение 100, 4-разрядный регистр признаков и пять 8-разрядных регистров ввода.

Функционирование МПС заключается в выполнении следующих действий:

- выбор команды из ПАМЯТИ по значению регистра адреса команд;
- изменение значений некоторых элементов массивов ПАМЯТЬ или

ОПЕРАНД в соответствии с алгоритмом выполнения выбранной команды.

3. Синтаксический контроль и преобразование в табличную форму

Описание МПС поступает на вход программ, которые проверяют его синтаксическую правильность и строят таблицы и рабочие комплексы. Таблицы строятся для упорядочения описания и обеспечения его наглядности. Рабочие комплексы используются при функционировании программной модели МПС. По описанию ПАМЯТЬ строятся таблица ТАПАМ и рабочий комплекс ПАМЯТЬ, представляющий символичный комплекс, в котором содержится последовательность слов или ячеек данных и программ. Все адреса скорректированы так, что слово, имеющее минимальный адрес, находится в первом элементе комплекса. В конце комплекса расположены списки начал, концов и количества участков памяти.

По описанию ОПЕРАЦИЯ строятся таблица ТАЦИЯ и рабочий комплекс ОПЕРАЦ, представляющий собой символичный комплекс, содержащий списки мнемочкодов и кодов операций и их характеристик.

По описанию ОПЕРАНД строятся таблица ТАРАНД и два рабочих

Л. А. Николаева

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ МИКРОПРОЦЕССОРНЫХ СИСТЕМ

1. Введение

В настоящее время микропроцессорные системы (МПС) становятся массовым средством вычислительной техники [1]. В процессе проектирования таких систем возникает необходимость в их имитационном моделировании на ЭВМ, что объясняется следующими причинами:

- малый объем оперативной памяти МПС;
- ограниченность состава внешних устройств МПС;
- необходимость параллельной разработки МПС и программного обеспечения (ПО);
- возможность комплексной отладки ПО с имитацией внешней среды.

Для имитации МПС необходимо иметь ее описание на функциональном уровне, определяющем список команд, структуру данных, временные соотношения, управляющие сигналы. Типовая МПС включает пять функциональных блоков: устройство ввода, память, арифметическое устройство, устройство управления, устройство вывода. Данные могут быть в прямом, дополнительном и обратном коде. Списки команд конкретных МПС и алгоритмы их функционирования могут существенно различаться, так как на выбор их влияет проблемная ориентация разрабатываемых МПС.

В данной статье сообщается об автоматизированной системе разработки имитаторов МПС на базе алгоритмического языка ЛЯПАС-М [2]. Система представляет пользователю следующие средства:

- 1) язык описания МПС на уровне систем команд;
- 2) расширяющийся набор программных моделей операций, выполняемых в МПС;
- 3) средства автоматической генерации имитатора конкретной МПС по ее программной модели функционирования (ПМФ).

Для разработки имитатора конкретной МПС пользователь данной системы должен описать МПС массивами, приведенными в п.2, подготовить ее ПМФ, если она не охватывается существующей в системе, и недостающие программы выполнения операций и с помощью генератора имитаторов (ГИМ) настроить ПМФ на имитацию данной МПС. Син-

таксический контроль описания приведен в п.3. ПМФ, реализованная в системе, описана в п.4. Генератор имитаторов и требования, предъявляемые к ПМФ, изложены в п.5. Все программные средства системы записаны на ДЯПАС -М и могут эксплуатироваться в рамках операционных систем ЕС и СМ ЭВМ.

2. Язык описания микропроцессорных систем

Описание МПС в системе выполняется с помощью трех массивов: ПАМЯТЬ, ОПЕРАЦИЯ и ОПЕРАНД. Массив ПАМЯТЬ представляет собой последовательность, составленную из слов (ячеек) ОЗУ, ПЗУ, ППЗУ. Разрядность слов может быть до 32 бит. В ПАМЯТИ содержатся данные и команды. ПАМЯТЬ можно разделить на участки двух типов, один из которых допускает только чтение (ПЗУ, ППЗУ), другой - чтение и запись в ОЗУ. Описание массива ПАМЯТЬ имеет следующую структуру:

ПАМЯТЬ П1П2... П_i... П_Р ** ,

где П_i - описание *i*-й группы массива. Полное описание *i*-й группы имеет следующий вид:

СЛОВО (С) ЧТЕНИЕ (И) ЗАПИСЬ (И) * ИМЯ (И) ,

где С указывает разрядность (количество бит) слова или ячейки памяти, а И - интервал или последовательность интервалов адресов слов памяти, допускающих чтение или запись и содержащих данные или программы с некоторым именем ИМЯ. Интервал может быть представлен одним числом или двумя, разделенными знаком " - ". Интервалы в последовательности разделяются запятыми. Слово СЛОВО, ЧТЕНИЕ, ЗАПИСЬ могут быть сокращены до одной буквы, перед ними и перед символом * могут стоять пробелы и в описании любое из них может быть опущено. Например, описание

ПАМЯТЬ С (8) Ч(1-100) Ч(120-300)

* Д (550-600) * П(200-300) З(550-600)

означает, что задана память, состоящая из последовательности 8-разрядных слов, при этом интервалы слов с номерами от 1 до 100 и от 120 до 300 открыты для чтения и интервал слов с номерами от 550 до 600 открыт для чтения и записи; в памяти в ячейках, имеющих номера от 550 до 600, находятся данные с именем Д, а в ячейках от 200 до 300 - программа с именем П.

Массив ОПЕРАЦИЯ - это список операций МП-системы (мнемикоды, десятичные коды) и их характеристик, указывающих, например, на возможность прерывания после выполнения операций, на необходимость записи в ячейку или чтения из ячейки памяти, на длительность

выполнения, длину команд. Описание массива ОПЕРАЦИЯ имеет следующую структуру:

ОПЕРАЦИЯ 01 02 ... 0i ... 0m **,

где 0i - описание i-й группы характеристик операций. Описание конкретной группы имеет следующий вид:

N / П / (C1[N1], ..., Ci[Ni], ..., Cn[Nn]),

где N - это общее название группы операций или характеристик, П - это некоторое пояснение характеристики, например, единица измерения; Ci - i-й код операции или значение i-й характеристики; Ni - количество соседних одинаковых значений характеристик. Компоненты [Ni] или / П / могут быть опущены, если Ni = 1 или нет пояснения П. Например, описание

ОПЕРАЦИЯ

МНЕМОНОД / БЦ / (ADD, SUB, MOV1, JMP)

КОД / 10 / (101, 102, 103, 104)

ОПЕРАНД / 01 * p, 10 - П / (3 [2], 1, 0)

ПРЕРЫВАНИЕ / 0 - нет / (1 [3], 0)

ВРЕМЯ / МКСЕК / (1, 2, 3 [2])

ДЛИНА / БИТ / (8 [2], 16, 8)

означает, что в МПС выполняются операции ADD, SUB, MOV1, JMP, записанные в буквенно-цифровом коде. Их десятичный код: 101, 102, 103, 104. По характеристике, имеющей имя ОПЕРАНД, определяется тип операнда, обрабатываемого операцией. В пояснении указано, что двоичное число 01 означает регистровый операнд, 10 - операнд из памяти или регистра и число 00 - отсутствие операндов. В данном случае операции ADD и SUB работают с операндами из памяти и регистровыми, операция MOV1 обрабатывает регистровый операнд и операция JMP безоперандная или операнд содержится непосредственно в команде. Характеристика с именем ПРЕРЫВАНИЕ указывает, возможно ли прерывание после выполнения операции. В пояснении отмечено, что нулевое значение характеристики означает невозможность прерывания после выполнения операции, то есть прерывание не допускается после команды JMP. Последующие характеристики указывают время выполнения этих операций в микросекундах и длину соответствующих команд в битах.

Массив ОПЕРАНД включает регистры, триггеры, признаки, счетчики, то есть те элементы и параметры ЭВМ, которые используются при реализации операций. Описание массива ОПЕРАНД имеет следующую структуру:

ОПЕРАНД 01, 02, ..., 0i, ..., 0m **,

где O_i - описание i -го операнда массива. Полная структура описания конкретного операнда имеет следующий вид:

$N - I (M) [P] <C>$,

где N - номер операнда в массиве, I - его имя, M - число операндов с одним общим именем I , P - разрядность операнда, C - его начальное значение (состояние). Любая из компонент (M) , $[P]$ или $<C>$ может быть опущена из описания операнда, если $M = 1$, $P = 32$, или $C = 0$ соответственно.

Например, описание

ОПЕРАНД

1 - регистр адреса команд (16) $<100>$,

2 - регистр признаков (4),

3 - регистр ввода (8) $[5] **$

означает, что при реализации операций данной МПС используется 16-разрядный регистр адреса команд, имеющий исходное значение 100, 4-разрядный регистр признаков и пять 8-разрядных регистров ввода.

Функционирование МПС заключается в выполнении следующих действий:

- выбор команды из ПАМЯТИ по значению регистра адреса команд;
- изменение значений некоторых элементов массивов ПАМЯТЬ или ОПЕРАНД в соответствии с алгоритмом выполнения выбранной команды.

3. Синтаксический контроль и преобразование в табличную форму

Описание МПС поступает на вход программ, которые проверяют его синтаксическую правильность и строят таблицы и рабочие комплексы. Таблицы строятся для упорядочения описания и обеспечения его наглядности. Рабочие комплексы используются при функционировании программной модели МПС. По описанию ПАМЯТЬ строятся таблица ТАПАМ и рабочий комплекс ПАМЯТЬ, представляющий символичный комплекс, в котором содержится последовательность слов или ячеек данных и программ. Все адреса скорректированы так, что слово, имеющее минимальный адрес, находится в первом элементе комплекса. В конце комплекса расположены списки начал, концов и количества участков памяти.

По описанию ОПЕРАЦИЯ строятся таблица ТАЦИЯ и рабочий комплекс ОПЕРАЦ, представляющий собой символичный комплекс, содержащий списки мнемочкодов и кодов операций и их характеристик.

По описанию ОПЕРАНД строятся таблица ТАРАНД и два рабочих

комплекса ОПЕРАН и ДОПЕРА. Элементы комплекса ОПЕРАН соответствуют операндам, зафиксированным в таблице ОПЕРАНД. Соответствующие элементы комплекса ДОПЕРА указывают количество левых свободных разрядов элементов комплекса ОПЕРАН.

Модуль контроля и преобразования МПС

Постановка задачи. Задан номер архива. Считать описание МПС из архива, проверить его синтаксическую правильность, построить таблицы ТАПАМ, ТАРАНД, ТАЦИЯ и рабочие комплексы и записать их в архив.

Макрооперация МОПИС (A/Z). Исходные данные: A - номер архива. Результат: Z $\neq$ 0, синтаксическая ошибка в описании или аварийный отказ при работе с архивом.

Модуль контроля и преобразования ПАМЯТИ

Постановка задачи: Задано описание МПС, номер символа в нем, с которого начинается описание ПАМЯТЬ, и номер архива. Требуется построить таблицу и рабочий комплекс ПАМЯТЬ и записать их в архив.

Если есть синтаксические ошибки, то тип каждой ошибки и ее адрес в описании занести в комплекс ошибок.

Макрооперация МОПАМ (Ф1, В, A/B, Л2, S). Исходные данные: Ф1 - описание МПС, В - начало описания ПАМЯТЬ, А - номер архива. Результат: В - конец описания ПАМЯТЬ, Л2 - комплекс ошибок; S $\neq$ 0, аварийный отказ при работе с архивом.

Упорядочение и построение таблицы ПАМЯТЬ

Постановка задачи. Заданы списки начал концов и типов участков памяти, списки начал, концов и имен программ и данных в памяти и номер архива. Требуется упорядочить списки по возрастанию начал участков памяти и начал данных и программ, построить таблицу и рабочий комплекс ПАМЯТЬ и записать их в архив.

Макрооперация УПОСТАПАМ (Л1, Л2, Ф3, Л4, Л5, Ф6, A/Л8, S). Исходные данные: Л1, Л2, Ф3 - списки начал, концов и типов участков памяти; Л4, Л5, Ф6, - списки начал, концов и имен программ и данных в памяти; А - номер архива. Результат: Л8 - комплекс ошибок, фиксирует большой размер таблицы; S $\neq$ 0, авария при записи в архив.

Модуль контроля и преобразования ОПЕРАЦИИ

Постановка задачи: Задано описание МПС, номер символа в нем, с которого начинается описание ОПЕРАЦИЯ, и номер архива. Требуется построить таблицу ТАЦИЯ и рабочий комплекс ОПЕРАНД и записать их в архив. Если есть синтаксические ошибки, то тип каждой ошибки и адрес в описании занести в комплекс ошибок.

Макрооперация МОЦИЯ ($\Phi 1$, В, А/В, Л2, S). Исходные данные: $\Phi 1$ - описание МПС, В - начало описания ОПЕРАЦИЯ, А - номер архива. Результат: В - конец описания ОПЕРАЦИЯ, Л2 - комплекс ошибок; S $\neq$ 0, аварийный отказ при работе с архивом.

Модуль контроля и преобразования ОПЕРАНДА

Постановка задачи. Задано описание МПС, номер символа в нем, с которого начинается описание ОПЕРАНД, и номер архива. Требуется построить таблицу ТАРАНД и рабочие комплексы ОПЕРАН и ДОПЕРА и записать их в архив. Если есть синтаксические ошибки, то тип каждой ошибки и ее адрес в описании занести в комплекс ошибок.

Макрооперация МОРАНД ($\Phi 1$, В, А/В, Л2, S). Исходные данные: $\Phi 1$ - описание МПС, В - начало описания ОПЕРАНД, А - номер архива. Результат: В - конец описания ОПЕРАНД, Л2 - комплекс ошибок; S $\neq$ 0, аварийный отказ при работе с архивом.

4. Программная модель функционирования МПС

Программная модель функционирования МПС - это программа, построенная по алгоритму, представленному на рис.1. Первым шагом при работе алгоритма является считывание из архива рабочих комплексов ПАМЯТЬ, ОПЕРАЦ, ОПЕРАН, ДОПЕРА, построенных по описанию МПС. В них отображены память МПС с данными и программами, набор операндов и их исходных значений, список операций и их характеристик. Следующим этапом проверяется возможность и наличие прерывания. Затем по алгоритму выбирается значение регистра адреса команд комплекса ОПЕРАН, указывающее адрес команды в комплексе ПАМЯТЬ, которую следует выполнить. Эта команда выбирается из ПАМЯТИ и анализируется. Определяется номер операции команды по ее коду в списке операций. Нулевой номер означает ошибочный код операции. Если характеристика операции указывает на обращение к па-

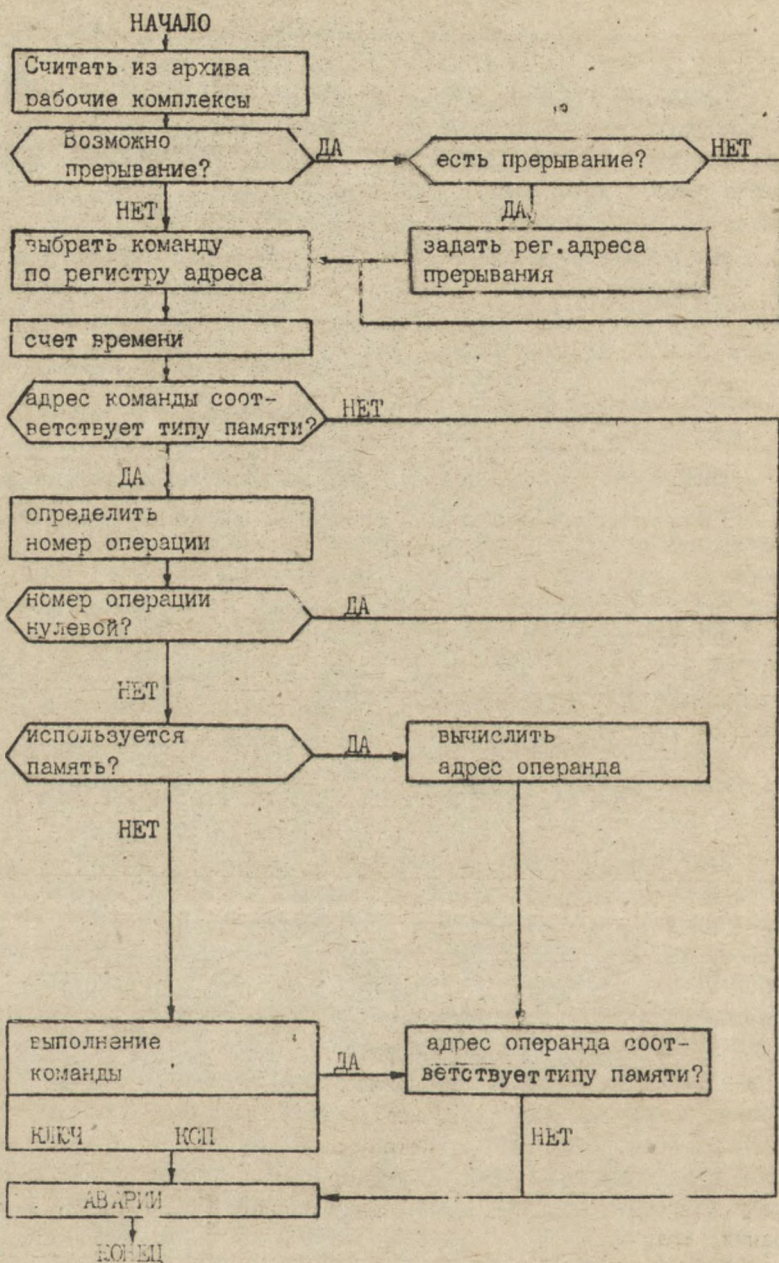


Рис. I

мости при ее выполнении, то определяется адрес операнда команды и возможность чтения или записи по этому адресу.

При нарушении соответствия выдается сообщение об ошибке. Если ошибок нет, команда выполняется. Этап выполнения команды записывается в виде двух макроопераций КЛЮЧ и КОП. КЛЮЧ представляет собой последовательность переключателей, выполняющих переход по номеру операции на макрооперацию, ее реализующую, а КОП - последовательность предложений с макрооперациями, реализующими операции МПС.

Постановка задачи. Задан номер архива, в котором хранятся рабочие комплексы, построенные по описанию МПС. Описать алгоритмы функционирования МПС в виде программы, использующей эти комплексы. При этом последовательность переключателей, выполняющих переход по номеру операции на макрооперацию, ее реализующую, обозначить макрооперацией КЛЮЧ (), а список макрооперации, реализующей операции МПС, - макрооперацией КОП ().

Макрооперация МПФ (А). Исходные данные: А - номер архива.

5. Генерация имитатора микропроцессорной системы

Постановка задачи. Заданы алгоритм функционирования МПС в виде программы ПМФ, алгоритмы операций МПС в виде подпрограмм, имена которых совпадают с мнемокодами этих операций, количество операций МПС, рабочий комплекс ОПЕРАЦ и номер списка характеристик в ОПЕРАЦ, указывающих использование регистров и памяти. При этом программа ПМФ написана так, что последовательность переключателей, выполняющих переход по номеру операции на макрооперацию, ее реализующую, обозначена макрооперацией КЛЮЧ (), а последовательность предложений с макрооперациями, реализующими операции МПС, - макрооперацией КОП (). Требуется преобразовать ПМФ в ИМИТАТОР. При этом изменить имя ПМФ на ИМИТ, вместо макрооперации КЛЮЧ () встроить соответствующие переключатели, вместо макрооперации КОП () вставить предложения.

Макрооперация ГИМ ($\Phi 1$, $\Phi 2$, С, Z, A/S). Исходные данные: $\Phi 1$ - ПМФ, $\Phi 2$ - ОПЕРАЦ, С - количество операций МПС; - номер списка характеристик в ОПЕРАЦ, С - количество операций МПС; - номер списка характеристик в ОПЕРАЦ, указывающих использование регистров и памяти; А - номер архива. Результат: S = 1, если нет символов в конце ПМФ; S = 2, если превышены размеры памяти при построении имитатора; S = 3, аварийный отказ при записи ИМИТА-

ТОРА в архив.

Литература

1. Гивоне Д., Россер Р. Микропроцессоры и микрокомпьютеры. Вводный курс. - М.: Мир, 1983. - 464 с.
2. Закревский А.Д., Торопов Н.Р. Система программирования ЛЯПАС-М. - Минск: Наука и техника, 1978. - 240 с.

А.Ю.Матросова, Г.А.Реморенко

УТОЧНЕНИЕ РЕЗУЛЬТАТОВ МОДЕЛИРОВАНИЯ В АЛФАВИТЕ $\mathcal{C}$

1. Постановка задачи

Задана комбинационная схема E , элементы которой реализуют элементарные функции алгебры логики. В частности, это может быть схема, составленная из элементов "и", "или", "не-и", "не-или", "не". Пусть $\mathcal{X}$ — множество переменных, сопоставляемых входам схемы, а $\mathcal{Y}$ — множество переменных, сопоставляемых ее выходам. Каждая переменная может принимать одно из семи значений множества $\mathcal{C}$, $\mathcal{C} = \{0, 1, X, \emptyset, *, \oplus\}$. Алфавит $\mathcal{C}$ аналогичен алфавиту, введенному в [1], но имеет другую интерпретацию символов.

Значение X будем интерпретировать как неопределенное и такое, которое не может быть доопределено до значений из $\{0, 1\}$. Значение $\emptyset$ будем интерпретировать как одно из двух возможных значений множества $\{0, 1\}$. Значение $\oplus$ — как одно из двух возможных значений множества $\{0, X\}$. Значение $*$ — как одно из двух возможных значений множества $\{1, X\}$. Наконец, значение $\emptyset$ — как одно из трех возможных значений множества $\{0, 1, X\}$.

Значения $0, 1, X$ будем называть элементарными, а значения $\emptyset, *, \oplus$ — неэлементарными. Каждому неэлементарному значению сопоставляется множество элементарных: например, значению $\emptyset$ — множество $\{0, X\}$, значению $\oplus$ — множество $\{0, 1, X\}$ и т.д. Элементарному значению сопоставляется одноэлементарное множество из него самого. Каждое значение условимся считать суммой соответствующих ему элементарных значений (слагаемых).

Обозначим через $\mathcal{E}_X$ вектор в алфавите $\mathcal{C}$, сопоставляемый входам схемы E , а через $\mathcal{E}_Y^I, \mathcal{E}_Y^H$ результаты подстановки вектора $\mathcal{E}_X$ в схему, вычисленные первым и вторым способом соответственно.

Комбинационной схеме E поставим во взаимно однозначное соответствие систему $\mathcal{F}(X)$ булевых функций, представленную формулами над множеством элементарных функций. Вид формул определяется структурой схемы E . Подстановку вектора $\mathcal{E}_X$ в схему E заменим подстановкой его в систему $\mathcal{F}(X)$.

Для изложения постановки задачи необходимо описать способы

вычисления векторов ε_y^I и ε_y^{II} , введенные в [2].

Обозначим через $M(\varepsilon_x)$ множество троичных векторов, полученных из вектора ε_x всевозможными заменами неэлементарных символов на соответствующие им слагаемые. Пусть $\varepsilon_x^0 = \{0x01\}$, тогда $M(\varepsilon_x) = \{0x01, 0x11, 0xx1, 1x01, 1x11, 1xx1\}$.

Введем операцию сложения троичных векторов. Вектор в алфавите S является суммой троичных векторов, если значения его компонент получаются сложением значений одноименных компонент троичных векторов. Например, задано множество троичных векторов:

$01x01, 11001, 1x001, xx000$, вектор $\varepsilon = 0xj00$ является суммой векторов этого множества.

Рассмотрим элементарные функции алгебры логики одной и двух переменных, аргументы функций определены на множестве S . Значение элементарной функции получается сложением элементарных значений, вычисленных для этой функции по соответствующим троичным векторам. Например, соотношение $0 \vee x = x$ получается следующим образом. Операция дивизионации выполняется над троичными векторами $0x$ и $1x$: $0 \vee x = x$, $1 \vee x = 1$. Значение x получается сложением значений x и 1 . Конкретный вид таблиц значений элементарных функций в алфавите S приведен в работе [1].

Вектор ε_y^I будем вычислять следующим образом. Подставим каждый из векторов множества $M(\varepsilon_x)$ в систему $\mathcal{F}(x)$ и вычислим соответствующий ему троичный вектор. Обозначим множество полученных таким образом векторов через M' . Сложим векторы множества M' , результат сложения обозначим через ε_y^I .

Вектор ε_y^{II} будем вычислять, подставляя ε_x в систему $\mathcal{F}(x)$ и вычисляя результат подстановки по таблицам элементарных функций.

Введем отношение поглощения между семизначными векторами, аналогичное отношению, определенному в [1].

Будем говорить, что вектор ε_1 поглощается вектором ε_2 и обозначать это символом $\varepsilon_1 \leq \varepsilon_2$, если для каждой компоненты этих векторов выполняется условие: множество слагаемых, соответствующее компоненте ε_1 , содержится в множестве слагаемых, соответствующих одноименной компоненте вектора ε_2 . Например, если $\varepsilon_1 = 00x01$ и $\varepsilon_2 = 00j00$, $\varepsilon_1 \leq \varepsilon_2$.

В работе [2] сформулирована и доказана теорема, согласно которой $\varepsilon_y^I \leq \varepsilon_y^{II}$. Вектор ε_y^I представляет "интегральную" информацию о поведении схемы на всем множестве $M(\varepsilon_x)$ троичных

векторов, порожденных ε_x . Вектор $\varepsilon_y^{\text{II}}$ задает «оценку сверху» этого поведения на том же множестве.

Пусть некоторая компонента вектора $\varepsilon_y^{\text{II}}$ принимает значение $\emptyset$. Это значит, что при подстановке трюичных векторов из $M(\varepsilon_x)$ в систему $\mathcal{F}(x)$ значение 1 невозможно на соответствующем компоненте выходе схемы. На этом выходе возможны значения из множества $\{0, x\}$, из соотношения $\varepsilon_y^{\text{I}} \leq \varepsilon_y^{\text{II}}$ следует, что не обязательно оба из них. Аналогичные утверждения можно сформулировать для значений $\emptyset, x$.

Пусть некоторая компонента вектора $\varepsilon_y^{\text{II}}$ принимает значение $\emptyset$. Это значит, что при подстановке трюичных векторов из множества $M(\varepsilon_x)$ в систему $\mathcal{F}(x)$ на соответствующем компоненте выходе схемы возможны значения из множества $\{0, 1, x\}$, причем не обязательно все из них.

Получение вектора ε_y^{I} связано с большим объемом вычислений. Получение вектора $\varepsilon_y^{\text{II}}$ по объему вычислений равносильно подстановке в систему $\mathcal{F}(x)$ одного трюичного вектора. Вычислив вектор $\varepsilon_y^{\text{II}}$, попробуем скорректировать значения его компонент так, чтобы в результате коррекции получить вектор ε_y^{I} .

Итак, рассмотрим следующую задачу. Известна структура схемы E , заданы векторы ε_x и $\varepsilon_y^{\text{II}}$, требуется получить вектор ε_y^{I} , не прибегая к перебору трюичных векторов множества $M(\varepsilon_x)$. Решение задачи сводим к анализу эквивалентных нормальных форм (ЭНФ) схемы E .

2. Анализ ЭНФ

В работе [3] отмечалось, что результат подстановки трюичного вектора в одновыходную комбинационную схему совпадает с результатом его подстановки в ЭНФ этой схемы. Здесь и далее мы будем иметь в виду ЭНФ, в которой вычеркнуты последовательности индексов, а несколько одинаковых букв одной и той же конъюнкции заменены одной буквой. Выясним условия, при которых ЭНФ на некотором трюичном векторе из $M(\varepsilon_x)$ принимает значение 1, а также значение 0 и, наконец, - значение x .

2.1. Обращение ЭНФ в единицу

Пусть схема E является одновыходной, обозначим символом $\mathcal{E}$ ее ЭНФ. Определим, существует ли в множестве $M(\varepsilon_x)$ трюичный вектор, обращающий $\mathcal{E}$ в единицу.

либо значение χ , либо значение 0. Теорема доказана.

Следствие. В множестве $M(\mathcal{E}_x)$ существует троичный вектор, на котором ЭНД $\mathcal{E}_x$ принимает значение χ .

Троичный вектор, на котором ЭНД $\mathcal{E}_x$ принимает значение χ , может обратить в единицу ДНД $\mathcal{D}_0$. В этом случае ЭНД $\mathcal{E}$ на рассматриваемом векторе принимает значение 1. В связи со сказанным нас будут интересовать такие троичные векторы из $M(\mathcal{E}_x)$, на которых ЭНД $\mathcal{E}_x$ принимает значение χ , а ДНД $\mathcal{D}_0$ — значение 0. Установление факта существования троичного вектора из множества $M(\mathcal{E}_x)$, на котором ЭНД $\mathcal{E}$ принимает значение χ , сведем к следующей процедуре.

1. Выберем очередную конъюнкцию из $\mathcal{E}_x$, найдем в ней переменные, принимающие в $\mathcal{E}_x$ значение 0. Припишем этим переменным значение 1.

2. В ДНД $\mathcal{D}_0$ заменим константами переменные, значения которых зафиксированы в п.1. Получим ДНД $\mathcal{D}_0^*$.

3. Решим уравнение $\mathcal{D}_0^* = 0$. Если решение существует, тогда в $M(\mathcal{E}_x)$ найдется троичный вектор, на котором ЭНД $\mathcal{E}$ принимает значение χ . Иначе, выберем следующую конъюнкцию из $\mathcal{E}_x$, переходя к п.1. Если все конъюнкции ЭНД $\mathcal{E}_x$ просмотрены, то в $M(\mathcal{E}_x)$ не существует вектора, на котором ЭНД $\mathcal{E}$ принимает значение χ .

3. Замена вектора $\mathcal{E}_y^{\bar{I}}$ вектором $\mathcal{E}_y^I$

Из соотношения $\mathcal{E}_y^I \leq \mathcal{E}_y^{\bar{I}}$ следует: если в компоненте вектора $\mathcal{E}_y^{\bar{I}}$ некоторое элементарное значение (слагаемое) из множества $\{0, 1, \chi\}$ отсутствует, то его нет и в соответствующей компоненте вектора $\mathcal{E}_y^I$. Имея вектор $\mathcal{E}_y^{\bar{I}}$, получим вектор $\mathcal{E}_y^I$, поступая следующим образом.

1. Рассмотрим очередную компоненту вектора $\mathcal{E}_y^{\bar{I}}$. Для значения этой компоненты построим множество ее элементарных значений (слагаемых).

2. Для каждого элементарного значения, анализируя ЭНД, сопоставляемую рассматриваемой компоненте, выясним, существует ли в $M(\mathcal{E}_x)$ вектор, на котором это значение достижимо. Если не все элементарные значения компоненты вектора $\mathcal{E}_y^{\bar{I}}$ достижимы, то, сложив достижимые из них, сформируем новое значение этой компоненты. Присваиваем его соответствующей компоненте вектора $\mathcal{E}_y^I$. Иначе, компоненте вектора $\mathcal{E}_y^I$ присваиваем старое значение рассматриваемой

мой компоненты вектора $\varepsilon_y^{\text{II}}$.

3. Перебрав все компоненты вектора $\varepsilon_y^{\text{II}}$, получим вектор ε_y^{I} .

4. Пример.

Рассмотрим пример. Пусть ЭНД Э некоторой схемы Е имеет вид

$$\bar{a}\bar{b}\bar{b} \vee \bar{a}\bar{b}c\bar{g} \vee \bar{a}\bar{b}d\bar{g} \vee \bar{a}\bar{c}\bar{d}\bar{b} \vee \bar{a}\bar{c}\bar{d}c\bar{g} \vee \bar{a}\bar{c}\bar{d}d\bar{g} \vee \bar{c}\bar{d}\bar{b}f \vee \\ \vee \bar{c}\bar{d}c\bar{t}g \vee \bar{c}\bar{d}d\bar{t}g \vee b\bar{t}\bar{g} \vee c\bar{g}\bar{g}\bar{f} \vee d\bar{g}\bar{g}\bar{f}. \quad a \ b \ c \ d \ t \ g$$

Входам схемы Е сопоставим вектор $\varepsilon_x = 0 \ x \ * \ 0 \ 1 \ 0$, для него $M(\varepsilon_x)$ состоит из 16 трюичных векторов. При подстановке ε_x в схему получаем вектор $\varepsilon_y^{\text{II}}$, $\varepsilon_y^{\text{II}} = 0$. Найдем вектор ε_y^{I} .

Сначала выясним, существует ли в $M(\varepsilon_x)$ вектор, обращающий ЭНД в нуль. Пронумеруем конъюнкции ЭНД слева направо от 1 до 12.

Упростим ЭНД, воспользовавшись следствиями из теорем 3-5.

Конъюнкции 3-9 вычеркиваются на основании следствия теоремы 4, а конъюнкции 11, 12 - на основании следствия теоремы 5. Из конъюнкций 1, 2, 10 вычеркиваются буквы на основании следствия теоремы 3. ЭНД $D_{\text{сокр}}$ имеет вид

$$D_{\text{сокр}} = a \vee a\bar{g} \vee \bar{g} = a \vee \bar{g}. \quad a \ g$$

Уравнение $D_{\text{сокр}} = 0$ имеет решение, вектор 0 1 обращает его в нуль. Следовательно, в $M(\varepsilon_x)$ существует вектор, на котором ЭНД Э обращается в нуль.

Определим, существует ли вектор, на котором ЭНД Э принимает значение x .

Построим ε^* :

$$\varepsilon^* = \bar{a}\bar{b}\bar{b} \vee \bar{a}\bar{b}c\bar{g} \vee \bar{a}\bar{b}d\bar{g} \vee \bar{a}\bar{c}\bar{d}\bar{b} \vee \bar{a}\bar{c}\bar{d}c\bar{g} \vee \bar{a}\bar{c}\bar{d}d\bar{g} \vee \\ \vee \bar{c}\bar{d}\bar{b} \vee \bar{c}\bar{d}c\bar{g} \vee \bar{c}\bar{d}d\bar{g} \vee b\bar{t}\bar{g} \vee c\bar{g}\bar{g}\bar{f} \vee d\bar{g}\bar{g}\bar{f}.$$

ДНД $D_\Phi = 0$, ЭНД ε_x имеет вид

$$\varepsilon_x = \bar{a}\bar{b}\bar{b} \vee \bar{a}\bar{b}c\bar{g} \vee \bar{a}\bar{b}d\bar{g} \vee \bar{a}\bar{c}\bar{d}\bar{b} \vee \bar{a}\bar{c}\bar{d}c\bar{g} \vee \bar{a}\bar{c}\bar{d}d\bar{g} \vee \bar{c}\bar{d}\bar{b} \vee \\ \vee \bar{c}\bar{d}c\bar{g} \vee \bar{c}\bar{d}d\bar{g} \vee b\bar{t}\bar{g}.$$

Выбираем первую конъюнкцию $\bar{a}\bar{b}\bar{b}$; на векторе $\begin{smallmatrix} a & b & c & d & t & g \\ 1 & x & x & x & 1 & 0 \end{smallmatrix}$ эта конъюнкция принимает значение x . Поскольку $D_\Phi = 0$, то ЭНД Э на этом векторе также принимает значение x . Итак, $\varepsilon_y^{\text{II}}$ совпадает с ε_y^{I} , $\varepsilon_y^{\text{II}} = \varepsilon_y^{\text{I}}$.

5. Коррекция результатов моделирования в схемах с памятью

Рассмотрим синхронное устройство, в котором комбинационная часть и память разделены. Элементы памяти реализуются, например,

линиями задержки, вынесенными в цепи обратных связей. На синхронное устройство поступает последовательность $\mathcal{Z}$ в алфавите $\mathcal{C}$. Начальное состояние устройства не известно и представляется вектором, компоненты которого сопоставляются линиям обратных связей и принимают значение X . Вычислим реакцию на последовательность двумя способами.

Обозначим через M множество троичных последовательностей, полученных из $\mathcal{Z}$ заменой неэлементарных значений элементарными. Для каждой последовательности из M вычислим выходную последовательность в предположении, что начальное состояние устройства не известно. Сложим векторы всех полученных таким образом выходных последовательностей. Будем складывать троичные векторы, сопоставляемые одноименным членам последовательностей, выполняя сложение по одноименным компонентам. Полученную в результате последовательность обозначим символом S^I .

Последовательность S^I получим, выполняя подстановки векторов последовательности $\mathcal{Z}$ в систему булевых функций, сопоставляемых структуре устройства. Вычисление векторов последовательности S^I сводится к многократному вычислению по очередному вектору ε_x вектора ε_y^I . Заметим, что вектор ε_x содержит составляющую, сопоставляемую псевдовходам устройства, а вектор ε_y^I — составляющую, сопоставляемую псевдовыходам этого устройства. Псевдовходы и псевдовыходы образуются при разрыве цепей обратных связей.

Поскольку структура устройства известна, то его входам и псевдовходам можно сопоставить эквивалентные нормальные формы. Это значит, что каждый вектор ε_y^I может быть заменен вектором ε_y^* в результате анализа эквивалентных нормальных форм. Составляющая ε_y^I , сопоставляемая псевдовходам, может быть использована для коррекции результатов дальнейших вычислений. Последовательность, полученную путем коррекции результатов на каждом шаге вычисления выходов по входам, обозначим символом S^* . Имеем $S^I \preceq S^* \preceq S^I$. Здесь подразумевается отношение поглощения между одноименными членами последовательностей.

Последовательность S^* может не совпадать с последовательностью S^I . Этот факт через составляющую ε_x , сопоставляемую псевдовходам, может вызвать несовпадение вторых векторов последовательностей S^I и S^* . Такое несовпадение может распространиться дальше.

Предложенный в работе метод коррекции результатов моделиро-

вания в алфавите S позволяет получать последовательность S^* , более "близкую" к S^I , чем S^II .

Чтобы получить S^I , не прибегая к перебору последовательностей из M , необходимо для выходов и псевдовыходов устройства построить эквивалентные нормальные формы по комбинационному эквиваленту устройства, содержащему ℓ его копий. Здесь ℓ - длина последовательности $\mathcal{L}$. Такой подход при достаточно больших значениях ℓ практически не реализуем, так же как практически не реализуем перебор всех трюичных последовательностей из M .

В заключение заметим, что для коррекции результатов моделирования в алфавите S можно избежать построения эквивалентных нормальных форм, заменяя анализ ЭНФ анализом структуры схемы.

Литература

1. Агibalов Г.П., Бузанов В.А., Липский В.В., Румянцев В.Ф. Математическая модель схемы из элементов с управляемой проводимостью.-Автоматика и телемеханика, 1982, № 9, с. 89-96.
2. Матросова А.Д. Семизначное моделирование асинхронных автоматов.-Автоматика и телемеханика (в печати).
3. Ангер С. Асинхронные последовательностные схемы.-М.: Наука, 1977.
4. Матросова А.Д. Построение полного теста для схем, синтезированных методом факторизации.-Автоматика и вычислительная техника, 1978, № 5, с. 42-46.
5. Закревский А.Д. К проверке тождеств в алгебре логики. - В кн.: Логический язык для представления алгоритмов синтеза релейных устройств. М.: Наука, 1966, с.159-163.

В.Д.Байда

К ПОСТРОЕНИЮ МНОЖЕСТВА МАКСИМАЛЬНЫХ УСТОЙЧИВЫХ СОСТОЯНИЙ АСИНХРОННОГО АВТОМАТА

Предлагается алгоритм построения множества максимальных устойчивых состояний по специальной системе ДНФ, представляющей функции переходов и выходов. Дается обоснование алгоритма.

Множество M^y максимальных устойчивых состояний является одной из форм компактного представления множества всех устойчивых состояний асинхронного автомата. Идея и метод построения этого множества были предложены в работе [1]. Свойства множества M^y , а также возможности его использования для построения функциональных моделей, учитывающих структурные состязания, были рассмотрены впоследствии в работах [2,3].

При построении M^y наибольшую сложность представляет получение множества $M_{\text{бн}}^y$ максимальных устойчивых состояний, для каждого из которых в соответствующем тричном векторе хотя бы одна компонента, сопоставляемая внутренней переменной, принимает определенное значение. В данной работе предлагается алгоритм построения множества $M_{\text{бн}}^y$. Множество $M_{\text{бн}}^y$ представляется в виде ДНФ $D_{\text{бн}}^y$. Алгоритм существенно проще и эффективнее процедуры построения множества $M_{\text{бн}}^y$, предложенной в работе [1].

Пусть асинхронный автомат описан системой частичных функций переходов-выходов, задающих области единичных и нулевых значений. Система функций представлена специальными дивъюнктивными нормальными формами $D_1, \bar{D}_1, \dots, D_{m,p}, \bar{D}_{m,p}$ отражающими информацию о состязаниях, определяемых структурной реализацией автомата. Здесь m - число выходных переменных автомата, p - число его внутренних переменных. Эта система ДНФ может быть получена специальной суперпозицией по структуре схемы [1], а также по функциональному описанию автомата, учитывающему его реальные состязания [3].

Пусть $\bar{x}_1, \dots, \bar{x}_p$ - внутренние переменные, сопоставляемые контурам обратных связей автомата, $x_1, \dots, x_p; y_1, \dots, y_m$ - входные и выходные переменные соответственно. Рассмотрим соответствующие внутренним переменным ДНФ $D_{m+1}, \bar{D}_{m+1}, \dots, D_{m+p}, \bar{D}_{m+p}$ и построим для них ДНФ $D_j^*, \bar{D}_j^*, \dots, D_p^*, \bar{D}_p^*$ следующим образом:

$$D_j^* = \bar{x}_j D_{m+j}; \quad \bar{D}_j^* = \bar{x}_j \bar{D}_{m+j}, \quad j = \overline{1, p}.$$

Здесь и далее символ "·" используется для обозначения операции логического умножения.

Пусть κ - некоторая конъюнкция в пространстве переменных $x_1, \dots, x_n$; $z_1, \dots, z_r$. Обозначим через $\mathcal{Z}$ - множество переменных $z_1, \dots, z_r$, а через $\mathcal{Z}(\kappa) = \{z_1, \dots, z_r\}$ - множество переменных из $\mathcal{Z}$, входящих в конъюнкцию κ .

Конъюнкцию κ будем называть 1(0)-устойчивой по переменным множества $\{z_1, \dots, z_r\} \subseteq \mathcal{Z}$, если:

а) $z_{jt} \in \mathcal{Z}(\kappa)$ и б) конъюнкция κ поглощается хотя бы одной конъюнкцией ДНФ $\mathcal{D}_{jt}^*(\mathcal{D}_{jt}^*)$, $t = \overline{1, s}$. Конъюнкцию κ , 1(0)-устойчивую по переменным множества $\mathcal{Z}(\kappa)$, будем называть 1(0)-устойчивой.

Конъюнкцию κ будем называть α -устойчивой по переменным множества $\{z_1, \dots, z_r\} \subseteq \mathcal{Z}$ если:

а) $z_{jt} \in \mathcal{Z} \setminus \mathcal{Z}(\kappa)$ и б) в ДНФ $\mathcal{D}_{m,j,t} \vee \overline{\mathcal{D}_{m,j,t}}$ не найдется ни одной конъюнкции, поглощающей конъюнкцию κ , $t = \overline{1, s}$. Конъюнкцию κ , α -устойчивую по переменным множества $\mathcal{Z} \setminus \mathcal{Z}(\kappa)$, будем называть α -устойчивой.

Конъюнкцию κ , 1(0)-устойчивую по переменным множества $\mathcal{Z}(\kappa)$ и α -устойчивую по переменным множества $\mathcal{Z} \setminus \mathcal{Z}(\kappa)$ будем называть устойчивой.

Будем говорить, что конъюнкция κ_1 α -поглощает конъюнкцию κ_2 и обозначать $\kappa_2 \preceq \kappa_1$, если $\kappa_2 \preceq \kappa_1$ (κ_1 поглощает κ_2) и $\mathcal{Z}(\kappa_1) = \mathcal{Z}(\kappa_2)$.

Конъюнкцию κ будем называть максимальной устойчивой (1(0)-устойчивой), если не существует конъюнкции κ_1 , удовлетворяющей условиям:

- а) κ_1 является устойчивой (1(0)-устойчивой) и
- б) $\kappa \preceq \kappa_1$.

Перечислим некоторые основные свойства конъюнкций, 1(0)-устойчивых и α -устойчивых по подмножеству переменных из $\mathcal{Z}$.

1. Если κ_1 1(0)-устойчива по переменным множества $\mathcal{Z}(\kappa_1)$, а κ_2 такова, что $\kappa_2 \preceq \kappa_1$, то κ_2 также 1(0)-устойчива по переменным множества $\mathcal{Z}(\kappa_1)$. Если при этом $\kappa_2 \preceq \kappa_1$, то κ_2 - 1(0)-устойчива по переменным множества $\mathcal{Z}(\kappa_2) = \mathcal{Z}(\kappa_1)$.
2. Если κ_1 α -устойчива по переменным множества $\mathcal{Z} \setminus \mathcal{Z}(\kappa_1)$, а κ_2 такова, что $\kappa_1 \preceq \kappa_2$, то κ_2 также α -устойчива по переменным множества $\mathcal{Z} \setminus \mathcal{Z}(\kappa_1)$. Если при этом $\kappa_1 \preceq \kappa_2$, то κ_2 - α -устойчива по переменным множества $\mathcal{Z} \setminus \mathcal{Z}(\kappa_2) = \mathcal{Z} \setminus \mathcal{Z}(\kappa_1)$.

3. Если $\kappa_1 - 1(0)$ -устойчива по переменным множества $\mathcal{Z}(\kappa_1)$, $\kappa_2 - 1(0)$ -устойчива по переменным множества $\mathcal{Z} \setminus \mathcal{Z}(\kappa_2)$ и $\kappa_2 \leq \kappa_1$, то $\kappa_1 - 1(0)$ -устойчива по переменным множества $\mathcal{Z} \setminus \mathcal{Z}(\kappa_2)$, а $\kappa_2 - 1(0)$ -устойчива по переменным множества $\mathcal{Z}(\kappa_1)$. Если при этом $\kappa_2 \leq \kappa_1$, то κ_1, κ_2 - устойчивы.

4. Если $\kappa_1 - 1(0)$ -устойчива по переменным множества $\mathcal{Z}(\kappa_1)$, а $\kappa_2 - 1(0)$ -устойчива по переменным множества $\mathcal{Z}(\kappa_2)$, то конъюнкция $\kappa = \kappa_1 \cdot \kappa_2 - 1(0)$ -устойчива по переменным множества $\mathcal{Z}(\kappa) = \mathcal{Z}(\kappa_1) \cup \mathcal{Z}(\kappa_2)$.

Очевидно, что конъюнкция κ является максимальной устойчивой, если она является максимальной $1(0)$ -устойчивой и χ -устойчивой одновременно. Таким образом, для построения ДНФ $\mathcal{D}_{\delta\mathcal{H}}^y$ достаточно построить ДНФ $\mathcal{D}^\sigma$, представляющую все максимальные $1(0)$ -устойчивые конъюнкции и выделить из $\mathcal{D}^\sigma$ те конъюнкции, которые являются χ -устойчивыми. Эта идея лежит в основе предлагаемого ниже алгоритма.

Пусть $\mathcal{D}_j^y = \mathcal{D}_j^* \vee \bar{\mathcal{D}}_j^*$. Каждая конъюнкция из ДНФ $\mathcal{D}_j^y$ $1(0)$ -устойчива по $\mathcal{Z}_j$. Кроме того, если конъюнкция κ поглощается некоторой конъюнкцией из $\mathcal{D}_j^y$, то конъюнкция κ также $1(0)$ -устойчива по $\mathcal{Z}_j$. Иными словами, ДНФ $\mathcal{D}_j^y$ задает область $1(0)$ -устойчивости по переменной $\mathcal{Z}_j$.

Пусть κ - некоторая конъюнкция и $\mathcal{Z}(\kappa) = \{\mathcal{Z}_{i1}, \dots, \mathcal{Z}_{is}\}$. Обозначим через κ_{is}^y некоторую конъюнкцию ДНФ $\mathcal{D}_{is}^y$.

Теорема 1. Если конъюнкция κ является максимальной $1(0)$ -устойчивой, то она представима в виде логического произведения $\kappa_{i1}^y \dots \kappa_{is}^y$.

Доказательство. Пусть κ - максимальная $1(0)$ -устойчивая конъюнкция. По определению $1(0)$ -устойчивости для любой переменной $\mathcal{Z}_{i2} \in \mathcal{Z}(\kappa)$ в ДНФ $\mathcal{D}_{i2}^y$ найдется конъюнкция κ_{i2}^y такая, что $\kappa \leq \kappa_{i2}^y$, $i = 1, s$. Пусть $\kappa^* = \kappa_{i1}^y \dots \kappa_{is}^y$. Тогда, очевидно, $\kappa \leq \kappa^*$, а поскольку $\mathcal{Z}(\kappa^*) = \mathcal{Z}(\kappa)$, то $\kappa \leq \kappa^*$.

Из построения κ^* следует, что $\kappa^* - 1(0)$ -устойчивая, а поскольку κ - максимальная $1(0)$ -устойчивая, то $\kappa = \kappa^*$, откуда следует, что $\kappa = \kappa_{i1}^y \dots \kappa_{is}^y$. Теорема доказана.

Заметим, что обратное утверждение в общем случае неверно.

Алгоритм построения $\mathcal{D}_{\delta\mathcal{H}}^y$ заключается в получении всех конъюнкций, удовлетворяющих условиям теоремы 1. в выделении среди них максимальных $1(0)$ устойчивых путем выполнения операции χ -поглощения и в проверке выполнения для этих конъюнкций условия χ -устойчивости.

Отметим, что выполнение операции χ -поглощения является правомерным на любом шаге алгоритма, поскольку исключает потерю решений. При перемножении дизъюнктивных нормальных форм операции склеивания и поглощения конъюнкций не допускаются.

Предлагаемый алгоритм построения $D_{\delta_n}^y$ состоит из двух последовательных этапов.

Этап 1. Шаг 0. Положим $D' = D_1^y \vee D_2^y \vee \dots \vee D_p^y$, $Z = \emptyset$, $i = 1$.

Шаг i . Выберем из ДНФ D' очередную конъюнкцию κ^i .

Если все конъюнкции ДНФ D' просмотрены, то при $i > 1$ положим $i = i - 1$ и перейдем к шагу i , а при $i = 1$ выполним в построенной ДНФ D^* операции χ -поглощения и перейдем к этапу 2. Иначе, обозначим через $Z(\kappa^i)$ множество переменных из $z_1, \dots, z_p$, входящих в конъюнкцию κ^i . Если $Z(\kappa^i) \setminus Z^{i-1} = \emptyset$, то включим κ^i в ДНФ D^* и перейдем к шагу i , иначе, выберем некоторую переменную z_j , $z_j \in Z(\kappa^i) \setminus Z^{i-1}$ и построим ДНФ D^{i+1} . ДНФ D^{i+1} получается путем логического перемножения конъюнкции κ^i и ДНФ $D_j^*(D^*)$ при условии, что z_j входит в κ^i без знака инверсии (со знаком инверсии). Если $D^{i+1} = 0$, то перейдем к шагу i , иначе, положим $Z^{i+1} = Z^i \cup \{z_j\}$, $i = i + 1$ и перейдем к шагу i .

Этап 2. Шаг 0. Положим $D^0 = D^*$, где D^* - ДНФ, полученная в результате выполнения этапа 1. Положим $i = 1$.

Шаг i . Построим ДНФ D^i , $D^i = D^{i-1} \cdot D^{i-1} \vee D^{i-1}$.

Выполним в D^i операции χ -поглощения. Если $D^i \neq D^{i-1}$, то положим $i = i + 1$ и перейдем к шагу i , иначе, положим $D^\sigma = D^i$, удалим из D^σ все те конъюнкции κ^i , которые не являются χ -устойчивыми хотя бы по одной переменной z_j , не входящей в $Z(\kappa^i)$. Полученную ДНФ обозначим через $D_{\delta_n}^y$ и окончим процедуру.

В отличие от алгоритма, предложенного в работе [1], поиска конъюнкций, удовлетворяющих условию теоремы 1, ведется направленно. Проверка выполнения условия 1(0)-устойчивости для получаемых конъюнкций не выполняется совсем, а проверка условия χ -устойчивости выполняется только один раз для ДНФ D^σ .

Обоснованием изложенного алгоритма может служить следующая теорема.

Теорема 2. Для любой конъюнкции κ , устойчивой по переменным множества $Z(\kappa)$, в ДНФ D^σ найдется конъюнкция κ^σ такая, что $\kappa \leq \kappa^\sigma$.

Доказательство. Пусть κ удовлетворяет условию теоремы. Это значит, что для любой переменной $z_{i_2} \in Z(\kappa)$ в ДНФ $D_{i_2}^y$ найдется конъюнкция $\kappa_{i_2}^y$ такая, что $\kappa \leq \kappa_{i_2}^y$. Из построения ДНФ D^*

(этап 1) следует, что для любой переменной $x_{iz} \in Z(\kappa)$ в $\mathcal{D}^*$ найдется конъюнкция κ_{iz}^* такая, что $\kappa \leq \kappa_{iz}^*$. Конъюнкции κ_{iz}^* получаются путем логического перемножения конъюнкций κ_{iz}^y .

Из построения ДНФ $\mathcal{D}^\sigma$ (этап 2) следует, что в $\mathcal{D}^\sigma$ найдется конъюнкция κ^σ такая, что $\kappa \leq \kappa^\sigma$ и $Z(\kappa) = Z(\kappa^\sigma)$. Конъюнкция κ^σ получается путем логического перемножения конъюнкций типа κ_{iz}^* . Это и означает, что в ДНФ $\mathcal{D}^\sigma$ существует конъюнкция κ^σ такая, что $\kappa \lesssim \kappa^\sigma$. Теорема доказана.

Литература

1. Матросова А.Ю., Чоп Н.В. Построение множества максимальных устойчивых состояний для асинхронного автомата. - Автоматика и телемеханика, 1982, № 2, с. 67-74.
2. Байда В.Д., Матросова А.Ю. Исследование функциональной модели асинхронного автомата, учитывающей структурные состязания. - Автоматика и телемеханика, 1982, № 4, с. 94-103.
3. Байда В.Д., Матросова А.Ю. Построение и использование одношаговой функциональной модели асинхронного автомата. - Автоматика и телемеханика, 1984, № 5, с. 117-124.

Ю.Д.Черкашин

МИНИМИЗАЦИЯ ЧИСЛА ОБРАТНЫХ СВЯЗЕЙ В ФУНКЦИОНАЛЬНЫХ СХЕМАХ

Пусть имеется произвольная функциональная схема. Данной схеме ставится в соответствие ориентированный граф $G = (X, \Gamma)$, вершины которого соответствуют элементам схемы, а каждая дуга (x, y) соответствует связи выходного полюса элемента схемы, соответствующего вершине x , с одним из входных полюсов элемента, соответствующего вершине y . Обратной связью схемы называется последовательность $a_1, a_2, \dots, a_m$ некоторых её элементов, в которой выходной полюс элемента a_i соединён с одним из входных полюсов элемента a_{i+1} , $i = 1, \dots, m-1$ и выходной полюс элемента a_m соединён с входным полюсом элемента a_1 . В графе G обратной связи схемы соответствует контур. Минимизация числа обратных связей в функциональной схеме заключается в нахождении минимального по мощности подмножества Z вершин графа G , при удалении которых вместе с инцидентными им дугами из графа G получается граф без контуров.

Таким образом (для ориентированного графа G), эта задача была поставлена в работе [1] и предложен алгоритм её приближённого решения. Было замечено, что быстродействие этого алгоритма быстро падает с ростом числа вершин графа. Целью данной работы является преодоление указанного недостатка.

Кратчайшим контуром называется контур, содержащий минимальное число вершин графа.

Предлагается приближенный алгоритм решения поставленной задачи. Алгоритм состоит из n шагов, где n — мощность искомого множества Z . На первом шаге $G_1 = G$, $Z_1 = \emptyset$; на n -м шаге $Z_{n+1} = Z$; при этом k -й шаг алгоритма состоит из следующих действий.

1. Поиск какого-либо кратчайшего контура (будем называть его опорным) подграфа G_k . Если опорного контура нет, то $Z = Z_k$ и $k = n$.
2. Нахождение вершины x_k в опорном контуре, через которую проходит максимальное число кратчайших контуров.
3. Построение $Z_{k+1} = Z_k \cup \{x_k\}$ и подграфа G_{k+1} , имеющего все вершины, кроме x_k , графа G_k и все его дуги, кроме дуг, инцидентных вершине x_k .

Рассмотрим подробно состав первого и второго действий дан-

ного алгоритма (действие 3 - тривиальное). В начале первого действия строится подмножество Y_k всех таких вершин графа G_k , что через каждую из них проходит хотя бы один кратчайший контур этого графа. Множество Y_k строится следующим образом. Рассматриваются последовательно все вершины графа G_k . Для каждой вершины a_i находится число q_i , равное минимальному числу вершин в контуре, проходящем через эту вершину. Если через a_i не проходит ни одного контура, то она не включается в Y_k . Пусть p - минимальное число из всех чисел q_i , тогда

$$a_i \in Y_k \leftrightarrow q_i = p. \text{ Число } q_i \text{ находится следующим образом.}$$

Строится последовательность множеств вершин: $M_1^i, M_2^i, \dots, M_t^i$, где $M_1^i = \{a_i\}$, а при $2 \leq t \leq i$, $M_t^i = N_t^i \setminus \bigcup_{j=1}^{t-1} M_j^i$, где $N_t^i = \bigcup_{m_j \in M_{t-1}^i} \Gamma m_j$.

Число q_i есть минимальное среди чисел из множества $\{1, 2, \dots, i\}$, удовлетворяющее условию

$$\Gamma^{-1} a_i \cap M_q^i \neq \emptyset.$$

После того, как построено множество Y_k , строится множество L вершин опорного контура следующим образом. Из множества Y_k выбирается произвольная вершина a_i и строится последовательность множеств $Q_1^i, Q_2^i, \dots, Q_p^i$, где $Q_t^i = M_t^i \cap Y_k$ при $1 \leq t \leq p$. Из множества $Q_p^i \cap \Gamma^{-1} a_i$ выбирается произвольная вершина c_p и строится последовательность вершин $c_p, c_{p-1}, \dots, c_1$ таким образом, что при $p-1 \geq j \geq 1$ выполняется $c_j \in \Gamma^{-1} c_{j+1} \cap Q_j^i$. Искомое множество $L = \{c_1, c_2, \dots, c_p\}$.

Число кратчайших контуров, проходящих через какую-либо вершину данного графа, называется весом этой вершины. После построения множества L вершин опорного контура находятся веса всех вершин, принадлежащих множеству L , и выбирается вершина x_k с максимальным весом. Рассмотрим алгоритм нахождения веса $\rho(a_i)$ произвольной вершины $a_i \in Y_k$. Нетрудно убедиться в справедливости следующих утверждений.

Утверждение 1.

В любом кратчайшем контуре, проходящем через вершину $a_i \in Y_k$, все вершины $c_j, j = 1, 2, \dots, p$ можно упорядочить так, что $c_j \in Q_j^i \cap \Gamma c_{j-1}, j = 2, 3, \dots, p$; c_1 есть вершина a_i .

Утверждение 2.

Пусть $R(c_1, c_2, \dots, c_{t-1}, c_t)$, где $c_j \in Q_j^i \cap \Gamma c_{j-1}$,

$j = 2, \dots, t$ — число кратчайших контуров, множество вершин каждого из которых содержит все вершины $c_1, c_2, \dots, c_t$, тогда имеет место следующее равенство:

$$R(c_1, \dots, c_{t-1}) = \sum_{c_t \in Q_t^i \cap \Gamma_{c_{t-1}}} R(c_1, \dots, c_t). \quad (I)$$

Утверждение 3.

Пусть $c_j \in Q_j^i \cap \Gamma_{c_{j-1}}, j = 2, \dots, t; q_2 \in Q_2^i \cap \Gamma_{c_1};$

$q_j \in Q_j^i \cap \Gamma_{q_{j-1}}, j = 3, \dots, t-1; c_t \in Q_t^i \cap \Gamma_{q_{t-1}},$

тогда имеет место равенство

$$R(c_1, c_2, \dots, c_{t-1}, c_t) = R(c_1, q_2, \dots, q_{t-1}, c_t). \quad (2)$$

Для доказательства достаточно воспользоваться формулой (I), применяя её поочередно для разложения левой и правой частей равенства (2), и утверждением I, на основании которого множества Q_j^i одинаковы при соответствующих шагах разложения левой и правой частей равенства (2). По определению веса $p(a_i)$ и числа $R(a_i)$ имеет место равенство

$$p(a_i) = R(a_i). \quad (3)$$

Применяя формулы (3) и (I), нетрудно найти искомый вес $p(a_i)$, формулу (2) при этом можно использовать для сокращения числа членов разложения. Нетрудно заметить, что число членов разложения с учётом сокращения не превышает n_k^2 , где n_k — число вершин в подграфе G_k . Общее число членов разложения при построении множества Y_k и выборе вершины x_k с максимальным весом не превышает числа $(p+1)n_k^2$, где p — число вершин в кратчайшем контуре. Если учесть, что $p \leq m$, $n_k \leq m$, $n \leq m$, то общее число членов разложения в предложенном алгоритме поиска множества Z не превышает m^4 , где m — число вершин в данном графе G . На основании этой оценки можно предположить, что быстродействие этого алгоритма значительно выше, чем алгоритма в [1]. Это предположение оправдалось при сравнительных испытаниях алгоритмов. Качество решения при этом (число элементов в множестве Z) ухудшилось незначительно.

Экспериментально статистические исследования алгоритма были проведены на ЭВМ ЕС 1022, снабжённой системой автоматического программирования ПС ЛЯПАС-М[2]. Для этого была сос-

тавлена программа на языке ЛЯПАС М [2] (РАЗМЫК4), представляющая данный алгоритм. На этом же языке были составлены и исследованы для сравнения программы еще трёх алгоритмов поиска множества Z : РАЗМЫК3, РАЗМЫК2 и РАЗМЫК1. Программа РАЗМЫК3 является аналогом программы РАЗМЫКОН1 [1]. Алгоритм, представленный программой РАЗМЫК2, есть последовательность шагов, на каждом из которых (на шаге номер k) в множество Z включается вершина из подграфа G_k , через которую проходит хотя бы один кратчайший контур. На каждом шаге алгоритма РАЗМЫК1 в множество Z включается вершина, через которую проходит хотя бы один контур.

Испитание каждой программы проводилось путём решения с её помощью нескольких серий примеров. Для примеров применялись псевдослучайные булевы матрицы (матрицы смежности), элементы которых не коррелированы между собой и с заданными вероятностями принимают значения "0" и "1". Для каждой серии задавались значения параметров, которыми являются: m — число вершин в графе, p — вероятность принятия значения "1" любым элементом матрицы смежности и ℓ — число примеров в серии. Далее обозначим $n = \frac{121}{\ell}$ — среднее число элементов в решении и t — среднее время решения одного примера в серии в секундах. Результаты исследований представлены в следующей таблице.

Про- грамма	$p = \frac{1}{8}, m = 50$			$p = \frac{1}{16}, m = 50$			$p = \frac{1}{16}, m = 60$			$p = \frac{1}{16}, m = 70$			$p = \frac{1}{16}, m = 80$		
	n	t	ℓ	n	t	ℓ	n	t	ℓ	n	t	ℓ	n	t	ℓ
РАЗМЫК1	37,6	1	100	26,3	1	100	35,8	1	100	43,8	3	5	51,4	3	5
РАЗМЫК2	30,6	3	100	18,6	3	100	26,0	6	100	31,8	9	5	40,0	15	5
РАЗМЫК3	25,1	54	30	14,8	21	30	18,6	84	10	24,0	906	1	32,0	2016	1
РАЗМЫК4	26,2	12	30	16,2	9	30	22,1	18	10	26,0	30	1	34,0	42	1

Литература

1. Черкашин Ю.Д. О размыкании контуров графа. — В кн.: Вычислительная техника в машиностроении. Минск: Наука и техника, 1973, с. 98–102.
2. Закревский А.Д., Торопов Н.Р. Система программирования ЛЯПАС-М. — Минск: Наука и техника, 1978. — 238 с.

В.А.Бузанов

О ВЫЧИСЛЕНИИ НЕИСПРАВНОСТЕЙ В СТРУКТУРНОМ АВТОМАТЕ

Если удовлетворительной моделью неисправности считается константная неисправность на полюсах и обратных связях схемы, что часто встречается в диагностике интегральных схем, то за модель схемы может быть принят структурный автомат. Ниже предлагается способ вычисления указанных неисправностей, обнаруживаемых заданной входной последовательностью автомата в многозначном алфавите. Алфавит принят многозначным в связи с увеличением интереса к переключательным схемам [1,2]. Как будет видно, предлагаемый метод вычисления неисправностей имеет разумную сложность, если сложность представления функций автомата практически приемлема.

Под структурным автоматом будем понимать множество $\mathcal{A} = (C^n, C^p, C^m, \psi, \varphi)$, где C^n , C^p и C^m (n -я, p -я и m -я декартовы степени множества $C = \{0, 1, \dots, k-1\}$) - входной, внутренний и выходной алфавиты в структурном алфавите C , а ψ и φ - функции переходов и выходов соответственно, заданные в интервальной форме, т.е. отображениями

$$\begin{aligned}\psi: D\psi \subseteq \tilde{C}^{n+p} &\rightarrow C^p \\ \varphi: D\varphi \subseteq \tilde{C}^{n+p} &\rightarrow C^m.\end{aligned}$$

Здесь, как обычно, $D\psi$ и $D\varphi$ - области определения функций ψ и φ , а $\tilde{C}$ - множество всех подмножеств множества C .

Расширенную функцию выходов автомата $\mathcal{A}$ для заданных входной последовательности $A = \alpha_1, \alpha_2, \dots, \alpha_s$, $\alpha_i \in C^n$, и начального состояния $q_1 \in C^p$ определим общепринятым способом как вектор-функцию

$$\varphi(A, q_1) = \varphi(\gamma_1) \varphi(\gamma_2) \dots \varphi(\gamma_s),$$

где $\gamma_1 = \alpha_1 q_1$, а $\gamma_i = \alpha_i \varphi(\gamma_{i-1})$, $i = 2, \dots, s$.

Элементы множества C называются векторами; элементы множества $\tilde{C}$ - интервалами. Аналогично, элементы множества C_Λ , где $C_\Lambda = C \cup \{\Lambda\}$, называются Λ -векторами, а элементы множества $\tilde{C}_\Lambda$ - Λ -интервалами. Для пары Λ -интервалов $u = u_1 u_2 \dots u_l$ и $v = v_1 v_2 \dots v_l$ определим операции $u + v = (u_1 + v_1) \dots (u_l + v_l)$ и $u \div v = (u_1 \div v_1) \dots (u_l \div v_l)$ следующим образом:

$$\begin{aligned}
 u_i + v_i &= \begin{cases} v_i \setminus \{\Lambda\}, & \text{если } u_i = \Lambda, \\ u_i \setminus \{\Lambda\}, & \text{если } u_i \neq \Lambda, \end{cases} \\
 u_i \div v_i &= \begin{cases} u_i \cup \{\Lambda\}, & \text{если } u_i \cap v_i \neq \emptyset, \\ u_i, & \text{если } u_i \cap v_i = \emptyset. \end{cases}
 \end{aligned}$$

Аргументной неисправностью автомата $\mathcal{A}$ (далее неисправностью) называется Λ -вектор $e \in C_{\Lambda}^{n+p}$. Расширенная функция $e\varphi$ выходов автомата $\mathcal{A}$, находящегося в состоянии q_i^* и имеющего неисправность e при воздействии на автомат последовательности $\#$, определяется как

$$e\varphi(\#, q_i^*) = \varphi(e + \gamma_1^*) \varphi(e + \gamma_2^*) \dots \varphi(e + \gamma_s^*),$$

где $\gamma_1^* = \alpha_1 q_1^*$, $\gamma_i^* = \alpha_i \varphi(e + \gamma_{i-1}^*)$, $i = 2, \dots, s$.

Поскольку $\alpha + \beta + \beta = \alpha$, выражение для $e\varphi$ можно переписать как

$$\begin{aligned}
 e\varphi(\#, q_i^*) &= \varphi(e + \gamma_1^* + \gamma_1 + \gamma_1) \varphi(e + \gamma_2^* + \gamma_2 + \gamma_2) \dots \varphi(e + \gamma_s^* + \gamma_s + \gamma_s) = \\
 &= \varphi(\varepsilon_1 + \gamma_1) \varphi(\varepsilon_2 + \gamma_2) \dots \varphi(\varepsilon_s + \gamma_s) \equiv \varphi(\#, q_i, \varepsilon_1, \dots, \varepsilon_s),
 \end{aligned}$$

где Λ -вектор из $\varepsilon_i = e + \gamma_i^* + \gamma_i$ может быть истолкован как проявление неисправности e при действии на автомат i -го члена входной последовательности $\#$. В этом случае задача вычисления неисправностей автомата $\mathcal{A}$, обнаруживаемых входной последовательностью $\#$, понимается как задача вычисления Λ -векторов, для которых

$$\varphi(\#, q_i, \varepsilon_1, \dots, \varepsilon_s) \bar{R} \varphi(\#, q_i),$$

где $\bar{R}$ - некоторое отношение различения на множестве значений функции φ , и для последовательностей выполняется

$$(\alpha_1, \alpha_2, \dots, \alpha_s) \bar{R} (\beta_1, \beta_2, \dots, \beta_s) \Leftrightarrow \exists i = 1, 2, \dots, s (\alpha_i \bar{R} \beta_i).$$

Решение задачи вычисления множества неисправностей ε_i , обнаруживаемых заданной входной последовательностью $\#$, основывается на следующем утверждении.

Пусть $\mathcal{U}$ - множество интервалов (элементов множества $\bar{C}^{n+p}$) на которых задана функция φ .

Теорема. $\varphi(\varepsilon_i + \gamma_i) \bar{R} \varphi(\gamma_i) \Leftrightarrow \exists u \in \mathcal{U} (\varphi(u) \bar{R} \varphi(\gamma_i) \& \varepsilon_i \in u \div \gamma_i)$.

Доказательство. Необходимость. $\varphi(\varepsilon_i + \gamma_i) \bar{R} \varphi(\gamma_i) \rightarrow$

$$\exists u \in \mathcal{U} (\varphi(u) \bar{R} \varphi(\gamma_i) \& \varepsilon_i + \gamma_i \in u) \rightarrow$$

$$\begin{aligned}
 &\exists u \in \mathcal{U} (\varphi(u) \bar{R} \varphi(\gamma_i) \& \forall k (\varepsilon_{ik} = \Lambda \& \gamma_{ik} \in u_k \vee \varepsilon_{ik} \in u_k \& \gamma_{ik} \notin u_k \\
 &\vee \varepsilon_{ik} \in u_k \& \gamma_{ik} \in u_k)) \rightarrow
 \end{aligned}$$

$$\exists u \in \mathcal{U} (\varphi(u) \bar{R} \varphi(\gamma_i) \& \forall k (u_k \div \gamma_{ik} = u_k \cup \{\Lambda\} \vee u_k \div \gamma_{ik} = u_k \vee$$

$$\forall u_k \div \gamma_{ik} = u_k \cup \{\Delta\} \rightarrow$$

$$\exists u \in U(\varphi(u) \bar{r}\varphi(\gamma_i) \& \forall k(\varepsilon_{ik} \in u_k \div \gamma_{ik})) \rightarrow \exists u \in U(\varphi(u) \bar{r}\varphi(\gamma_i) \& \varepsilon_i \in u \div \gamma_i).$$

Достаточность. $\exists u \in U(\varphi(u) \bar{r}\varphi(\gamma_i) \& \varepsilon_i \in u \div \gamma_i) \rightarrow$

$$\exists u \in U(\varphi(u) \bar{r}\varphi(\gamma_i) \& \forall k(\varepsilon_{ik} \in u_k \vee \varepsilon_{ik} \in u_k \cup \{\Delta\})) \rightarrow$$

$$\exists u \in U(\varphi(u) \bar{r}\varphi(\gamma_i) \& \forall k(\varepsilon_{ik} + \gamma_{ik} \in u_k)) \rightarrow$$

$$\exists u \in U(\varphi(u) \bar{r}\varphi(\gamma_i) \& \varepsilon_i + \gamma_i \in u) \rightarrow \varphi(\varepsilon_i + \gamma_i) \bar{r}\varphi(\gamma_i).$$

Что требовалось доказать.

Пусть $E_i = \bigcup_{\varphi(u) \bar{r}\varphi(\gamma_i)} (u \div \gamma_i)$, тогда множество всех неисправностей

ε_i , обнаруживаемых последовательностью $A = \alpha_1, \alpha_2, \dots, \alpha_s$, есть $E = \bigcup_{i=1}^s E_i$. Таким образом, сложность вычислений не является экспоненциальной функцией от $n+p$, как при тривиальном переборе неисправностей, а зависит лишь от сложности представления функций φ , $\bar{\varphi}$ и последовательности A , а для практических целей эту сложность естественно считать разумной, т.к. в противном случае трудно или невозможно представить исходные данные задачи.

Данный метод вычисления неисправностей может быть использован при вычислении и (или) оценке качества тестов.

Литература

1. Хейес Дж.П. Обобщённая теория переключательных схем и её применение для проектирования СВИС. - ТИИЭР, 1982, т. 71, №1, с. 5-19.
2. Агибалов Г.П. и др. Логическое проектирование переключательных автоматов. - Томск: Изд-во Том. ун-та, 1983. - 164 с.

Н.В.Евтушенко

О ПРИНАДЛЕЖНОСТИ ПОСЛЕДОВАТЕЛЬНОСТИ МНОЖЕСТВУ КОНТРОЛЬНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ АВТОМАТА

Методы построения контрольных последовательностей для автоматов не содержат обычно доказательства того факта, что построенная данным методом последовательность действительно контрольная. Более того, большинство работ по контрольным последовательностям содержат лишь рекомендации, каким образом можно построить такую последовательность. В связи с этим возникает вопрос, как определить, является ли построенная входная последовательность контрольной для данного автомата. В настоящей работе сформулированы необходимые и достаточные условия этого свойства для входных последовательностей инициальных автоматов. В случае не инициального автомата эти условия должны выполняться для всех допустимых начальных состояний автомата.

Под автоматом далее понимается инициальный конечный автомат, т.е. шестерка $(X, Q, Y, \psi, \varphi, q_0)$. Число состояний $|Q|$ называется далее порядком автомата. Чтобы исключить тривиальные случаи, не представляющие интереса, будем полагать порядок автомата большим 1, а число выходных символов не меньшим 2. Обозначим через X^* множество слов в алфавите X , включая и пустое слово Λ ; для каждого $\alpha \in X^*$ через $\alpha|_k$ обозначим начальный отрезок слова α длины k , через $\alpha[k]$ k -ю букву в слове α , и через $|\alpha|$ — длину слова α . Определим функции $\bar{\psi}: X^* \rightarrow Q$ и $\bar{\varphi}: X^* \rightarrow Y^*$ следующим образом:

$$\bar{\psi}(\Lambda) = q_0, \quad \bar{\varphi}(\Lambda) = \Lambda,$$

и если $\alpha = \beta x$ и $\bar{\psi}(\beta)$ и $\bar{\varphi}(\beta)$ уже определены, то $\bar{\psi}(\beta x) = \bar{\psi}(\beta) \varphi(x, \bar{\psi}(\beta))$ и $\bar{\varphi}(\alpha) = \bar{\varphi}(\beta) \varphi(x, \bar{\psi}(\beta))$.

Автомат A назовем истоковым, если

$$\forall q \in Q \exists \alpha \in X^* (\bar{\psi}(\alpha) = q).$$

Пару (α, β) , $\alpha \in X^*$, $\beta \in Y^*$, назовем вход-выходной последовательностью автомата A (обозначение: $(\alpha, \beta) \in R(A)$), если $\bar{\varphi}(\alpha) = \beta$.

Будем говорить, что автоматы $A = (X, Q, Y, \psi, \varphi, q_0)$ и $B = (X, S, Y, \zeta, \varphi_0)$ изоморфны (обозначение: $A \cong B$), если существует взаимно однозначное отображение $f: Q \rightarrow S$ со свойством

$$f(q_0) = s_0 \text{ и } \forall q \in Q \forall x \in X (f(\psi(x, q)) = \zeta(x, f(q)) \wedge \varphi(x, q) = \varphi_0(x, f(q))). \quad (I)$$

Входную составляющую α вход-выходной последовательности автомата A назовем контрольной для A , если для любого автомата B , не большего, чем A порядка, и не изоморфного A , $(\alpha, \beta) \notin \mathcal{Q}(B)$, т.е. (α, β) не является вход-выходной последовательностью для автомата B .

П о с т а н о в к а з а д а ч и . Для истокового автомата A и некоторой его входной последовательности α требуется установить, является ли α контрольной последовательностью для автомата A .

Введем необходимые определения.

Пусть $(\alpha, \beta) \in X^* \times Y^*$ и $|\alpha| = |\beta| = r$. Последовательность чисел $\theta = i_1 \dots i_{r+1}$, в которой $i_1 = 1$ и $i_j \in \{1, \dots, n\}$, $j = 2, \dots, r+1$, назовем n -допустимой относительно пары (α, β) , если

$$\forall k, j \in \{1, \dots, r\} (\theta[k] = \theta[j] \wedge \alpha[k] = \alpha[j] \rightarrow \theta[k+1] = \theta[j+1] \wedge \beta[k] = \beta[j]) \quad (2)$$

Если последовательность θ n -допустима относительно (α, β) , то по тройке (α, β, θ) можно построить множество автоматов $\mathcal{Q}(\alpha, \beta, \theta)$. Множеству $\mathcal{Q}(\alpha, \beta, \theta)$ принадлежит всякий автомат $A = (X, \{1, \dots, n\}, Y, \psi, \varphi, 1)$, в котором

$$\forall k \in \{1, \dots, r\} (\psi(\alpha[k], \theta[k]) = \theta[k+1] \wedge \varphi(\alpha[k], \theta[k]) = \beta[k]) \quad (3)$$

Возможность построения таких автоматов следует из (2).

Для пары (α, β) построим дерево, вершины которого суть числа из множества $\{1, \dots, n\}$. Корнем дерева является вершина 1. Для вершины j , находящейся на i -м ярусе дерева, возможны следующие случаи.

1. Если $i = r+1$, то вершина j объявляется листом дерева.
2. Если $i \leq r$ и на пути из корня дерева в вершину j есть вершина j на k -м ярусе, $k < i$, $\alpha[k] = \alpha[i]$ и $\beta[k] \neq \beta[i]$, то вершина j i -го яруса объявляется листом дерева.
3. Если $i \leq r$ и на пути из корня дерева в вершину j есть вершина j на k -м ярусе, $k < i$, $\alpha[k] = \alpha[i]$, $\beta[k] = \beta[i]$ и j' -вершина этого пути на $(k+1)$ -м ярусе, то вершина j' помещается на $(i+1)$ -й ярус и соединяется ребром с вершиной j i -го яруса.
4. Если ни один из выше-перечисленных случаев не имеет места, то на $(i+1)$ -й ярус помещаются вершины $1, \dots, n$, каждая из которых соединяется с вершиной j i -го яруса.

Пусть теперь A - истоковый автомат и (α, β) - вход-выходная последовательность автомата A . Построим для пары (α, β) дерево описанным выше способом. Основным результатом работы является следующая

Теорема. Последовательность α является контрольной для автомата A , если и только если для каждой вершины $(r+1)$ -го яруса и последовательности вершин θ_i на пути из корня дерева в эту вершину

$$\alpha(\alpha, \beta, \theta_i) = \{ \beta \} \quad \text{и } B \text{ изоморфен } A.$$

На основании этой теоремы для решения выше-поставленной задачи составлен алгоритм обхода соответствующего дерева, который реализован программно в языке ЛЯПАС-М.

Прежде чем перейти к доказательству теоремы, докажем некоторые вспомогательные утверждения.

Утверждение 1. Если автоматы $A = (X, Q, Y, \psi, \varphi, q_0)$ и $B = (X, S, Y, \{, \bar{,} s_0)$ изоморфны и f есть соответствующее отображение изоморфизма, то

$$\forall \alpha \in X^* (f(\bar{\psi}(\alpha)) = \bar{f}(\alpha)).$$

Для доказательства утверждения воспользуемся индукцией по длине слова α .

Для $k=1$, ввиду (I), имеем

$$f(\bar{\psi}(\alpha)_1) = f(\psi(\alpha)_1, q_0) = \bar{f}(\alpha)_1, s_0) = \bar{f}(\alpha)_1.$$

Пусть теперь утверждение верно для любого α , длина которого $k \leq m$. Покажем, что утверждение верно и для слова α длины $(m+1)$. Ввиду предположения индукции и (I) имеем

$$f(\bar{\psi}(\alpha)) = f(\bar{\psi}(\alpha)_m, \alpha_{[m+1]}) = f(\psi(\alpha_{[m+1]}, \bar{\psi}(\alpha)_m) = \bar{f}(\alpha_{[m+1]}, f(\bar{\psi}(\alpha)_m)) = \bar{f}(\alpha).$$

Утверждение 1 доказано.

Пусть слово θ n -допустимо для пары (α, β) . Тогда из метода построения автоматов в $\alpha(\alpha, \beta, \theta)$ следует

Утверждение 2. Для любого автомата A из $\alpha(\alpha, \beta, \theta)$ пара (α, β) является вход-выходной последовательностью автомата A .

Непосредственно из определения n -допустимой последовательности относительно пары (α, β) следует

Утверждение 3. Для любой вход-выходной последовательности (α, β) , $|\alpha| = r$, автомата $A = (X, \{1, \dots, n\}, Y, \{, \bar{,} 1)$ последовательность θ длины $r+1$, в которой $\theta(1) = 1$, и для всех $j = 2, \dots, r+1$

$$\theta[j] = \bar{f}(\alpha)_{j-1},$$

n -допустима относительно (α, β) и $A \in \alpha(\alpha, \beta, \theta)$.

Пусть (α, β) - вход-выходная последовательность некоторого истокового автомата порядка n , и θ - n -допустимая относительно (α, β) последовательность.

Утверждение 4. Если $|\alpha(\alpha, \beta, \theta)| > 1$, то последователь-

ность α не является контрольной для автомата A .

Доказательство. Пусть $|\alpha(\alpha, \beta, \theta)| > 1$, $S = \{1, \dots, n\}$, $B = (X, S, Y, \psi_B, \varphi_B, 1) \in \alpha(\alpha, \beta, \theta)$, $C = (X, S, Y, \psi_C, \varphi_C, 1) \in \alpha(\alpha, \beta, \theta)$, $B \neq C$, $A = (X, Q, Y, \psi, \varphi, q_0)$. Если $B \neq A$ или $C \neq A$, то справедливость утверждения 4 следует из утверждения 2 и определения контрольной последовательности. Пусть поэтому $B \equiv A$ и $C \equiv A$. Т.к. $B \neq C$, то

$$\exists x \in X \exists s \in S (\psi_B(x, s) \neq \psi_C(x, s) \vee \varphi_B(x, s) \neq \varphi_C(x, s)).$$

По определению $\alpha(\alpha, \beta, \theta)$ это возможно лишь в случае

$$\forall j \in \{1, \dots, z\} ((\alpha[j], \theta[j]) \neq (x, s)).$$

Пусть $s = \bar{\psi}_B(j)$ для некоторого $j \in X^*$. Такое j всегда существует ввиду истоковости A и изоморфизма автоматов A и B . Ввиду $|Y| \geq 2$ и по определению множества $\alpha(\alpha, \beta, \theta)$ этому множеству принадлежит и автомат $D = (X, S, Y, \psi_D, \varphi_D, 1)$, в котором для всех $(x_i, s_i) \in X \times S$

$$\psi_D(x_i, s_i) = \psi_A(x_i, s_i)$$

и

$$\varphi_D(x_i, s_i) = \begin{cases} \varphi_A(x_i, s_i), & \text{если } (x_i, s_i) \neq (x, s); \\ y \neq \varphi_B(x_i, s_i), & \text{если } (x_i, s_i) = (x, s). \end{cases}$$

Из определения автомата D следует, что

$$\varphi_D(x, \bar{\psi}_D(j)) \neq \varphi_A(x, \bar{\psi}_A(j)).$$

Поэтому, ввиду утверждения 1, автомат D не изоморфен автомату B , а следовательно, и автомату A . Справедливость утверждения 4 следует теперь из утверждения 2. Утверждение 4 доказано.

Доказательство теоремы. Необходимость утверждения теоремы следует из утверждений 2 и 4 и определения контрольной последовательности.

Достаточность. Предположим противное. Пусть условия теоремы выполнены, и тем не менее существует автомат $B = (X, P, Y, \psi_B, \varphi_B, p_1)$, $|P| \leq n$, не изоморфный A , такой, что $(\alpha, \beta) \in \mathcal{Q}(B)$. Без ограничения общности будем считать, что состояния в P упорядочены и $P = \{p_1, \dots, p_k\}$. Построим автомат $C = (X, \{1, \dots, k\}, Y, \psi_C, \varphi_C, 1)$, в котором для всех $(x, i) \in X \times \{1, \dots, k\}$ справедливо $\psi_C(x, i) = j \leftrightarrow \psi_B(x, p_i) = p_j$

$$\varphi_C(x, i) = \varphi_B(x, p_i).$$

Очевидно, автоматы C и B изоморфны, и следовательно, $(\alpha, \beta) \in \mathcal{Q}(C)$. Построим последовательность θ , $|\theta| = z + 1$, в алфавите $\{1, \dots, k\}$, в которой $\theta[1] = 1$ и для всех $j \in \{2, \dots, z + 1\}$

$$\theta[j] = \bar{\psi}_C(\alpha[j-1]).$$

Согласно утверждению 3, θ $\mathcal{H}$ -допустима для (α, β) , и поэтому в дереве, построенном для пары (α, β) , есть путь длины $z + 1$, последовательность θ , вершин которого совпадает с θ . Согласно утверждению 3 $C \in \alpha(\alpha, \beta, \theta)$, что противоречит условиям теоремы. Полученное противоречие и доказывает достаточность утверждения теоремы.

Г.П. Агибалов

ЗАДАЧА ВЫБОРА И ДЕКОМПОЗИЦИЯ СХЕМ С РАЗМЕЩЕНИЕМ И ТРАССИРОВКОЙ В КОМПОНОВОЧНОМ ПРОСТРАНСТВЕ

1. Введение

Задачей выбора мы называем всякую такую задачу, в которой требуется в каждом из заданных непересекающихся множеств (классов) указать по представителю так, чтобы на системе указанных представителей достигала минимума некоторая заданная числовая характеристика и удовлетворялись некоторые заданные ограничения. Задача декомпозиции схем с размещением и трассировкой в компоновочном пространстве (задача ДРТ), которую мы рассматриваем, заключается в разбиении заданной схемы на подсхемы (блоки) и в размещении последних и связывающих их цепей в некотором конечном пространстве так, что достигает минимума сумма длин трасс, занятых этими цепями, и удовлетворяются ограничения одновременно на суммы весов элементов, цепей и полюсов в каждом блоке, на совместимость элементов в блоке, на совместимость трасс, занятых различными цепями, и на местоположение в заданном пространстве полюсов схемы. Это есть вариант общей задачи физического проектирования схем [1] — их компоновки в заданном физическом пространстве (на базовом кристалле, на печатной плате, в стойке, в шкафу, в аппаратурном помещении и т.п.).

В данной работе задача ДРТ сводится к некоторой задаче выбора и предлагается алгоритм решения произвольной задачи выбора. Последний сформулирован как алгоритм сокращенного обхода дерева поиска по глубине в пространстве систем представителей, содержащий ряд параметров, с помощью которых он может быть настроен на любую конкретную задачу выбора. В работе указываются выражения для параметров, при которых данный алгоритм решает задачу ДРТ.

2. Идеиные основы алгоритма решения задачи выбора

По своей сущности задача выбора относится к числу комбинаторных задач минимизации [2]. Наиболее общий метод их решения заключается в осуществлении направленного (не обязательно полного) перебора допустимых (удовлетворяющих заданным ограниче-

ям) решений (в нашем случае - систем представителей) с запоминанием того из них, которому отвечает меньшее значение целевой функции (характеристики системы представителей). Этот перебор осуществляется обычно с помощью так называемого дерева поиска, или решений [3,4], в котором вершины сопоставляются подзадачам рассматриваемой задачи, а пути оси корня дерева к его листьям представляют (в той или иной форме) допустимые решения последней. Количество концевых вершин дерева определяет объем пространства поиска и является мерой сложности задачи.

Имеется два альтернативных подхода к организации дерева поиска, соответствующих двум различным способам представления и построения допустимых решений. В одном случае каждое такое решение представляется набором значений булевых переменных x_{ij} и строится путем последовательного определения компонент в наборе. Например, в задаче выбора переменная x_{ij} принимает значение 1, если из i -го класса выбирается j -й представитель, и значение 0 в противном случае. Дерево поиска при таком подходе представляет собой часть дикотомического дерева высоты N , равной числу всех переменных задачи, а пространство поиска - часть N -мерного булева пространства. Подзадача, соответствующая некоторой вершине дерева, является результатом фиксирования значений некоторых переменных, приписанных ребрам пути от корня к данной вершине. Так, в задаче выбора если n - число классов и m - число элементов в классе, то $N = m \cdot n$, а пространство поиска состоит из m^n булевых векторов, каждый из которых составлен из n единичных (с одной единицей) векторов длины m . Нетрудно видеть, что в этом случае задача решения как задача математического (вообще говоря, нелинейного) 0-1 программирования [4,5].

При втором подходе к организации дерева поиска каждое допустимое решение представляется в явном виде (для задачи выбора - совокупностью представителей) и строится последовательно (элемент за элементом) [6]. В этом случае подзадача задачи выбора, соответствующая некоторой вершине i -го яруса дерева, является результатом исключения из первых ($i-1$) классов, представители которых приписаны ребрам пути от корня к данной вершине, всех элементов, кроме этих представителей. Её разбиение в свою очередь на подзадачи осуществляется путем приписывания представителей i -го класса ребрам i -го яруса дерева, исходящим из данной вершины. Таким образом, решение всей задачи

ищется непосредственно в пространстве систем представителей заданных классов.

Не вдаваясь в дискуссию о преимуществах и недостатках указанных подходов (этот вопрос заслуживает отдельного рассмотрения), при разработке алгоритма решения задачи выбора мы остановились на втором подходе к организации дерева поиска.

Любой алгоритм решения задачи, использующий дерево поиска, сводится в конечном счете к нахождению в последнем пути, представляющего решение. Поиск требуемого пути осуществляется в процессе последовательного обхода некоторой части дерева, начиная с корня. Осуществление обхода не предполагает задания дерева в явном виде; скорее наоборот, именно в его процессе как раз и происходит последовательное построение дерева. В зависимости от порядка обхода различают три основных принципа поиска [4]: по глубине, по ширине и по оценке. В первом случае переходы выполняются только между смежными вершинами, во втором — каждая вершина i -го яруса строится раньше, чем вершина $(i+1)$ -го яруса, а в третьем — продолжение обхода совершается из той вершины построенной части дерева, которой соответствует подзадача с наименьшей оценкой целевой функции. Последний принцип поиска характерен для методов ветвей и границ [2,3]. Реализация поиска по ширине или по оценке требует запоминания всего построенного фрагмента дерева, в то время как для поиска по глубине достаточно хранить лишь пути от корня к текущей вершине.

При определении принципа поиска для реализации в алгоритме решения задачи выбора это обстоятельство послужило в пользу поиска по глубине. Таким образом, алгоритм выбора строится далее как алгоритм обхода дерева поиска по глубине в пространстве систем представителей.

С целью более быстрого достижения ответа задачи в алгоритмы обхода включают механизмы сокращения области фактического поиска, не ведущие к потере решения. Один из таких механизмов основан на использовании нижней границы для целевой функции. Действие этого механизма на дереве поиска в пространстве систем представителей заключается в следующем: если $F_0 \geq F(R)$, где F_0 — нижняя граница значения целевой функции на решении подзадачи, соответствующей некоторой вершине u дерева поиска, и $F(R)$ — значение целевой функции на некотором допустимом решении R всей задачи, известном в момент, когда процесс обхода достиг вершины u , то обход поддерева с корнем в данной вершине не выполняется.

Перед началом обхода необходимое допустимое решение R строится некоторым эвристическим алгоритмом, а в процессе обхода заменяется допустимым решением с меньшим значением целевой функции всякий раз, когда такое решение становится известным.

С помощью данного механизма, как видим, решается вопрос о том, следует ли данную подзадачу разлагать на более простые подзадачи или же можно проигнорировать ее вообще. Другой механизм сокращения касается как раз множества тех подзадач, на которые следует разлагать данную задачу после того, как решение о необходимости ее разложения состоялось. При поиске в булевом пространстве это множество состоит всегда из двух подзадач, при поиске в пространстве систем представителей число подзадач в нем может достигнуть числа элементов в заданном классе. Говорят, что подзадача A мажорирует подзадачу B , если на решении последней целевая функция принимает не меньшее значение, чем на решении первой. Для некоторых задач отношение мажорирования между подзадачами определяется характером самой задачи и может быть установлено априори без обращения к решениям данных подзадач. В этом случае можно сформулировать алгоритм разложения задачи, который не порождает заведомо мажорируемые подзадачи. Обход только тех вершин дерева, которые порождаются таким алгоритмом, сокращает, очевидно, область поиска без потери решения задачи.

Следует заметить, однако, что использование механизмов сокращения обхода дерева поиска полезно не всегда; иногда оно не только не ускоряет процесс обхода, но, наоборот, ведет к его замедлению. Это происходит, в частности, всякий раз, когда обход поддерева, соответствующего некоторой подзадаче, требует меньше времени, чем реализация того или иного механизма отсекающего данное поддерево. Чтобы этого не случилось, прибегают к следующему компромиссу. Для каждого механизма сокращения указывают границы для номеров тех ярусов дерева поиска, на которых этот механизм включается. Границы подбираются опытным путем (из экспериментов на машине) так, чтобы минимизировалось среднее время решения задачи.

Используя указанные механизмы сокращения, алгоритм решения задачи выбора строится, таким образом, как алгоритм сокращенного обхода дерева поиска по глубине в пространстве систем представителей.

3. Алгоритм сокращенного обхода дерева поиска по глубине в пространстве систем представителей

Алгоритм предназначен для решения произвольной задачи выбора в следующей постановке.

Пусть задана некоторым эффективным способом конечная упорядоченная совокупность конечных непересекающихся множеств, или классов $X_1, \dots, X_n$. Набор $(x_1, \dots, x_n)$, где $x_i \in X_i$ для каждого $i = 1, \dots, n$, называется системой представителей этих классов, а любая его проекция $(x_1, \dots, x_k)$, $k \leq n$, — подсистемой таких представителей. Если каждый элемент подсистемы u входит в подсистему v , то пишут $u \in v$. Пусть имеется также некоторое эффективно проверяемое условие (предикат) P , которое может выполняться или не выполняться на подсистемах представителей данных классов так, что если подсистема u одноэлементна, то $P(u)$ истинно, и если $u \in v$ и $P(v)$ истинно, то $P(u)$ тоже истинно. Пусть, наконец, задана числовая функция F , определенная на подсистемах представителей так, что для любых таких подсистем u и v из $u \in v$ следует $F(u) \leq F(v)$. Требуется указать такую систему представителей заданных классов $X_1, \dots, X_n$, на которой выполняется условие P и достигает минимума функция F .

Зададимся некоторым алгоритмом φ , называемым далее алгоритмом перечисления, который в зависимости от представителей $x_1, \dots, x_{i-1}$ классов $X_1, \dots, X_{i-1}$ соответственно и классов $X_i, \dots, X_n$ перечисляет в классе X_i некоторое подмножество $V_i = \varphi(x_1, \dots, x_{i-1}, X_i, \dots, X_n)$ его представителей таких, что для любого $x_i \in V_i$ подсистема $(x_1, \dots, x_{i-1}, x_i)$ удовлетворяет условию P . В частности, если V_i всегда состоит из всех представителей класса X_i с указанным свойством, то алгоритм φ обозначается как $\tilde{\varphi}$. Зададимся, кроме того, некоторой операцией ψ , называемой далее операцией сокращения, которая в зависимости от $x_1, \dots, x_{i-1}, X_i, \dots, X_n$ выбирает в V_i некоторое подмножество $Y_i = \psi(x_1, \dots, x_{i-1}, X_i, \dots, X_n, V_i)$. В случае, если ψ есть тождественная операция, т.е. если всякий раз $Y_i = V_i$, она обозначается буквой I . Пусть заданы также пары натуральных чисел (a_0, a_1) и (b_0, b_1) , не превосходящих n , и для каждого $i = 1, \dots, n$ введено

$$\Phi_i(y_i) = \begin{cases} \tilde{\varphi}(y_i), & i \notin [a_0, a_1] \cup [b_0, b_1], \\ \varphi(y_i), & i \in [a_0, a_1] \setminus [b_0, b_1], \\ \varphi(y_i, \varphi(y_i)), & i \in [a_0, a_1] \cap [b_0, b_1], \\ \varphi(y_i, \tilde{\varphi}(y_i)), & i \in [b_0, b_1] \setminus [a_0, a_1], \end{cases}$$

где $Y_i = (x_1, \dots, x_{i-1}, x_i, \dots, x_n)$.

Определим дерево $\mathcal{D}_\varphi^Y$ с корнем (дерево поиска) следующим образом. Число ярусов в нем не превосходит $n+1$. Каждая вершина v i -го яруса, $1 \leq i \leq n$, ставится в соответствие набору множеств $(X_i, X_{i+1}, \dots, X_n)$. Рёбра, исходящие из неё, сопоставляются элементам множества $U_i = P_i(x_1, \dots, x_{i-1}, x_i, \dots, x_n)$, где $x_1, \dots, x_{i-1}$ — представители классов $X_1, \dots, X_{i-1}$ соответственно, сопоставленные рёбрам пути, ведущего в $\mathcal{D}_\varphi^Y$ от корня к данной вершине. Если $U_i = \emptyset$, то из вершины v не исходит ни одного ребра, и она называется концевой. Рёбра n -го яруса объявляются висячими, их концы считаются листьями дерева.

Всякий путь в дереве $\mathcal{D}_\varphi^Y$ задает (несет) подсистему, а всякий путь от корня к листу — систему представителей, сопоставленных рёбрам этого пути. Пусть R_φ^Y обозначает множество всех систем представителей, несомых в $\mathcal{D}_\varphi^Y$ путями от корня к листьям, и $\hat{R}_\varphi^Y$ — его подмножество, составленное из систем с наименьшим значением функции F . По определению алгоритма $\tilde{\varphi}$ и операции I дерево $\mathcal{D}_\varphi^Y$ является частью дерева $\mathcal{D}_{\tilde{\varphi}}^Y$, множество $R_{\tilde{\varphi}}^Y$ состоит из всевозможных систем представителей классов $X_1, \dots, X_n$, на которых выполняется условие P , и $\hat{R}_{\tilde{\varphi}}^Y$ образует множество всех решений задачи выбора. Алгоритм φ и операция ψ называются допустимыми для данной задачи, если $\hat{R}_\varphi^Y \cap \hat{R}_{\tilde{\varphi}}^Y \neq \emptyset$, т.е. если в дереве $\mathcal{D}_\varphi^Y$ имеется хотя бы один путь, несущий решение задачи. В известном смысле допустимые алгоритм φ и операция ψ выполняют функции соответственно разбиения задачи на подзадачи и исключения мажорируемых подзадач. Пары (a_0, a_1) и (b_0, b_1) задают границы для номеров тех ярусов дерева поиска, на которых применяются алгоритм и операция ψ соответственно. Вне этих границ порождение подзадач осуществляется с помощью тривиального алгоритма $\tilde{\varphi}$ и тождественной операции I .

Пусть, далее, f_0 и F_0 — неотрицательные числовые функции, называемые функциями нижней оценки, определенные на наборах $Y_i = (x_1, \dots, x_{i-1}, x_i, \dots, x_n)$, где $x_1, \dots, x_{i-1}$ — представители классов $X_1, \dots, X_{i-1}$ соответственно, так, что для любых представителей $x_i, x_{i+1}, \dots, x_n$ классов $X_i, X_{i+1}, \dots, X_n$, для которых система $x_1, \dots, x_n$ удовлетворяет условию P , имеет место

$$F(x_1, \dots, x_n) \geq F_0(Y_i) \geq f_0(Y_i) \geq 0,$$

и сложность вычисления $f_0(Y_i)$ пренебрежимо мала. Пусть, наконец, $[c_0, c_1]$ — область применения функции F_0 и

$$F_i(y_i) = \begin{cases} f_0(y_i), & i \notin [c_0, c_1], \\ F_0(y_i), & i \in [c_0, c_1]. \end{cases}$$

Обход дерева $\mathcal{D}_\varphi^*$ представляет собой последовательность операций двух типов - спуск и подъем, осуществляемых по дереву при определенных условиях. Спуск есть переход из вершины v i -го яруса в смежную вершину $(i-1)$ -го яруса для $i > 1$, а подъем есть переход из вершины v i -го яруса в вершину z $(i+1)$ -го яруса, соединенную с v не пройденным ранее ребром. Спуск выполняется в отсутствие условий для подъема. Подъем же из неконцевой вершины v i -го яруса по ребру, соответствующему элементу x_i , осуществляется при условии, если

$$F(R) > F_{i+1}(x_1, \dots, x_i, x_{i+1}, \dots, x_n),$$

где $(x_1, \dots, x_{i-1})$ - подсистема представителей, несомая в $\mathcal{D}_\varphi^*$ путем от корня к вершине v , и R - известная в момент выполнения данной операции система представителей заданных классов, удовлетворяющая условию P . Перед началом обхода в качестве R принимается система представителей R , выбираемая с помощью некоторого эвристического алгоритма. В отсутствие такового условно считается $R_0 = \emptyset$ и $F(R_0) = \infty$. Всякий раз, когда достигается лист дерева v , за R принимается $R(s_v)$ - система представителей, несомая путем s_v от корня к вершине v . Обход начинается в корне и заканчивается в листе v , в котором либо $F(R(s_v)) = F_0(x_1, \dots, x_n)$, либо не выполняются условия ни для спуска, ни для подъема. По окончании обхода система представителей R является решением задачи, если $R \neq \emptyset$; в противном случае задача не имеет решения.

Ниже приводится формальное описание данного алгоритма.

Алгоритм выбора

1. $R := R_0$, $R' := \emptyset$, $i := 1$.

2. Вычисляется множество $\{x_1^{(i)}, \dots, x_{c_i}^{(i)}\} = \Phi_i(R', x_1, \dots, x_n)$; $k_i := 0$.

3. $k_i := k_i + 1$. Если $k_i > c_i$, то п. 4; в противном случае $x_{k_i}^{(i)}$ присоединяется к R' в качестве i -го члена и $i := i + 1$.
Если $i \geq n$, то п. 5, иначе при условии

$$F(R) > F_i(R', x_1, \dots, x_n)$$

выполняется п. 2, в противном случае п. 4.

4. Если $i = 1$, то п. 6; иначе $i := i - 1$, из R' исключается i -й член и п. 3.

5. Если $F(R) \leq F(R')$, то п.4; в противном случае $R := R'$ и, если $F(R) = F_0(x_1, \dots, x_n)$, то п.6, иначе п.4.

6. Конец. Если $R \neq \emptyset$, то R есть решение, в противном случае решения не существует.

Все сказанное в [6, с.24-25] относительно возможности использования сформулированного там алгоритма разбиения для нахождения приближенного решения можно дословно повторить применительно к данному алгоритму. Фигурирующие в алгоритме объекты - допустимые алгоритм перечисления φ и операция сокращения ψ , функция нижней оценки F_0 , границы их применения (a_0, a_1) , (b_0, b_1) и (c_0, c_1) , функция нижней оценки f_0 и начальная система представителей R_0 - играют роль параметров, которые для каждой конкретной задачи выбора свои и подбираются соответственно ее условиям.

Основная цель, которая преследуется при этом подборе, заключается в обеспечении как можно более высокого быстродействия данного алгоритма в сравнении с его быстродействием при тривиальных параметрах ($\varphi = \tilde{\varphi}$, $\psi = I$, $R_0 = \emptyset$, $F_0 = f_0 = \min F(x_i)$). Это быстродействие зависит от сокращающих способностей алгоритма φ и операции ψ , измеряемых разностями $|\tilde{\varphi}(y)| - |\varphi(y)|$ и $|Z| - |\varphi(y, z)|$ соответственно, от степени точности оценок f_0 , F_0 и начальной системы представителей R_0 и от времени реализации данных параметров. Влияние последнего учитывается подбором границ для применения параметров из условия минимизации среднего времени решения задачи. В каждом конкретном случае такой подбор осуществляется экспериментально - путем решения соответствующего представительного множества примеров.

4. Декомпозиция с размещением и трассировкой в компоновочном пространстве

4.1. Постановка задачи

В интуитивном представлении компоновочное пространство - это часть реального физического пространства, разбитая посредством перегородок, или стенок, на конечное число клеток конечной вместимости так, что любые две клетки имеют не более, чем по одной общей перегородке. В простейшем случае клетки располагаются в ряд, образуя так называемое 1-мерное компоновочное пространство; в другом случае это может быть плоскость, разбитая на квадраты - плоское 2-мерное пространство и т.д. Клетки с общей перегородкой

называются далее смежными.

Формально любое компоновочное пространство представляется парой объектов $Q = (J, \sigma)$, где J - множество клеток и σ - отношение смежности между ними, и является, таким образом, графом, вершины которого суть клетки, а рёбра - перегородки, разделяющие смежные клетки. Всякое дерево t с вершинами и ребрами, принадлежащими этому графу, называется трассой компоновочного пространства. Относительно клеток и перегородок в t говорят, что они пересекаются данной трассой. Количество первых называется длиной трассы t и обозначается $\ell(t)$. Трасса, являющаяся простой цепью графа, называется простой трассой.

Под схемой далее понимается тройка объектов (X, Z, Z_0) , где X - множество элементов схемы, Z - множество ее цепей и $Z_0 \subseteq Z$ - множество ее внешних цепей, или полюсов. Элемент схемы определяется множеством своих полюсов и весом - некоторой количественной характеристикой сложности элемента. По определению множества полюсов различных элементов в схеме не пересекаются. Цепь схемы - это подмножество множества всех полюсов элементов в схеме. Она может характеризоваться дополнительно весом - значением некоторого параметра ее физического исполнения (толщины, кратности и т.п.). Предполагается, что множество всех цепей схемы образует разбиение множества всех полюсов ее элементов. Цепь схемы называется инцидентной элементу, если она содержит хотя бы один полюс этого элемента. Схема (Y, U, U_0) называется подсхемой схемы (X, Z, Z_0) , если $Y \subseteq X$, $u \in U$ тогда и только тогда, когда $u \in Z$ для некоторого $z \in Z$ и в u входят все полюсы из z , принадлежащие элементам в Y , и $u_0 \in U_0$ тогда и только тогда, когда $u_0 \notin Z$ или $u_0 \in Z_0$. По определению считается, что вес цепи и подсхемы совпадает с весом цепи $z \ni u$ схемы.

Пусть заданы схема $S = (X, Z, Z_0)$ и компоновочное пространство $Q = (J, \sigma)$. Рассмотрим произвольные отображения $\alpha: X \rightarrow J$ и $\beta: Z \rightarrow T$, где T - множество всех трасс пространства Q . Для каждой цепи $z \in Z$ определим подмножество клеток $J(\alpha, z)$, отнеся к нему всякую ту и только ту клетку $j \in J$, для которой найдется элемент $e \in X$ такой, что $\alpha(e) = j$ и цепь z инцидентна элементу e .

Пару отображений (α, β) назовем размещением схемы S в пространстве Q , если для любой цепи $z \in Z$ каждая клетка в $J(\alpha, z)$ принадлежит трассе $\beta(z)$ и каждая висячая вершина дерева $\beta(z)$ принадлежит множеству $J(\alpha, z)$; в этом случае говорят, что цепь

z реализуется трассой $\beta(z)$. Отображение α называется размещением элементов, отображение β - размещением цепей.

В этом определении $\alpha(e)$ - клетка, в которую помещается элемент e ; $J(\alpha, z)$ - множество всех тех клеток, в которых при размещении α оказываются полюсы элементов, принадлежащие цепи z ;

$\beta(z)$ - дерево, показывающее способ трассировки цепи z в заданном пространстве. Если трасса $\beta(z)$ содержит клетку j или перегородку u , то говорят, что цепь z трассируется через клетку j или перегородку u соответственно.

Обратим внимание на тот факт, что в размещении схемы размещение элементов может выбираться произвольно, в то время как размещение цепей - в зависимости от выбранного размещения элементов. В случае 1-мерного пространства размещение цепей единственно для каждого размещения элементов.

Размещение (α, β) будем характеризовать величиной $F(\alpha, \beta)$, равной сумме длин трасс, занятых при данном размещении, а именно:

$$F(\alpha, \beta) = \sum_{z \in Z} l(\beta(z)).$$

Будем называть ее стоимостью размещения (α, β) . По определению, $F(\alpha, \beta)$ - это количество пересечений клеток компоновочного пространства Q теми трассами, которыми при размещении (α, β) реализуются цепи схемы S .

Пусть заданы также числа w_j , p_j и q_j , характеризующие некоторым образом вместимость клетки $j \in J$, и отношения несовместимости на множествах элементов схемы S и трасс компоновочного пространства Q . Несовместимость элементов реальной схемы может быть обусловлена, например, их взаимными наводками. В случае плоского 2-мерного пространства под несовместимостью трасс может подразумеваться, например, их пересекательность, понимаемая в общепринятом смысле.

Задача ставится следующим образом: требуется указать размещение (α, β) схемы S в компоновочном пространстве Q с минимальной стоимостью $F(\alpha, \beta)$ так, что удовлетворяются следующие ограничения $A \div E$.

А. Элементы, помещаемые в каждую клетку, попарно совместимы.

Б. Трассы, реализующие разные цепи, совместимы.

В. Сумма весов элементов внутри клетки j не превосходит заданного числа w_j , $j \in J$.

Г. Сумма весов цепей, трассируемых через клетку j , не превосходит заданного числа p_j , $j \in J$.

Д. Сумма весов цепей, трассируемых через перегородки, ограничивающие клетку j , не превосходит заданного числа $p_j, j \in J$.

Е. Для каждой цепи $z \in Z_0$ (т.е. для каждого полюса схемы) указана некоторая клетка $a(z)$, через которую обязательно должна трассироваться данная цепь. (В реальной ситуации это может быть, например, клетка, в которой находится источник или приемник сигнала, передаваемого по цепи z , входящий в ту часть аппаратуры, которая является внешней по отношению к схеме S и которая уже распределена по компоновочному пространству. Это может быть также место, где зафиксировано положение соответствующего контакта разъема и т.п.)

Нетрудно видеть, что поставленная задача сочетает в себе как "декомпозиционный", так и "размещенческий" аспекты и состоит фактически в разбиении схем на подсхемы, в распределении последних по клеткам заданного пространства и в указании в нем трасс для связывающих эти подсхемы цепей, причем все это должно быть сделано так, чтобы достигала минимума сумма длин указанных трасс и удовлетворялись ограничения, вытекающие из ограниченности размеров клеток - на суммы весов элементов, цепей и полюсов в подсхеме (ограничения В, Г, Д), из недопустимости физического соседства некоторых элементов - на совместимость элементов в подсхеме (ограничение А), из физической неосуществимости некоторых пар трасс - на совместимость трасс различных цепей (ограничение Б) и из заданности внешней среды - на местоположение полюсов схемы (ограничение Е).

4.2. Сведение к задаче выбора

Пусть схема S состоит из n элементов $e_1, \dots, e_n$, а компоновочное пространство Q - из m клеток $1, 2, \dots, m$. Будем представлять трассы в Q при помощи m -компонентных булевых векторов с компонентами, поставленными во взаимно однозначное соответствие клеткам и принимающими значение 1 тогда и только тогда, когда представляемые трассы пересекают соответствующие клетки. Пусть Z_i обозначает множество цепей схемы S , инцидентных элементу $e_i, i = 1, \dots, n$.

Рассмотрим произвольные элемент e_i и клетку k . Каждой цепи $z \in Z_i$ поставим в соответствие некоторую простую трассу в Q с началом в клетке k . Всякое такое сопоставление назовем трассировкой для элемента e_i относительно клетки k . Оно указывает

некоторый вариант трасс для цепей, инцидентных элементу e_i , взятому в отрыве от других элементов схемы, при условии, что этот элемент помещается в клетку k . Пусть z_{ik} - число всех возможных трассировок для e_i относительно k и θ_{ike} - произвольная из таких трассировок, $\ell = 1, \dots, z_{ik}$. Представим ее булевой матрицей β_{ike} , построенной по следующим правилам. Строки и столбцы матрицы ставятся во взаимно однозначное соответствие цепям схемы S и клеткам компоновочного пространства Q соответственно. В строке, соответствующей цепи Z , записывается вектор, представляющий трассу, которая при трассировке θ_{ike} сопоставлена этой цепи.

Аналогичным образом всякому размещению (α, β) схемы S в пространстве Q можно поставить в соответствие булеву матрицу $B(\alpha, \beta)$, в которой строка, соответствующая цепи Z , представляет трассу $\beta(Z)$. Будем называть ее матрицей трасс данного размещения. Обозначая через $P(\beta)$ число единиц в булевой матрице B , нетрудно заметить, что $F(\alpha, \beta) = V(B(\alpha, \beta))$.

Пусть X_i обозначает множество всех матриц β_{ike} с фиксированным $i \in \{1, \dots, n\}$. Любая трасса компоновочного пространства является объединением некоторых простых трасс, поэтому матрица трасс любого размещения (α, β) является (покомпонентной) дизъюнкцией некоторых представителей классов $X_1, \dots, X_n$, а именно таких матриц $\beta_{1k_1\ell_1}, \dots, \beta_{nk_n\ell_n}$, для которых $k_i = \alpha(e_i)$, $i = 1, \dots, n$, и объединение простых трасс, поставленных при трассировках $\theta_{1k_1\ell_1}, \dots, \theta_{nk_n\ell_n}$ в соответствие цепи $z \in Z$, есть $\beta(z)$. Обратное утверждение в общем случае неверно, в связи с чем имеет смысл ввести следующее понятие.

Система M представителей классов $X_1, \dots, X_n$ называется правильной, если их дизъюнкция есть матрица трасс некоторого размещения (α, β) схемы S в пространстве Q ; в этом случае говорят, что система M представляет собой размещение (α, β) . По определению, система матриц $\beta_{1k_1\ell_1}, \dots, \beta_{nk_n\ell_n}$ правильна, если и только если в матрице

$$\Delta = \|\Delta_{zj}\| = \beta_{1k_1\ell_1} \vee \dots \vee \beta_{nk_n\ell_n}$$

строка, соответствующая любой цепи $z \in Z$, представляет собой трассу, которая пересекает каждую клетку множества $J(\alpha, z)$, где α определяется как $\alpha(e_i) = k_i$, $i = 1, \dots, n$, и каждая висячая вершина которой принадлежит множеству $J(\alpha, z)$; в этом случае указанная система матриц представляет размещение (α, β) , где

$\beta(z)$ для $z \in Z$ представляется строкой в $\mathcal{D}$, соответствующей z . Стоимость этого размещения равна $\nabla(\mathcal{D})$. Это размещение допустимо, т.е. удовлетворяет всем ограничениям $A + E$ задачи тогда и только тогда, когда выполнены следующие условия:

- 1) из $k_i = k_j$ следует совместимость элементов e_i и e_j ;
- 2) трассы, представляемые разными строками в $\mathcal{D}$, совместимы;
- 3) для каждого $j \in \{k_1, \dots, k_n\}$

$$\sum_{e \in M_j} w(e) \leq w_j,$$

где $w(e)$ - вес элемента e и $e_i \in M_j \leftrightarrow k_i = j$;

- 4) для каждого $j = 1, \dots, m$

$$\sum_{z \in Z} d_{zj} \cdot q(z) \leq q_j,$$

где $q(z)$ - вес цепи z ;

- 5) для каждого $j = 1, \dots, m$

$$\sum_{z \in Z} \sum_{k \in \sigma(j)} d_{zj} \cdot d_{zk} \cdot q(z) \leq p_j,$$

где $\sigma(j)$ - множество клеток, смежных с клеткой j ;

- 6) $d_{za(z)} = 1$ для каждой цепи $z \in Z_0$.

Таким образом, задача декомпозиции схемы S с размещением и трассировкой в компоновочном пространстве Q сводится к следующей задаче выбора, именуемой далее задачей о дизъюнктивно-минимальной системе представителей, или ДМ-задачей: требуется указать правильную систему таких представителей $v_{ik, e_i}, \dots, v_{nk, e_n}$ классов $X_1, \dots, X_n$ соответственно, которые удовлетворяют условиям 1-6 и в дизъюнкции которых число единиц минимально.

Заметим, что здесь под X_i не обязательно подразумевать множество всех матриц v_{ike} с фиксированным $i \in \{1, \dots, n\}$. Можно считать, например, что X_i состоит только из тех таких матриц, которые представляют так называемые допустимые трассировки θ_{ike} , где трассировка θ_{ike} называется допустимой, если совокупность трасс, которые при этой трассировке ставятся в соответствие цепям множества Z_i , удовлетворяет всем ограничениям Б, Г, Д и Е, а именно: трассы, соответствующие разным цепям, совместимы, сумма весов цепей, поставленных в соответствие трассам, пересекающим некоторую клетку j или ограничивающие ее

перегородки, не превосходит заданных чисел q_i или p_j соответственно и трасса, соответствующая цепи $z \in Z_0$, оканчивается клеткой $a(z)$. В этом случае условие 6 автоматически выполняется и в постановке задачи требование удовлетворения ему со стороны искомой системы представителей может быть снято.

Впервые ДМ-задача как задача выбора системы булевых матриц с минимальным числом единиц в их дизъюнкции была поставлена и решена автором в работе [7] в связи с декомпозицией и размещением схем в одномерном пространстве по минимуму суммарной длины межблочных связей при одном ограничении - Е. В [6] ее постановка и решение даны в связи с декомпозицией и размещением схем в произвольном пространстве и при двух ограничениях - В и Е. В обоих случаях алгоритм решения строится как алгоритм сокращенного обхода дерева поиска в пространстве систем представителей, т.е. как сформулированный в п.3 алгоритм выбора с подходящими параметрами - алгоритмом перечисления φ , операцией сокращения ψ , функцией нижней оценки F_0 и начальной системой представителей R_0 . Ниже такие параметры формулируются для алгоритма выбора, решающего ДМ-задачу в общем случае.

4.3. Параметры алгоритма выбора

Система представителей $B_{1,k_1,l_1}, \dots, B_{n,k_n,l_n}$ классов $X_1, \dots, X_n$ называется допустимой, если она удовлетворяет всем условиям 1-6. Подсистемы правильных или допустимых систем представителей называются соответственно правильными или допустимыми подсистемами. Простая трасса, соединяющая некоторую пару клеток, называется минимальной, если не существует другой простой трассы, соединяющей эти клетки, каждая клетка которой принадлежит данной трассе. Матрица $B_{j,k,l} \in X_i \cup \dots \cup X_n$ называется совместимой с правильной подсистемой матриц $(B_{1,k_1,l_1}, \dots, B_{i-1,k_{i-1},l_{i-1}})$, если последние в совокупности с $B_{j,k,l}$ образуют допустимую подсистему представителей своих классов и для каждой цепи $z \in Z_j$ трасса, сопоставленная ей при трассировке $\theta_{j,k,l}$, в случае $z \notin Z_0 \cup Z_1 \cup \dots \cup Z_{i-1}$ состоит из единственной клетки k_j , а в противном случае является минимальной простой трассой, соединяющей клетку k_j с клеткой, принадлежащей трассе, представленной соответствующей цепи Z строкой матрицы

$$\theta_{i-1} = B_0 \vee B_{1,k_1,l_1} \vee \dots \vee B_{i-1,k_{i-1},l_{i-1}},$$

где в булевой матрице $B_0 = \|b_{z,j}\|$ элемент $b_{z,j} = 1 \leftrightarrow j \in a(z)$,

и не содержит других клеток этой трассы.

Алгоритм φ определяется следующим образом: в перечисляемое им множество $\varphi(v_{ik_1l_1}, \dots, v_{i-1k_{i-1}l_{i-1}}, x_i, \dots, x_n)$ входит всякая такая и только такая матрица $v_{ik_1l_1} \in X_i$, которая совместима с подсистемой $(v_{1k_1l_1}, \dots, v_{i-1k_{i-1}l_{i-1}})$. Считается, что алгоритм φ применяется на всех ярусах дерева поиска, т.е. для него в алгоритме выбора $a_0 = 1$ и $a_1 = n$.

Для определенного так алгоритма φ все системы представителей, несомые в дереве $\mathcal{D}_\varphi$ путями от корня к листьям, правильны и допустимы, и любая правильная и допустимая система представителей классов $X_1, \dots, X_n$ либо задается некоторым из этих путей, либо получается из некоторой такой системы перераспределением единиц между ее матрицами без изменения их дизъюнкции или добавлением единиц в некоторые из них. Отсюда следует допустимость алгоритма φ .

Операция ψ определяется как тождественная.

Для булевых матриц $A = \|a_{ij}\|$, $B = \|b_{ij}\|$ и $C = \|c_{ij}\|$ одинакового размера будем говорить, что A получена вычитанием C из B и писать $A = B - C$, если $a_{ij} = 1 \leftrightarrow (b_{ij} = 1 \wedge c_{ij} = 0)$. Пусть далее $\bar{X}_j$ для $j \geq i$ обозначает класс матриц, которые получены вычитанием матрицы $\mathcal{D}_{i-1}$ из матриц класса X_j , совместимых с подсистемой $(v_{1k_1l_1}, \dots, v_{i-1k_{i-1}l_{i-1}})$. Число единиц в булевой матрице называется весом этой матрицы.

Значение функции нижней оценки $f_0(v_{1k_1l_1}, \dots, v_{i-1k_{i-1}l_{i-1}}, x_i, \dots, x_n)$ определяется как сумма $f_i + \varphi(\mathcal{D}_{i-1})$, где f_i — наибольший вес матрицы среди матриц наименьшего веса в классах $\bar{X}_i, \dots, \bar{X}_n$:

$$f_i = \max_{j \geq i} \min_{B \in \bar{X}_j} \varphi(B).$$

Начальная система представителей $R_0 = (v_{1k_1l_1}, \dots, v_{nk_nl_n})$ строится путем последовательного выбора допустимых представителей по минимуму приращения стоимости, а именно: в ней для каждого $i = 1, \dots, n$ матрица $v_{ik_1l_1}$ выбирается из множества $V_i = \varphi(v_{1k_1l_1}, \dots, v_{i-1k_{i-1}l_{i-1}}, x_i, \dots, x_n)$ так, что для любой другой матрицы $v_{ike} \in V_i$, $ke \neq k_1l_1$, имеет место

$$\varphi(\mathcal{D}_{i-1} \vee v_{ik_1l_1}) \leq \varphi(\mathcal{D}_{i-1} \vee v_{ike}).$$

В случае $V_i = \emptyset$ для некоторого $i \in \{1, \dots, n\}$ считается $R_0 = \emptyset$.

4.4. Задача с одним ограничением

Рассмотрим задачу декомпозиции схем с размещением и трассировкой в отсутствие всех ограничений, кроме одного - Е. В этом случае удастся ввести более эффективные операцию сокращения φ , функцию нижней оценки F_0 и алгоритм выбора начальной системы k_0 .

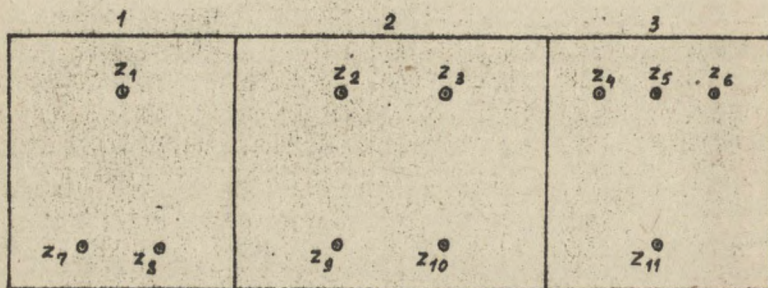
Условимся далее единицу в строке a и столбце b булевой матрицы называть ab -единицей. Пусть далее X_i обозначает множество только тех матриц b_{ikl} с фиксированным $i \in \{1, \dots, n\}$, которые представляют трассировки θ_{ikl} , сопоставляющие каждой цепи $z \in Z_0 \cap Z_i$ минимальные простые трассы, соединяющие клетку k с клеткой $a(z)$. Рассмотрим произвольный класс $\bar{X}_k, k \geq i$, и некоторую матрицу $B \in \bar{X}_k$. Всякую ab -единицу в B назовем фиктивной, если для некоторого $j \geq i, j \neq k$, каждая матрица класса $\bar{X}_j$ имеет ab -единицу; ab -единица в B называется существенной, если для каждого $j \geq i, j \neq k$, в классе $\bar{X}_j$ нет матрицы с ab -единицей; ab -единица разности матриц называется существенной, если она существенна в уменьшаемом; матрица B называется мажорируемой, если в классе $\bar{X}_k$ имеется другая матрица C такая, что число существенных единиц в $B-C$ не меньше числа всех единиц в $C-B$.

Введем следующие элементарные преобразования множества классов $\bar{X}_i, \dots, \bar{X}_n$.

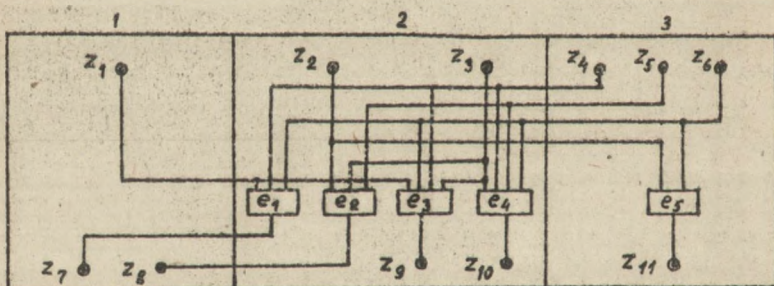
1. Исключение из некоторого класса некоторой мажорируемой матрицы.
2. Уничтожение (замена нулями) всех фиктивных единиц во всех матрицах некоторого класса.

Будем последовательно применять указанные преобразования классов $\bar{X}_i, \dots, \bar{X}_n$ до тех пор, пока можно, т.е. пока не останется в них ни мажорируемых матриц, ни матриц с фиктивными единицами. Полученные в конечном итоге классы обозначим через $X'_i, \dots, X'_n$. Их конкретный вид зависит от того, в каком порядке, к каким конкретно матрицам и в каких классах применялись элементарные преобразования, так как уничтожение фиктивных единиц может привести к появлению отсутствовавших ранее мажорируемых матриц, а исключение мажорируемых матриц, в свою очередь, может повлечь появление новых фиктивных единиц.

Полагая, что операция вычитания из матриц и их элементарное преобразование 2, изменяя значения матриц, сохраняет за ними их



a



b

Рис. 1

обозначения, определим операцию сокращения ψ как операцию исключения из результата работы алгоритма φ всякой матрицы $b_{ik\ell}$, для которой в x'_i нет матрицы с тем же символом, а функцию нижней оценки F_0 и начальную систему представителей R_0 так же, как и в общем случае, подставив в их определения вместо классов

$x_i, \dots, x_n$ классы $x'_i, \dots, x'_n$ соответственно и исключив проверку на допустимость подсистемы $(b_{1k_1\ell_1}, \dots, b_{i-1k_{i-1}\ell_{i-1}})$. Допустимость ψ и справедливость F_0 в данном случае следует из того легко проверяемого факта, что элементарные преобразования классов $\bar{x}_i, \dots, \bar{x}_n$ не изменяют минимального веса системы представителей классов $x_1, \dots, x_{i-1}, \bar{x}_i, \dots, \bar{x}_n$, в которой представители первых $(i-1)$ классов зафиксированы и равны $b_{1k_1\ell_1}, \dots, b_{i-1k_{i-1}\ell_{i-1}}$ соответственно.

Заметим, что указанное выше упрощение классов матриц можно вводить в определение не только параметров, сокращающих обход дерева поиска, но и множеств классов, сопоставляемых вершинам последнего. В тех случаях, когда возможно явное задание классов, это ускоряет реализацию параметров и обход дерева в целом. Кроме того, в этих случаях возможна реализация механизма исключения повторяющихся вершин (подзадач) подобно тому, как это делается в алгоритме разбиения.

4.5. Пример

Пусть требуется разместить в линейке из трех одинаковых клеток 1, 2, 3 (рис. 1-а) схему S , состоящую из пяти элементов $e_1, \dots, e_5$ и одиннадцати цепей $z_1, \dots, z_{11}$, являющихся одновременно полюсами схемы. Разные полюсы одного и того же элемента принадлежат разным цепям: полюсы элемента e_1 - цепям z_1, z_4, z_6 и z_7 , полюсы элемента e_2 - цепям z_2, z_3, z_5 и z_8 , полюсы элемента e_3 - цепям z_1, z_3, z_4, z_6 и z_9 , полюсы элемента e_4 - цепям z_3, z_4, z_5, z_6 и z_{10} и полюсы элемента e_5 - цепям z_2, z_6 и z_{11} . В клетке 1 зафиксировано местоположение контактов разъема для цепей z_1, z_7 и z_8 , в клетке 2 - для цепей z_2, z_3, z_9 и z_{10} и в клетке 3 - для цепей z_4, z_5, z_6 и z_{11} .

Построим классы матриц $x_1, \dots, x_5$ (таблица). В классе x_i , $i=1, \dots, 5$, перечислены матрицы b_{ij} , представляющие собой всевозможные трассировки для элемента e_i относительно клеток $j=1, 2, 3$. Для упрощения записи в матрицах опущены пустые (состоящие из одних нулей) строки.

$$x_1:$$

β_{11}	1	2	3
z_1	1	0	0
z_4	1	1	1
z_6	1	1	1
z_7	1	0	0

β_{12}	1	2	3
z_1	1	1	0
z_4	0	1	1
z_6	0	1	1
z_7	1	1	0

β_{13}	1	2	3
z_1	1	1	1
z_4	0	0	1
z_6	0	0	1
z_7	1	1	1

$$x_2:$$

β_{21}	1	2	3
z_2	1	1	0
z_3	1	1	0
z_5	1	1	1
z_8	1	0	0

β_{22}	1	2	3
z_2	0	1	0
z_3	0	1	0
z_5	0	1	1
z_8	1	1	0

β_{23}	1	2	3
z_2	0	1	1
z_3	0	1	1
z_5	0	0	1
z_8	1	1	1

$$x_3:$$

β_{31}	1	2	3
z_1	1	0	0
z_3	1	1	0
z_4	1	1	1
z_6	1	1	1
z_9	1	1	0

β_{32}	1	2	3
z_1	1	1	0
z_3	0	1	0
z_4	0	1	1
z_6	0	1	1
z_9	0	1	0

β_{33}	1	2	3
z_1	1	1	1
z_3	0	1	1
z_4	0	0	1
z_6	0	0	1
z_9	0	1	1

$$x_4:$$

β_{41}	1	2	3
z_3	1	1	0
z_4	1	1	1
z_5	1	1	1
z_6	1	1	1
z_{10}	1	1	0

β_{42}	1	2	3
z_3	0	1	0
z_4	0	1	1
z_5	0	1	1
z_6	0	1	1
z_{10}	0	1	0

β_{43}	1	2	3
z_3	0	1	1
z_4	0	0	1
z_5	0	0	1
z_6	0	0	1
z_{10}	0	1	1

$$x_5:$$

β_{51}	1	2	3
z_2	1	1	0
z_6	1	1	1
z_{11}	1	1	1

β_{52}	1	2	3
z_2	0	1	0
z_6	0	1	1
z_{11}	0	1	1

β_{53}	1	2	3
z_2	0	1	1
z_6	0	0	1
z_{11}	0	0	1

В разности матриц $\delta_{13} - \delta_{22}$ одна существенная единица - $(z_3, 3)$, вес матрицы $\delta_{22} - \delta_{32}$ равен 1, поэтому матрица δ_{23} мажорируема. Аналогичным образом можно убедиться в мажорируемости матриц δ_{13} , δ_{31} , δ_{41} , δ_{51} и δ_{52} . После исключения матрицы δ_{41} единицы с координатами $(z_3, 1)$ и $(z_5, 1)$ становятся существенными в матрицах δ_{21} и $\delta_{21} - \delta_{22}$ и, так как вес матрицы $\delta_{22} - \delta_{21}$ равен 1, то матрица δ_{21} становится мажорируемой. То же самое можно сказать и про матрицу δ_{11} . Таким образом, с помощью элементарного преобразования I из классов $x_1, \dots, x_5$ исключаются матрицы δ_{23} , δ_{13} , δ_{31} , δ_{41} , δ_{51} , δ_{52} , δ_{21} и δ_{11} . После этого $(z_5, 2)$ - единица матрицы δ_{42} становится фиктивной, так как в классе x_2 единственная оставшаяся матрица имеет $(z_5, 2)$ - единицу. Подобным же образом фиктивными становятся единицы с координатами $(z_1, 2)$, $(z_4, 2)$ и $(z_4, 6)$ в матрице δ_{32} , $(z_1, 2)$ - единица в матрице δ_{33} и единицы с координатами $(z_4, 2)$ и $(z_6, 2)$ в матрице δ_{42} . После их уничтожения обнаруживается мажорируемость матриц δ_{33} и δ_{43} . С исключением этих матриц дальнейшее применение элементарных преобразований становится невозможным. Итогом является множество классов $x'_1 = \{\delta_{12}\}$, $x'_2 = \{\delta_{22}\}$, $x'_3 = \{\delta_{32}\}$, $x'_4 = \{\delta_{42}\}$ и $x'_5 = \{\delta_{53}\}$. Его единственная система представителей $(\delta_{12}, \delta_{22}, \delta_{32}, \delta_{42}, \delta_{53})$ является решением дм-задачи для рассматриваемого примера. Представляемое ею размещение схемы показано на рис. 1, б. Стоимость этого размещения минимальна и равна 18.

Л и т е р а т у р а

1. Фридман А., Менон П. Теория и проектирование переключательных схем. - М.: Мир, 1978. - 580 с.
2. Кофман А. Введение в прикладную комбинаторику. - М.: Наука, 1975. - 480 с.
3. Рейкгольд Э., Нивельгельт Ю., Део Н. Комбинаторные алгоритмы. Теория и практика. - М.: Мир, 1980. - 476 с.
4. Кристофидес Н. Теория графов. Алгоритмический подход. - М.: Мир, 1978. - 432 с.
5. Geoffrion A.M., Marsten R.E. Integer programming: a frame-work and state-of-the-art survey. - *Management Science*, 1972, v.18, N9.
6. Агibalов Г.П., Беляев В.А. Технология решения комбинаторно-логических задач. - Томск: Изд-во Томского ун-та, 1981. - 126 с.
7. Агibalов Г.П. Об одном подходе к размещению аппаратуры. - Автоматика и вычислительная техника, 1971, № 1, с.56-61.

Н.А.Елисеева

АЛГОРИТМ ТРАССИРОВКИ ЭЛЕКТРИЧЕСКИХ ЦЕПЕЙ

Предлагается алгоритм нахождения трассы для заданной электрической цепи на поле печатной платы или полупроводникового кристалла. Область платы или кристалла, разрешённая для трассировки, задаётся множеством прямоугольников. Кроме того, заданы несвязные между собой фрагменты произвольной формы. Им соответствуют выводы элементов или проложенные ранее проводники рассматриваемой цепи. Требуется найти минимальное по мощности множество прямоугольников, связывающее данные фрагменты, удаление из которого любого прямоугольника нарушает условие связности фрагментов.

Пусть S – множество прямоугольников, ρ – бинарное отношение пересечения между ними и $G = (S, \rho)$ – граф этого отношения. Предполагается, что граф G связан. Пусть $G_1, G_2, \dots, G_k$ ($k \geq 2$) – связанные подграфы графа G , такие, что $G_i \cap G_j = \emptyset$; $i \neq j$; $i, j = \overline{1, k}$. Семейство подграфов $G_1, G_2, \dots, G_k$ в дальнейшем, где это удобно, будем обозначать через M . Подграф $H \subset G$ называется связывающим подграфом для $G_1, G_2, \dots, G_k$, если подграф $H \cup G_1 \cup G_2 \cup \dots \cup G_k$ связан. Связывающий подграф с минимальным числом вершин называется минимальным связывающим подграфом.

Решаемая в работе задача ставится следующим образом. Для заданных подграфов $G_1, G_2, \dots, G_k$ в графе G найти минимальный связывающий подграф H .

Для любой пары подграфов G_i и G_j обозначим через $P(i, j)$ множество всех простых цепей в графе $G - \bigcup_{i \neq j} G_i$, связывающих подграфы G_i и G_j и не содержащих вершины других подграфов заданного семейства M . Поставим в соответствие заданному семейству M граф $K = (V, U)$ с множеством вершин $V = \{1, 2, \dots, k\}$ и множеством ребер U – таким, что ребро $(i, j) \in U - P(i, j) \neq \emptyset$. Каждому ребру $(i, j) \in U$ поставим в соответствие множество $P(i, j)$. В дальнейшем, если ребро обозначается одним символом, например u , соответствующее ему множество будем обозначать через $P(u)$. Одним из подходов к решению сформулированной задачи может быть следующий. Для графа K построим всевозможные связывающие деревья (остовы) $K_1, K_2, \dots, K_r$. Рассмотрим некоторый остов $K_s(V, U^s)$ с множеством ребер $U^s = \{u_1, u_2, \dots, u_{k-1}\}$ и с соответствующими мно-

жествами $p(u_1), p(u_2), \dots, p(u_{k-1})$. Набор $P = p_1, p_2, \dots, p_{k-1}$ цепей, выбранных по одной из каждого множества $p(u_i), i = \overline{1, k-1}$, назовём системой представителей для K_g . Нетрудно видеть, что $P = p_1 \cup p_2 \cup \dots \cup p_{k-1}$ есть некоторый связывающий подграф для $G_1, G_2, \dots, G_k$ и для выбранного остова K_g . Перебрав все возможные системы представителей для $K_1, K_2, \dots, K_g$, найдём систему с минимальным числом вершин. Ей соответствует минимальный связывающий подграф H . Нетрудно оценить сложность такого перебора

$$N = \sum_{i=1}^k |p(u_1^i)| \cdot |p(u_2^i)| \cdot \dots \cdot |p(u_{k-1}^i)|.$$

Таким образом, задача нахождения минимального связывающего подграфа H распадается на три самостоятельные задачи: 1) построение графа K ; 2) перечисление всех остовов графа K ; 3) выбор минимальной по числу вершин системы представителей для остова K_g .

Очевидно, что сложность перебора систем представителей быстро растёт с ростом числа вершин и ребер в графе G и числа связывающих подграфов $G_1, G_2, \dots, G_k$, так как это приводит к резкому увеличению числа остовов K_g и числа простых цепей, соединяющих все пары подграфов G_i и G_j в графе G . Одним из способов уменьшения числа N возможных систем представителей является снижение мощности всех множеств $p(u_i)$ за счёт исключения из него тех цепей, которые не войдут в минимальную систему представителей.

Рассмотрим алгоритм построения множества простых цепей, связывающих все пары подграфов G_i и G_j в графе G и сформулируем некоторые достаточные условия исключения цепей из $p(u_i)$. Для изложения алгоритма введём необходимые понятия и определения. Обозначим через $O_i(G_i)$ — I-приемники вершин S_i графа G_i в графе $G - G_i$. Для каждого $G_i \in M$ введём подмножество $D_p(S_i)$ вершин графа G следующим образом:

$$D_0(S_i) = S_i, \quad D_1(S_i) = O_1(S_i), \quad (I)$$

$$D_p(S_i) = O_i(D_{p-1}(S_i)) - \bigcup_{j=1}^k D_{p-2}(S_j), \quad p = 2, 3, \dots$$

В дальнейшем вершины $D_p(S_i)$ будем называть p -приемниками вершин S_i . Пусть τ_i и τ_j натуральные числа такие, что $D_{\tau_i}(S_i) \cap D_{\tau_j}(S_j) = P_{\tau_i, \tau_j} \neq \emptyset$. Из правила (I) построения $D_{\tau_i}(S_i)$ и $D_{\tau_j}(S_j)$ следует, что: 1) $|\tau_i - \tau_j| \leq 1$; 2) наличие $P_{p, p-1} \neq \emptyset$ означает наличие $P_{p, p-1} \neq \emptyset$, причём $P_{p, p-1} = P_{p-1, p}$. Поэтому

для определенности будем считать $\tau_i \geq \tau_j$ и обозначать $\tau_i = p$, $\tau_j = a$. Обозначим через $P_m(i, j)$ все простые цепи длины m , соединяющие вершины S_i и S_j в графе G . Очевидно, $P_m(i, j)$ можно определить как последовательность $m = p + a - 1$ компонент $p_1^m, p_2^m, \dots, p_m^m$, где $p_\ell^m \cap p_\tau^m = \emptyset$, $\ell = \tau$; $\ell, \tau = \overline{1, m}$. Каждая компонента p_ℓ^m , $\ell = \overline{1, m}$ есть непустое подмножество вершин графа G , таких и только таких, которые одновременно являются ℓ - приемниками вершин S_i и $(m - \ell + 1)$ - приемниками вершин S_j . Тогда простая цепь $p_\ell \in P_m(i, j)$, соединяющая подграфы G_i и G_j есть последовательность m вершин графа G , взятых по одной из каждой компоненты p_ℓ^m , $\ell = \overline{1, m}$, без нарушения условия связности вершин.

Будем говорить, что множество простых цепей $P_m(i, j)$ поглощает множество простых цепей $P_n(i, j)$ длины $n \geq m$, если для каждой компоненты p_ℓ^m , $\ell = \overline{1, m}$, найдется компонента p_τ^n , такая что $p_\ell^m \supseteq p_\tau^n$. Очевидно, поглощаемое множество $P_n(i, j)$ можем исключить из рассмотрения, так как включение в систему представителей любой цепи $p_\ell \in P_n(i, j)$ добавит те же вершины, что и поглощающая цепь $p_\ell \in P_m(i, j)$ и те вершины, которые не принадлежат ни одной другой цепи, включенной в систему представителей.

Будем говорить, что множество $P_n(i, j)$ простых цепей, соединяющих вершины S_i и S_j в графе G , является дополнением множеств $P_{n_1}(i, \ell)$ и $P_{n_2}(j, \ell)$, соединяющих вершины S_i и S_ℓ , S_j и S_ℓ , соответственно, если $P_n(i, j)$ поглощается множеством цепей $P_m(i, j) = P_{n_1}(i, \ell) \cup P_{n_2}(j, \ell) \setminus P_{n_1}(i, \ell) \cap P_{n_2}(j, \ell)$ и $m = n$. Это означает, что множество цепей $P_{n_1}(i, \ell) \cup P_{n_2}(j, \ell)$ связывающих подмножества вершин S_i , S_j и S_ℓ поглощает множества цепей $P_n(i, j) \cup P_{n_1}(i, \ell)$ и $P_n(i, j) \cup P_{n_2}(j, \ell)$. Следовательно $P_n(i, j)$ можем исключить из рассмотрения.

Множество простых цепей $P_{m+1}^z(i, j) = p_1^{m+1}, p_2^{m+1}, \dots, p_{\tau-1}^{m+1}, p_{\tau}^{m+1}, p_{\tau+1}^{m+1}, \dots, p_m^{m+1}$ назовем τ - расширением множества $P_m(i, j)$ по компоненте p_τ^m относительно вершин S_i , если $P_{m+1}^z(i, j)$ получено из $P_m(i, j)$.

добавлением компоненты P_2^{m+1} , содержащей вершины, которые являются z - приемниками вершин S_i и 1 - приемниками вершин P_2^m , причем $P_2^{m+1} \cap (P_2^m \cup P_{2+1}^m) = \emptyset$. Компоненты P_2^{m+1} , $l=1, 2-1$ есть не пустые подмножества вершин, являющихся l - приемниками вершин S_i и $(z-l)$ - приемниками вершин P_2^{m+1} .

Аналогично вводится z - расширение множества $P_m(i, j)$ по компоненте P_2^m относительно вершин S_j :

$$P_{m+1}^z(i, j) = P_1^m, P_2^m, \dots, P_z^m, P_{z+1}^{m+1}, P_{z+2}^{m+1}, \dots, P_{m+1}^{m+1} = \\ = P_1^{m+1}, P_2^{m+1}, \dots, P_{m+1}^{m+1}.$$

Здесь P_{z+1}^{m+1} есть z - приемники вершин S_j и 1 - приемники вершин P_z^m , причем $P_{z+1}^{m+1} \cap (P_z^m \cup P_{z+1}^m) = \emptyset$.

Компонента P_2^{m+1} , $l = z+2, m+1$, есть $(l-z)$ -приемники вершин P_2^m и $(m-l-2)$ - приемники вершин S_j и

$$P_2^{m+1} \cap P_z^{m+1} = \emptyset \text{ для } l+t; l, t = 1, m+1.$$

Таким образом, в множество простых цепей $P(i, j)$, сопоставленное ребру (i, j) графа K , включаем взаимнонепоглощаемые множества простых цепей $P_m(i, j)$ различной длины и непоглощаемые расширения поглощаемых множеств. Легко показать, что часть поглощаемых цепей и дополняющие цепи можно исключить из рассмотрения в процессе параллельного вычисления p - приемников вершин $S_1, S_2, \dots, S_k$:

а) удаляя $(p-2)$ - приемники вершин $\bigcup S_i$; б) для S_i и S_j при очередном $P_{p,a} \neq \emptyset$, удаляя вершины $P_{p,a}$ из p - приемников вершин S_i и, при $a = p-1$, из p - приемников вершин S_j удаляя $(p-1)$ - приемники вершин S_i .

Алгоритм построения графа $K(V, U)$ формулируется следующим образом.

Имеем: граф $G(S, p)$, семейство подграфов $M = G_1, G_2, \dots, G_k$;

$$G_i = (S_i, p), \quad i = 1, k;$$

$$K = (V, U_n); \quad V = 1, 2, \dots, k; \quad U_n = \emptyset.$$

Положим $p = 0$.

I. $p := p+1$. Вычисляем p - приемники $D_p(S_1), D_p(S_2), \dots, D_p(S_k)$. Если $\bigcup_{i=1}^k D_p(S_i) = \emptyset$ пе-

переходим в 8, иначе, если найдется j , такое, что

$$D_p(S_j) \neq \emptyset, \text{ а } \bigcup_{i=1, i \neq j}^k D_p(S_i) = \emptyset \text{ и } \bigcup_{i=1, i \neq j}^k D_{p-1}(S_i) = \emptyset,$$

то переходим в 8, иначе принимаем $i=j=1$, $a=p-1$, переходим в 2.

2. $j:=j+1$. Если $j > k$, переходим в 5, в противном случае находим пересечение $P_{p,a} = D_p(S_i) \cap D_p(S_j)$. Если $P_{p,a} \neq \emptyset$, переходим в 3, иначе снова выполняем 2.

3. Строим множество простых цепей длины $m = p + a - 1$. Если $P(i,j) = \emptyset$, переходим в 4, иначе проверяем на поглощение множество $P_m(i,j)$ с элементами множества $P(i,j)$, сопоставленного ребру (i,j) графа K . Если поглощение есть, переходим в 7, иначе выполняем 4.

4. Множество простых цепей $P_m(i,j)$ включаем в $P(i,j)$, переходим в 6.

5. $i:=i+1, j:=i$, если $i > k$, переходим в 2, в противном случае, если $p=a$, переходим в 1, иначе, принимаем $a=p, i=j=1$, переходим в 2.

6. $D_p(S_i) := D_p(S_i) \setminus P_{p,a}$, если $a=p$, переходим в 2, иначе $D_p(S_j) := D_p(S_j) \setminus D_{p-1}(S_i) \cap D_p(S_j)$, переходим в 2.

7. Находим непоглощаемое τ - расширение множества $P_n(i,j)$. Если такое расширение есть, принимаем $m := m+1$, $P_m(i,j) := P_m^{\tau}(i,j)$, переходим в 4, иначе в 6.

8. Граф K построен.

Для решения задачи 2 в задаче нахождения минимального связывающего подграфа H , имеются довольно эффективные методы перечисления всех остовов без повторения, начиная с некоторого начального $[I]$.

В качестве начального остова K_0 в графе K выберем остов, содержащий ребра, которым приписаны самые короткие цепи $P_1^0, P_2^0, \dots, P_{k-1}^0$ в графе G . Число вершин соответствующего связывающего графа

$$H_0(S^0, f) = P_1^0 \cup P_2^0 \cup \dots \cup P_{k-1}^0$$

есть $|S^0| \leq \sum_{i=1}^{k-1} m_i^0 = N$,

где m_i^0 - длина цепи P_i^0 , $i = 1, k-1$. Число вершин $|S^0| = N$, если все цепи изолированы друг от друга (не имеют общих вершин).

Пусть цепи $P_1^0, P_2^0, \dots, P_{k-1}^0$, упорядочены по возрастанию их длины. Граф H_0 задает p -разбиение множества M подграфов $G_1, G_2, \dots, G_k$ на непересекающиеся классы $M_1^p, M_2^p, \dots, M_n^p, n \leq k$, такие, что два графа G_i и $G_j \in M_t^p$, если и только если имеется простая цепь $P_m(i, j)$ длины $m \leq 2p-1$, соединяющая вершины S_i и S_j .

Найдем p -разбиение множества M на классы при $p = 0, 1, \dots, p_{k-1}$. Очевидно, суммарное число классов есть $N_0 = (p_1 - 1)k + n_0 + n_1 + \dots + n_{\Delta p} \leq |S^0|$,

где n_i - есть число классов в разбиении при $p = p_1 + i$, $\Delta p = p_{k-1} - p_1$, $p_1 = m_1^0 - a_1^0 + 1$, $p_{k-1} = m_{k-1}^0 - a_{k-1}^0 + 1$.

Если окажется, что $|S^0| = N_0$, то $P_1^0, P_2^0, \dots, P_{k-1}^0$ и есть искомая система представителей. В противном случае минимальный связывающий подграф H будем искать среди подграфов H_θ с числом вершин $|S^3|$, удовлетворяющим условию

$$(p_1 - 1)k + \sum_{i=1}^{\Delta p} n_i^0 < |S^3| \leq |S^0|.$$

Одним из подходов к решению задачи 3 выбора системы представителей является сведение ее к ΔM -задаче [2] следующим образом. Для текущего остова K_θ введем систему булевых матриц $\alpha_1, \alpha_2, \dots, \alpha_{k-1}$. Столбцом матрицы α_i , $i = 1, k-1$, сопоставим вершины графа G , а строкам простые цепи из $P(\alpha_i)$. Тогда объединение строк, взятых по одной из каждой матрицы соответствует некоторому связывающему подграфу H_θ . Возникает задача нахождения системы представителей с наименьшим дизъюнктивным весом (ΔM -задача).

Л И Т Е Р А Т У Р А

1. Кристофидес Н. Теория графов. Алгоритмический подход. - М.: Мир, 1978. - 432 с.

2. Агибалов Г.П., Беляев В.А. Технология решения комбинаторно-логических задач методом сокращенного обхода дерева поиска. - Томск: Изд-во Том. ун-та, 1981. - 126 с.

Ю. И. Дубков

СОКРАЩЕНИЕ ПЕРЕБОРА В ЗАДАЧАХ КОМБИНАТОРИКИ

Рассматривается задача, частным случаем которой являются известные задачи о покрытиях, о назначениях, коммивояжера и т.п. Предлагается параметрический метод поиска решения поставленной задачи. Формулируется проблема выбора оптимального значения параметра конструируемых на базе этого метода алгоритмов.

Постановка задачи

Пусть $A = (a_1, \dots, a_n)$ — упорядоченная по возрастанию цепочка вещественных чисел, $n \geq 1$,

$\mathcal{A}_n$ — множество всех цепочек натуральных чисел вида $(\pi_1, \dots, \pi_n)$, где $\pi_1 < \pi_2 < \dots < \pi_n$, $\pi_i \in \{1, \dots, n\}$,

$\mathcal{A}_{nm}$ — множество всех цепочек из $\mathcal{A}_n$ длины m ,

$\rho_A \Pi = \sum_{i \in \Pi} a_i$ — стоимость цепочки $\Pi \in \mathcal{A}_n$.

$(A, \mathcal{A})$ — задача: для данных A и $\mathcal{A} \subseteq \mathcal{A}_{nm}$ найти цепочку $\Pi^* \in \mathcal{A}$ такую, что $\rho_A \Pi^* = \min_{\Pi \in \mathcal{A}} \rho_A \Pi$.

Решение

$(A, \mathcal{A})$ — задача относится к классу NP -трудных задач [1], поэтому существующие ныне методы поиска её оптимального решения [2, 3, 4] есть лишь модификации универсального метода тривиального перебора всех элементов множества $\mathcal{A}$. Но так как всякий универсальный алгоритм применяется только к узкому, эмпирически определённом классу $\mathcal{K}$ конкретных задач, то для $\mathcal{K}$ подбирается алгоритм (использующий ту или иную стратегию перебора), который наилучшим образом экономит вычислительные ресурсы при поиске решений задач именно из этого класса $\mathcal{K}$. Одну из таких стратегий мы и рассмотрим ниже. Она заключается в следующем. Для всяких $i \in \{1, \dots, m\}$ и $j \in \{i, i+1, \dots, i+m-i\}$ с помощью оценочных соотношений устанавливается возможность для элемента j быть i -й компонентой искомого решения Π^* . Если такая возможность существует, то число j объявляется кандидатом на замещение i -й позиции цепочки Π^* . Затем из кандидатов формируются двойки, тройки и т.д., m -ки кандидатов, которые, в свою очередь, оцениваются на возмо-

жность быть частью цепочки Π^* . В процессе построения Π^* ужесточаются оценочные критерии и соответственно уточняются списки кандидатов, что обеспечивает сходимость процесса к решению задачи. Опишем в общих чертах алгоритм, основанный на нашей стратегии, в котором используется один вариант интересующих нас оценочных критериев.

Пусть $\mathcal{H}_\alpha$ — множество цепочек из $\mathcal{H}_n$ такое, что $\mathcal{H}_\alpha \cap \mathcal{H} = \mathcal{H}$ и $\mathcal{H}_\alpha$ содержит все подцепочки всех цепочек из $\mathcal{H}$. Здесь подцепочкой цепочки Π называется произвольная цепочка, полученная из Π удалением её некоторых членов. Сформируем множества $M_i = \{i, i+1, \dots, n-m+i\}$, $i = \overline{1, m}$, и пусть p есть верхняя оценка стоимости оптимального решения Π^* , в частности, если нам известно, что некоторая цепочка Π принадлежит множеству $\mathcal{H}$, то полагаем $p = p_\Pi$, в противном случае $p = \sum_{i=1}^n a_i + 1$.

I. Из множества M_1 удаляем все элементы x такие, для которых не существует цепочки $\Pi_x = (x, i_2, \dots, i_m)$, удовлетворяющей соотношениям:

$$\Pi_x \in \mathcal{H}_{nm}, \quad (I)$$

$$p_x = p_\Pi \leq p, \quad (2)$$

$$i_2 \in M_2, \dots, i_m \in M_m,$$

$$(x, i_2), \dots, (x, i_m) \in \mathcal{H}_\alpha.$$

Если элемент x остался в M_1 , то p_x назовём весом этого элемента.

Затем из множества M_1 удаляются все элементы x такие, для которых не существует цепочки $\Pi_x = (i_2, x, i_3, \dots, i_m)$, удовлетворяющей соотношениям (I), (2) и

$$i_2 \in M_1, i_3 \in M_2, \dots, i_m \in M_m, \\ (i_2, x), (x, i_3), \dots, (x, i_m) \in \mathcal{H}_\alpha.$$

Аналогично обрабатываются множества $M_2, \dots, M_m$ и возвращаются в I. Процедуру обработки множеств $M_1, \dots, M_m$ продолжают до тех пор, пока очередная итерация изменяет элементный состав наших множеств. Таким образом, формируются множества одноэлементных кандидатов (I-кандидатов) на замещение соответствующих позиций цепочки Π^* . Элементы всякого множества M_i ранжируются по возрастанию их весов, $i \in \overline{1, m}$.

2. Пусть $x \in M_1$ и $xM_2 = \{(x, y) \in \mathcal{H}_\alpha : y \in M_2\}$, $i = \overline{2, m}$.

Из множества xM_2 удаляем все элементы (x, y) такие, для которых не существует цепочки $\Pi_{xy} = (x, y, i_3, \dots, i_m)$, удовлетворяющей

щей соотношениям:

$$\Pi_{xy} \in \Pi_{nm}, \quad (3)$$

$$\rho_{xy} = \rho_A \Pi_{xy} < \rho, \quad (4)$$

$$(x, i_1) \in x M_1, \dots, (x, i_m) \in x M_m,$$

$$(x, y, i_1), \dots, (x, y, i_m) \in \Pi_\alpha.$$

Если элемент (x, y) остался в $x M_2$, то ρ_{xy} назовём весом этого элемента.

Затем из множества $x M_2$ удаляем все элементы (x, y) такие, для которых не существует цепочки $\Pi_{xy} = (x, i_2, y, i_4, \dots, i_m)$, удовлетворяющей соотношениям (3), (4) и

$$(x, i_2) \in x M_2, (x, i_4) \in x M_4, \dots, (x, i_m) \in x M_m,$$

$$(x, i_2, y), (x, y, i_4), \dots, (x, y, i_m) \in \Pi_\alpha.$$

Аналогично обрабатываются множества $x M_4, \dots, x M_m$ и возвращаются в 2. Процедуру обработки множеств $x M_2, \dots, x M_m$ продолжают до тех пор, пока очередная итерация изменяет элементный состав наших множеств. Таким образом, формируются множества двухэлементных кандидатов (2-кандидатов) на замещение соответствующих частей цепочки Π^* , начинающихся с фиксированного элемента $x \in M_1$. Аналогично формируются множества 2-кандидатов, начинающихся с любого x из M_1 . Элементы всякого множества $x M_i$ ранжируются по возрастанию их весов, $x \in M_1, i \in \{2, \dots, m\}$. Совершенно аналогично можно построить множества 3-кандидатов, 4-кандидатов и т.д. Благодаря ранжированию весов кандидатов, нетрудно организовать процедуру их оптимального перебора с целью скорейшего построения m -кандидата Π , для которого выполняется $\Pi \in \Pi$ и $\rho_A \Pi < \rho$. Если такой кандидат Π найден, то полагаем $\rho = \rho_A \Pi$ и возвращаемся в I. Если же все кандидаты просмотрены, но среди них отсутствует m -кандидат, то в случае $\rho = \sum_{i=1}^m \alpha_i + 1$ решения Π^* не существует, так как $\Pi = \emptyset$, иначе решением Π^* является цепочка, выделенная нами на предыдущем шаге итерации и стоимость которой есть ρ .

Замечания

I. Индекс α можно рассматривать как параметр описанного выше универсального алгоритма поиска решения (A, Π) -задачи. Очевидно, что чем мощнее множество Π_α , тем проще формирование множеств

кандидатов, однако эти множества сами становятся достаточно представительными и при поиске решения Π^* приходится осуществлять их полный перебор. Проблема выбора оптимального значения параметра алгоритма заключается в формировании классов $(A, \mathcal{T})$ -задач и нахождении для каждого класса такого значения параметра α , при котором поиск решения Π^* для каждой задачи из этого класса осуществляется с минимальными затратами.

2. Нетрудно построить модификации нашего алгоритма, отыскивающие как приближённые решения $(A, \mathcal{T})$ -задачи, так и решения задачи на допустимость: для данных A , $\mathcal{T}$ и вещественного d найти цепочку $\Pi^* \in \mathcal{T}$ такую, что $\rho_A \Pi^* < d$.

3. Тривиально доказывается утверждение: класс всех $(A, \mathcal{T})$ -задач совпадает с классом $(A', \mathcal{T}')$ -задач, в которых $\mathcal{T}' \subseteq \mathcal{T}_{n,m}$ и $m \leq n/2$.

4. Как правило, задачи о назначениях, коммивояжера и т.п. ставятся в матричной форме. Легко осуществить переход к постановке таких задач в векторной форме. Для этого элементы матрицы стоимостей $A' = \|a_{ij}\|$, $i = \overline{1, p}$, $j = \overline{1, q}$, "вытягиваются" в упорядоченную по возрастанию a_{ij} цепочку $A = (a_1, \dots, a_n)$, $n = p \cdot q$, соответственно этому упорядочению "вытягиваются" в булевы векторы все матрицы множества $\mathcal{E}'$ булевых матриц (в котором надо найти оптимальное решение), затем каждый вектор $E = (e_1, \dots, e_n)$ из вновь образовавшегося множества $\mathcal{E}''$ булевых векторов преобразуется в цепочку $(i_1, \dots, i_m)$, где i_j - номер j -й слева единицы в векторе E . Множество всех таких цепочек и есть множество $\mathcal{T}'$. Решение Π^* $(A, \mathcal{T})$ -задачи нетрудно преобразовать в булеву матрицу из $\mathcal{E}'$, являющуюся решением задачи в матричной форме.

Литература

1. Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. - М.: Мир, 1982. - 416 с.

2. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. - М.: Мир, 1979. - 536 с.

3. Кристофидес Н. Теория графов. Алгоритмический подход. - М.: Мир, 1978. - 432 с.

4. Агибалов Г.П., Беляев В.А. Технология решения комбинаторно-логических задач методом сокращённого обхода дерева поиска. - Томск: Изд-во Том. ун-та, 1981. - 126 с.

В.А.Беляев

АЛГОРИТМ РАСКРАСКИ ГРАФА МЕТОДОМ СОКРАЩЁННОГО ОБХОДА ДЕРЕВА ПОИСКА

Задача раскраски графа находит многочисленные приложения на различных этапах проектирования цифровой аппаратуры. В конструкторском проектировании задача раскраски возникает в связи с вопросами компоновки, распределения по слоям и трассировки межсоединений элементов схем. В логическом проектировании она возникает при минимизации числа состояний автомата, при кодировании его внутренних состояний и при получении минимальных д.н.ф. булевых функций [1]. В теоретическом программировании к ней сводится задача получения наиболее экономного распределения памяти в операторных схемах программ [2, 3]. Известная трудоёмкость существующих алгоритмов решения этих задач практической сложности побуждает к изысканию новых подходов и методов, совершенствованию известных алгоритмов. В работе предлагается усовершенствованный алгоритм решения задачи раскраски графа методом сокращённого обхода дерева поиска [4]. Усовершенствование проводится по двум путям: во-первых, в случае существования в заданном графе разделяющего полного подграфа решение задачи раскраски всего графа заменяется решением более простых задач раскраски для покрывающих подграфов с меньшим числом вершин, во-вторых, вводятся более эффективные, чем в [4], параметры метода сокращённого обхода.

Задача раскраски графа ставится следующим образом: для заданного симметрического графа $G=(V, U)$ требуется указать такое разбиение множества вершин V , называемое минимальной раскраской графа G , в котором число классов (красок) минимально и каждый класс есть внутренне устойчивое множество (ВУМ), т.е. множество с попарно несмежными вершинами.

Введём необходимые определения и понятия. Говорят, что вершина $x \in V$ связного графа $G=(V, U)$ является точкой сочленения, если подграф, получаемый удалением x , несвязен [5]. Полный подграф H в графе G называется разделяющим полным подграфом, если подграф, получаемый удалением H из G , несвязен. По определению, одновершинный разделяющий подграф является точкой сочленения. Пусть H — разделяющий полный подграф в графе G и $G'_1, \dots, G'_k$, $k \geq 2$ подграфы, являющиеся компонентами связности в $G - H$. Обозначим через $R_1, \dots, R_k$ минимальные раскраски подграфов

$G_i = G'_i \cup H, \dots, G_k = G'_k \cup H$ соответственно, а через $\chi(G)$ — хроматическое число графа G , т.е. число классов (красок) в минимальной раскраске графа G , тогда справедливы следующие соотношения:

$$\chi(G) \geq \max(\chi(G_1), \dots, \chi(G_k)), \quad (I)$$

$$\chi(G_1), \dots, \chi(G_k) \geq \chi(H).$$

Из определения раскраски следует, что все вершины полного подграфа H содержатся в разных блоках разбиений $R_1, \dots, R_k$. Пусть $\chi(H) = h$; перенумеруем все вершины в H натуральными числами $1, 2, \dots, h$ и в соответствии с этой нумерацией упорядочим блоки в разбиениях $R_1, \dots, R_k$ так, чтобы блок, содержащий вершину i из H , имел номер $i \in \{1, \dots, h\}$ в своём разбиении, порядок остальных блоков (если они имеются) в соответствующих разбиениях $R_1, \dots, R_k$ произволен. Упорядоченные таким образом разбиения обозначим через $R_1^*, \dots, R_k^*$ соответственно. Пусть $t = \max(|R_1|, \dots, |R_k|)$ и пусть для определённости $R_i^* = (A_1^i, \dots, A_{t_i}^i)$, $i \in \{1, \dots, k\}$, тогда построим множество $R^* = \{B_1, \dots, B_t\}$, в котором каждый элемент B_j , $j \in \{1, \dots, t\}$ есть объединение классов с номерами j из разбиений $R_1^*, \dots, R_k^*$, т.е.

$$B_j = \bigcup_{i=1}^k A_j^i, \quad j \in \{1, \dots, t\} \text{ и } A_j^i = \emptyset \text{ где } j > t_i \in R_i. \quad (2)$$

Непосредственно из построения следует, что множество $R^* = \{B_1, \dots, B_t\}$ образует разбиение множества V вершин графа G . В дальнейшем процесс получения разбиения R^* по правилу (2) будем называть склеиванием разбиений $R_1, \dots, R_k$, а само разбиение R^* — результатом склеивания.

Имеет место следующая теорема.

Теорема I. Множество $R^* = \{B_1, \dots, B_t\}$ есть минимальная раскраска графа G .

Сначала покажем, что каждый блок B_j , $j \in \{1, \dots, t\}$, разбиения R^* есть внутренне устойчивое множество в графе G . Из определения разделяющего полного подграфа $H \subset G$ следует, что объединение любых внутренне устойчивых множеств, взятых по одному из разных компонент связности графа $G \setminus H$, является внутренне устойчивым множеством в графе G . В дальнейшем будем называть такое объединение ВУМ типа I. Рассмотрим набор B блоков, взятых по одному из каждого разбиения-раскраски $R_1, \dots, R_k$ соответствующих графов $G'_1 \cup H, \dots, G'_k \cup H$ и содержащих одну и ту же вершину a полного графа H . Удалив вершину a из каждого класса набора B , получим

набор внутренне устойчивых множеств из разных компонент связности $G'_1, \dots, G'_k$, объединение которых есть ВУМ типа 1. В силу того, что вершина a графа H совместима с каждым подмножеством в наборе B , она совместима и с объединением этих подмножеств, т.е. объединение классов в наборе B есть внутренне устойчивое множество, назовём его ВУМ типа 2. Из правила (2) построения блоков разбиения R^* следует, что каждый блок B_j является ВУМ типа 1 или ВУМ типа 2, т.е. состоит из попарно несмежных вершин в G . Следовательно, каждый класс разбиения R^* есть внутренне устойчивое множество и R^* есть раскраска графа G .

По построению число блоков в R^* равно t , следовательно, на основании (1) R^* есть минимальная раскраска графа G .

Таким образом, процесс минимальной раскраски заданного графа G в случае существования в нём разделяющего полного подграфа H предлагается разлагать на следующие этапы:

- нахождение компонент связности $G'_1, \dots, G'_k$ графа $G \sim H$;
- нахождение минимальных раскрасок $R_1, \dots, R_k$ для каждого из графов $G'_1 \cup H, \dots, G'_k \cup H$;
- получение минимальной раскраски R^* графа G путём склеивания разбиений-раскрасок $R_1, \dots, R_k$ по правилу (2).

Преимущество такого подхода подтверждают следующие рассуждения. Поскольку объём перебора $\delta(n, m)$ при решении задачи раскраски с ростом числа n и m соответственно вершин и рёбер в графе G растёт нелинейно, то суммарный объём вычислений, затрачиваемый для раскраски подграфов $G_1, \dots, G_k$, покрывающих G , значительно ниже, чем $\delta(n, m)$. Следовательно, располагая эффективным алгоритмом нахождения разделяющего полного подграфа в заданном графе G , удаётся значительно сократить объём вычислений, заменяя задачу раскраски графа G задачами раскраски графов $G_1, \dots, G_k$ с меньшим числом вершин, которые могут быть решены любым известным алгоритмом.

В [4] задача раскраски графа решается методом сокращённого обхода дерева поиска, в качестве параметров используется алгоритм Ψ построения максимальных внутренне устойчивых множеств, операция Ψ сокращения и функция F_0 нижней оценки, которые в основном определяют эффективность метода в целом. В данной работе предлагаются улучшенные параметры, повышающие эффективность метода сокращённого обхода для задачи раскраски графа.

Ниже описывается более эффективный, чем в [4], алгоритм Φ_1 построения множества всех максимальных внутренне устойчивых под-

множеств вершин заданного графа $G = (V, U)$, содержащих некоторую фиксированную вершину v из V ; при изложении алгоритма предполагается, что вершины графа G упорядочены некоторым образом и образуют ряд $v_1, \dots, v_n$, в котором $v_i = v$. Пусть также $O_i(A)$ обозначает множество всех вершин в G , смежных вершинам из A . Описываемый алгоритм осуществляет направленный перебор по дереву и является модификацией алгоритма построения максимальных независимых множеств из [6]. Работа алгоритма состоит в последовательном выполнении шагов ветвления и возвращения, причём шаг возвращения выполняется лишь тогда, когда невозможен шаг ветвления. В процессе поиска на некотором шаге i рассматриваются три множества: A_i - строящееся внутренне устойчивое множество; B_i - наибольшее множество совместимых с A_i вершин такое, что $A_i \cap B_i = \emptyset$, и после добавления любой вершины из B_i к A_i получается новое совместимое множество A_{i+1} , т.е. B_i состоит из вершин графа G , являющихся кандидатами на расширение множества A_i и, наконец, множество C_i содержит все несовместимые с A_i вершины из G , т.е. $C_i = O_i(A)$. Тогда шаг ветвления в дереве поиска состоит в выборе вершины $v_{i_2} \in B_i$, включении её в A_i для построения множества $A_{i+1} = A_i \cup \{v_{i_2}\}$, вычислении новых множеств: $B_{i+1} = B_i - (\{v_{i_2}\} \cup O_i(v_{i_2}))$, $C_{i+1} = C_i \cup O_i(v_{i_2})$. Шаг возвращения алгоритма Ψ состоит в возвращении к исходному множеству A_i и удалении вершины v_{i_2} из B_i . Нетрудно видеть, что множество A_i на некотором шаге i является максимальным тогда и только тогда, когда

$$B_i = \emptyset \quad \text{и} \quad A_i \cup C_i = V. \quad (3)$$

В [6] предложено достаточное условие для осуществления шага возвращения, заключающееся в следующем. Пусть B_i^* подмножество тех вершин из первоначального B_i , которые уже использовались для расширения A_i и B_i^+ - те вершины из B_i , которые ещё не использовались. Тогда условие существования в B_i^* вершины v такой, что

$$O_i(v) \cap B_i^+ = \emptyset, \quad (4)$$

является достаточным для осуществления шага возвращения. Данное условие в алгоритме Ψ удобнее использовать при выполнении шага ветвления в следующем виде. Пусть $v_{i_2} \in B_i$ вершина, выбранная для ветвления и v_i вершина с максимальным номером в множестве $O_i(v_{i_2})$, тогда, удалив из B_i вершины с номерами $i+1, \dots, n$, мы тем самым исключим из кандидатов на расширение множества A_i все те вершины v , для которых выполнено условие (4), поскольку любое расширение множества $A_{i+1} = A_i \cup \{v_{i_2}\}$ не является максимальным внутренне устойчивым множеством.

состоит из следующих подмножеств в V : $\{1, 3, 5, 9\}$, $\{1, 3, 5, 10\}$, $\{1, 3, 8, 9\}$, $\{1, 6, 8, 9\}$. Для простоты изображения на рис.2 множество $\{a, b, c\}$ обозначается без скобок через a, b, c .

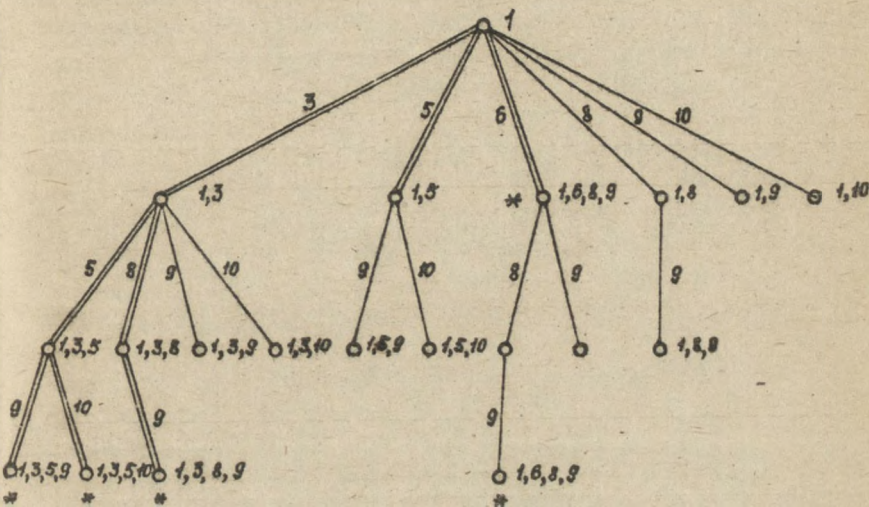


Рис. 2

За функцию нижней оценки F_0 принимается число $h = \max(\alpha, \delta)$, где α есть наибольшее число вершин в максимальном полном подграфе в G (кликовое число G [6]) и $\delta = \lfloor \frac{n}{2} \rfloor$, n — число вершин в графе G , а c — число независимости G [6] (или кликовое число дополнительного графа $\bar{G}$).

Кликовое число χ для любого графа находится следующим быстродействующим алгоритмом χ_0 . При изложении алгоритма предполагается, как и раньше, что вершины в графе G упорядочены и образуют ряд $v_1, \dots, v_n$ и граф G не пустой ($n \geq 2$), т.е. содержит по крайней мере одно ребро. Алгоритм осуществляет направленный перебор с использованием дерева поиска. В алгоритме осуществляется последовательное выполнение шагов двух типов — ветвления и возвращения. Причем шаг ветвления выполняется всегда, когда невозможен шаг возврата. На каждом шаге s ветвления, рассматривается текущее множество вершин D_s ($D_1 = V, s = 1$), в котором выбирается вершина с наименьшим номером, строится множество $D_{s+1} = D_s \setminus \{u\}$ и, если $|D_{s+1}| + s \leq \chi$, выполняется шаг возвращения, заключающийся в вычислении новых $D_s = D_s \cup \{u\}$ и $s := s - 1$ и переходе к выполнению шага s

Кроме того, при выполнении очередного шага ветвления сократить число ветвлений можно исключив в A_3 все те вершины v из B_3 , для которых справедливо

$$O_1(v) \subseteq C_3, \quad (5)$$

поскольку отсутствие любой такой вершины в расширении множества приведёт к нарушению условия (3) максимальности.

Ниже даётся формальное описание алгоритма.

Алгоритм φI .

1. $A_1 := \{v_1\}$, $C_1 := A_1 \cup O_1(A_1)$, $B_1 := V - C_1$, $D_1 := B_1$, $\delta := 0$.
2. Положим $\delta := \delta + 1$ и, если $B_\delta = \emptyset$, выполним п.4, иначе п.3.
3. $D_\delta := B_\delta$ и, если не существует вершины $v \in B_\delta$ со свойством $O_1(v) \subseteq C_\delta$, выполним п.5, иначе положим $A_{\delta+1} := A_\delta \cup \{v\}$, $B_{\delta+1} := B_\delta - \{v\}$ и, в случае $B_\delta = \emptyset$, выполним п.3, а при $B_\delta = \emptyset$ - п.4.
4. Если $A_{\delta+1} \cup C_{\delta+1} \neq V$, то п.5, иначе $A_{\delta+1}$ отмечается как очередной элемент перечисляемого множества и п.5.
5. Если $D_\delta = \emptyset$, положим $\delta := \delta - 1$ и п.6, иначе пусть u есть вершина с наименьшим номером в D_δ , а v_2 - вершина с наибольшим номером в $O_1(u)$, тогда положим $A_{\delta+1} := A_\delta \cup \{u\}$, $B_{\delta+1} := B_\delta - \{u\}$, $C_{\delta+1} := C_\delta \cup \{u\} \cup O_1(u)$, $D_{\delta+1} := D_\delta - \{v_2, \dots, v_k\}$, $B_{\delta+1} := B_\delta - C_{\delta+1}$ и выполним п.2.
6. Если $\delta > 0$, выполним п.5, в противном случае алгоритм свою работу заканчивает.

Пример 1. Проиллюстрируем работу описанного алгоритма φI . Построим множество всех максимальных внутренне устойчивых множеств (МВУМ)

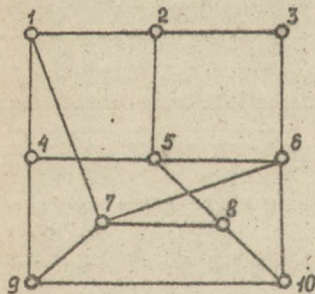


Рис.1

для графа G из [7] на рис.1 с множеством вершин $V = \{1, \dots, 10\}$ и начальной фиксированной вершиной $v = 1$. На рис.2 изображено дерево перебора D , которое обходит алгоритм φI из [4] в процессе построения МВУМ для данного примера; двойными линиями изображена та часть дерева D , которую обходит описанный выше алгоритм φI при построении того же множества.

Корню дерева на рис. 2 соответствует множество $A_1 = \{1\}$, а концевые вершины, отмеченные знаком (*), представляют перечисляемые МВУМ. Каждое ребро помечено вершиной u , выбираемой для расширения A_3 при реализации соответствующего шага ветвления в алгоритме φI , а каждая вершина помечена соответствующим множеством A_3 .

Таким образом, множество всех максимальных совместимых подмножеств

ветвления; в противном случае выполняется очередной шаг ветвления. В случае $|D_{\delta+1}| = \emptyset$ величина γ принимается равной δ и выполняется шаг возврата. После завершения работы алгоритма величина γ есть кликовое число графа G .

Алгоритм γ_0 .

1. $\delta := 1$, $\gamma := 2$, $D_1 := \{v_1, \dots, v_n\}$.
2. Если $D_\delta = \emptyset$, то п.3, иначе в множестве D_δ выбрать вершину v_α с наименьшим номером, положить $D_\delta := D_\delta - \{v_\alpha\}$, $D_{\delta+1} := D_\delta \cap Q(v_\alpha)$ и, если $|D_{\delta+1}| + \delta \leq \gamma$, то п.2, в противном случае положить $\delta := \delta + 1$ и, если $D_\delta \neq \emptyset$, выполнить п.2, иначе положить $\gamma := \delta - 1$ и п.3.
3. Положить $\delta := \delta - 1$ и при $\delta \neq 0$, если $|D_\delta| + \delta > \gamma$, выполним п.2, а при $|D_\delta| + \delta \leq \gamma$ - п.3; при $\delta = 0$ алгоритм работу заканчивает, значение γ есть кликовое число графа G .

Для нахождения C -числа независимости графа G достаточно применить алгоритм γ_0 к графу $\bar{G}$.

Например, применив описанный алгоритм к графу G из [6] на рис.3 с множеством вершин $V = \{1, \dots, 16\}$ получим кликовое число $\alpha = 2$, применив алгоритм к графу $\bar{G}$ получим кликовое число дополнительного графа $\beta = 5$; в результате имеем $F_0 = \max(2, \lceil 16/5 \rceil) = 4$.

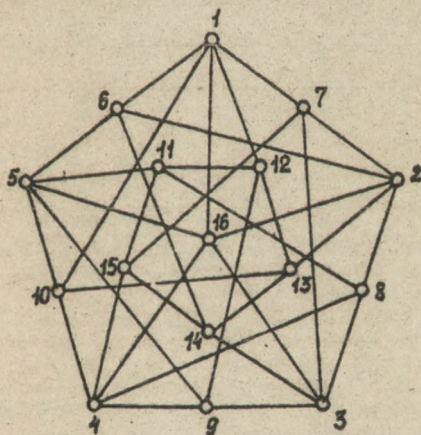


Рис.3

Применение указанных параметров позволяет находить минимальную раскраску графов методом сокращённого обхода дерева поиска с меньшими вычислительными затратами.

Л и т е р а т у р а

1. Фрицнович Г.Ф. Синтез асинхронных конечных автоматов и раскраска вершин графов. - В кн.: II Всесоюзное совещание по теории релейных устройств и конечных автоматов. Тезисы докладов. Рига, 28 сент. - I окт. 1971, с. 33 - 34.
2. Бухгольц Н.В., Дьяченко В.Ф., Лазарев В.Г. и др. Алгоритмы минимизации оперативной памяти ЦВМ. - В кн.: Проблемы синтеза цифровых автоматов. М.: Наука, 1967, с. 112 - 118.
3. Ершов А.П., Кожухин Г.И. Об оценках хроматического числа связанных графов. - ДАН СССР, 1962, т.142, №2, с. 270 - 273.
4. Агибалов Г.П., Беляев В.А. Технология решения комбинаторно-логических задач методом сокращённого обхода дерева поиска. Томск, Изд-во Том. ун-та, 1981. - 125 с.
5. Берж К. Теория графов и её применения. - М.: ИЛ, 1962. - 320 с.
6. Кристофидес Н. Теория графов. Алгоритмический подход. - М.: Мир, 1976. - 432 с.
7. Wilkov R.S., Kim W.H. A Practical Approach to the Chromatic Partition Problem. - J. Franklin Inst., 1970, 289, №5, p. 333 - 349.

В.А.Ильин, С.С.Катаева

ПОЛНЫЙ ОБХОД ГРАФА С ОГРАНИЧЕНИЯМИ НА СТРУКТУРУ МАРШРУТА

Введение

Задача построения на графе маршрута, проходящего по каждому ребру, по крайней мере, один раз и имеющего минимальную суммарную длину, известна в литературе как задача минимального полного обхода графа. Если конец и начало обхода совпадают, то эту задачу называют задачей почтальона [1] или китайского почтальона [2]. Задача построения маршрута полного обхода, периодически проходящего через заданную вершину, известна в прикладной теории графов как задача "сбора домашнего мусора" [2]. Содержательно она интерпретируется как задача построения оптимального маршрута для мусоросборщика, обслуживающего сеть улиц и по истечении каждого рабочего дня возвращающегося на базу. В вышеуказанных задачах не накладывается никаких ограничений на структуру маршрута.

В настоящей статье рассматривается задача полного обхода с некоторыми дополнительными ограничениями на маршрут, касающимися более детального учета реальных условий, накладываемых на маршрут обхода в ряде прикладных задач, в частности, и в задаче "сбора домашнего мусора".

Сущность дополнительных ограничений состоит в следующем. Во-первых, вводится два режима движения по графу, которые для задачи сбора мусора интерпретируются как режимы сбора и транспортировки. Во-вторых, периодически маршрут должен проходить через какие-либо вершины выделенного множества вершин. В задаче сбора мусора выделенное множество вершин - это места сброса или утилизации мусора.

Данные ограничения являются вполне естественными и для целого ряда других прикладных задач, например, для задач планирования контрольно-проверочных работ на сетях связи или сетях дорог.

Алгоритмы, описанные в литературе (см., например, [1-7]) для задач без ограничений, при наличии ограничений применить не удастся. В статье предлагаются поисковые алгоритмы построения оптимального обхода графов. Сущность алгоритмов состоит в генерации последовательности маршрутов полного обхода, удовлет-

Вносящих всем ограничениям. Алгоритмы дают возможность последовательно улучшать маршрут. Ресурсы времени и памяти ЭВМ накладывают ограничения на ширину поиска. Предложенные в статье алгоритмы находят приближенные к оптимальным маршруты полного обхода при заданных ресурсах на память и время. Чем больше отведенные ресурсы, тем ближе к оптимальному полученный маршрут. Предложенные алгоритмы в приемлемое время дают хорошие результаты на графах с несколькими сотнями вершин и ребер.

1. Постановка задачи

Пусть дан неориентированный граф G . Через X обозначим множество вершин графа G , через U - множество ребер, через $D = \|d_{ij}\|_{N \times N}$ - матрицу длин ребер (N - количество вершин графа). На множестве X выделено множество K . Задано число T , соответствующее временному периоду движения по графу (рабочий день, цикл). Задана начальная вершина x_0 . Дано некоторое подвижное средство (ПС), способное двигаться по графу с какой-либо из двух скоростей v_1 и v_2 , $v_1 < v_2$. Скорость v_1 является рабочей скоростью, скорость v_2 - транспортной или скоростью свободного пробега. Цель ПС заключается в обходе графа G с выполнением следующих условий:

1. Обход графа G начинается и оканчивается в вершине x_0 , $x_0 \in K$.
2. Каждое ребро графа должно быть пройдено с рабочей скоростью только один раз. С транспортной скоростью ребра могут проходиться сколько угодно раз.
3. Движение по графу G совершается рабочими циклами, длительность каждого из которых одинакова и равна T . Каждый i -й рабочий цикл ($i = 1, 2, \dots$) состоит в следующем. Движение ПС начинается в некоторой вершине $k_{i-1} \in K$ ($k_0 = x_0$) в момент $(i-1)T$. ПС прекращает движение в вершине $k_i \in K$ в момент $t_i \leq iT$ (не исключается случай, что $k_{i-1} = k_i$). От момента t_i до начала $(i+1)$ цикла движение не производится. Вершина k_i является начальной вершиной периода $(i+1)$. Внутри каждого рабочего цикла ПС может двигаться как со скоростью v_1 , так и со скоростью v_2 . Чередование скоростей может быть произвольным в пределах условия 2.

В задаче "сбора домашнего мусора" необходимо предусмотреть

периодический возврат в начальную вершину (базу). Поэтому введем еще одно условие.

4. По истечении каждого промежутка времени длительности не более, чем $n \cdot T$, с момента выхода из X_0 маршрут должен вновь проходить через вершину X_0 (число n фиксировано).

Необходимо построить такой маршрут, удовлетворяющий условиям I - 4, для которого время обхода минимально.

2. Алгоритмы целенаправленного перебора

Предложим алгоритм (точнее говоря, серию алгоритмов с одной типичной схемой) упорядоченного поиска маршрута, удовлетворяющего условиям I - 4 и оптимального по критерию минимума времени реализации маршрута. Прежде введем ряд обозначений. Пусть на графе G дана некоторая цепь C_i (не обязательно содержащая все ребра множества U), начинающаяся в вершине X_0 и состоящая из последовательности не обязательно различных ребер $u_{0i_1}, u_{i_1i_2}, \dots, u_{i_{k-1}i_k}$ (ребро u_{ij} соединяет вершины X_i и X_j и имеет длину $\ell(u_{ij})$). Через $\ell(C_i)$ обозначаем сумму длин всех ребер цепи C_i .

Через $m(C_i)$ обозначается число периодов длительности T , понадобившихся для реализации цепи C_i , т.е. цепь C_i реализуется ПС за промежуток времени $[0, m(C_i)T]$. Через $S(C_i)$ обозначается суммарная длина той части цепи C_i , которая была пройдена с рабочей скоростью V_i . Через $L(C_i)$ обозначаем суммарную длину ребер множества

$$U \setminus \{u_{0i_1}, u_{i_1i_2}, \dots, u_{i_{k-1}i_k}\}.$$

Через L_0 обозначаем суммарную длину всех ребер графа G .

Пусть для совокупности величин $L_0, \ell(C_i), S(C_i), L(C_i), m(C_i)$, определенных для каждой цепи C_i , удовлетворяющей условиям 4 - 5 определена положительная числовая функция $f(L_0, \ell(C_i), S(C_i), L(C_i), m(C_i))$, которой придается смысл оценки качества цепи C_i . Значение функции f для цепи C_i обозначаем $f(C_i)$ или f_i . Выбор функции f определяется критерием оптимальности следующим образом. Если цепь удовлетворяет условиям I - 4, то $f(C)$ - значение критерия оптимальности для цепи C . Вид функции f полностью определяет сущность алгоритмов, общая схема которых ниже описывается. Обсуждение вопроса задания функции f и связанных с этим следствий будет дано после описания алгоритма.

Описание алгоритма

Шаг 1. Из вершины x_0 (считаем, что $x_0 \in K$) строятся все цепи, удовлетворяющие условиям 2-4 и реализуемые за один период T . Получается множество цепей $C_1, C_2 \dots C_{n_1}$. Цепям, согласно условию 3, соответствует последовательность $k_1, k_2 \dots k_{n_1}$ вершин, являющихся концевыми вершинами цепей. Вычисляются оценки $f(C_i)$, $i=1, 2 \dots n_1$.

Шаг 2. Пусть построены цепи $C_1, C_2 \dots C_{n_p}$. Среди чисел $f(C_i)$, $i=1, 2 \dots n_p$ выбирается минимальное. Пусть $f(C_i) = \min_{i=1, 2, \dots, n_p} f(C_i)$. Из вершины k_j , соответствующей концу цепи C_j , строятся цепи R_{jq} , $q=1, 2 \dots$, удовлетворяющие условиям 2-4 и реализуемые за один период T (за промежуток времени $[m(C_j)T; (m(C_j)+1)T]$). Из последовательности $C_1 \dots C_{n_p}$ цепь C_j удаляется. Формируются новые цепи, каждая из которых получается добавлением к цепи C_j какой-либо из цепей R_{jq} , $q=1, 2 \dots$. Новые цепи вносятся в последовательность цепей и для них вычисляются оценки.

Шаг 3. Если среди цепей $C_1 \dots C_{n_{p+1}}$ есть цепь C^* , удовлетворяющая всем условиям 1-4 и такая, что

$$f(C^*) \leq f(C_i), \quad i=1, 2 \dots n_{p+1},$$

то вычисления оканчиваются и C^* - оптимальная цепь. Иначе алгоритм возвращается на шаг 2.

Теперь вернемся к вопросу о выборе функции $f(L(C), l(C), S(C), L(C), m(C))$. Если в качестве критерия оптимальности выступает критерий минимума времени полного обхода сети G , то f можно выбрать в виде

$$f(C) = m(C)T + \frac{L(C)}{v_1}. \quad (I)$$

Величина $m(C)T$ - время движения по цепи C , $L(C)/v_1$ - нижняя оценка времени обхода оставшейся после обхода цепи C части графа G . Величина $L(C)/v_1$ введена в функцию из тех соображений, что время обхода оставшейся части графа G не может быть меньше времени обхода суммарной длины $L(C)$ с рабочей скоростью v_1 . На шаге 2 алгоритма поиска оптимального решения дистраивается та цепь, для которой оценка времени прохождения минимальна.

Из интуитивных соображений ясно, что поиск оптимального решения на основе функции (I) является довольно широким и при больших размерностях графа G возникающие при этом массивы могут

иметь очень большую размерность.

Для сужения ширины поиска уместнее ввести следующую функцию

$$f(C) = \frac{L(C)}{S(C)} \cdot m(C) \cdot T, \quad (2)$$

смысл которой поясняется следующим образом. Величина $S(C)/m(C) \cdot T$ является средней скоростью прохождения ребер цепи C в рабочем режиме. Чем выше эта скорость, тем меньше времени затрачивается на прохождение всех ребер графа G . По скорости $S(C)/m(C) \cdot T$ оценивается время прохождения оставшейся после удаления цепи C части графа G , суммарная длина ребер этой части равна $L(C)$. Разделив $L(C)$ на $S(C)/m(C) \cdot T$, получаем функцию (2). Еще одну функцию можно получить, если в (2) $L(C)$ заменить на L - суммарную длину ребер графа G . В этом случае $f(C)$ имеет смысл оценки времени обхода всего графа G , а не только оставшейся части. Заметим, что оценка типа (2) уже не гарантирует нахождение оптимальной цепи, так как в процессе поиска дотраиваются только те цепи, у которых среднее время обхода графа в рабочем режиме оценивается на основе скорости обхода цепи, построенной к данному моменту. Если у оптимальной цепи на начальном отрезке эта скорость мала, то оптимальная цепь в дальнейшем поиске возможно не будет дотраиваться. Но оценки типа (2) дают более узконаправленный поиск и отсекают массу плохих вариантов, что при большой размерности графа G очень существенно. Результатом поиска, возможно, будет некоторое приближенное решение.

Иногда оказывается необходимым наряду с критерием минимума времени рассмотреть возможность одновременной минимизации расстояния, проходимого с транспортной скоростью V_2 . Алгоритм решения задачи остается тем же самым, но функцию f необходимо задавать с учетом возникающих дополнительных требований. Рассмотрим функцию вида

$$f(C) = \frac{L(C) - S(C)}{m(C) \cdot T} + \beta \frac{L(C)}{V_2}, \quad (3)$$

где β - масштабный коэффициент. Величина $L(C) - S(C)$ представляет собой длину пути, пройденного в цепи C с транспортной скоростью V_2 , и первое слагаемое в (3) является средней скоростью движения ТС в транспортном режиме. Второе слагаемое аналогично второму слагаемому в (1). Необходимость введения коэффициента диктуется тем, что первое слагаемое имеет смысл скорости, второе - времени, к тому же коэффициент отражает вес (соответст-

вующий той или иной степени важности) второго слагаемого. При поиске решения на основе функции (3) достраиваются те цепи, для которых минимальны сумма средней скорости движения ТС в транспортном режиме и оценка времени прихождения оставшейся части графа G в рабочем режиме. В результате поиска будет найдено некоторое компромиссное решение. Оно само по себе не будет оптимально ни в смысле минимума времени, ни в смысле минимума расстояния, проходимого в транспортном режиме, оно будет оптимально в смысле минимума комбинации этих двух критериев.

Другой путь нахождения решения при нескольких критериях состоит в том, что каждой уже построенной цепи C_i , $i=1, 2, \dots, n_p$, сопоставляется две оценки $f^{(1)}(C_i)$ и $f^{(2)}(C_i)$, соответствующие двум критериям. Один из критериев (например, время) считается основным, второй (например, расстояние, проходимое в транспортном режиме) дополнительным. Если несколько цепей имеют одно и то же минимальное значение основного критерия, то из этих цепей достраивается та, у которой минимальна оценка по второму критерию.

Анализируя предложенный выше алгоритм, можно заметить, что при некотором видоизменении он аналогичен схеме метода "ветвей и границ" [8], являющегося эффективным методом решения комбинаторных задач. Известно, что метод "ветвей и границ" в каждой конкретной задаче требует учета специфики задачи и наполняется вычислительными алгоритмами только с учетом этой специфики. В методе "ветвей и границ" предполагается, что оценки частичного решения не должны быть больше оценок полного решения. Для функции вида (1) справедливо

$$f(C) \leq f(\bar{C}), \quad (4)$$

где $\bar{C}$ - цепь полного обхода, удовлетворяющая всем условиям I-4; C - часть цепи $\bar{C}$.

Неравенство (4) справедливо в силу следующих простых рассуждений. Имеем

$$f(\bar{C}) = m(\bar{C})T = m(C)T + m(\bar{C} \setminus C)T,$$

где $\bar{C} \setminus C$ - часть цепи $\bar{C}$, получающаяся после удаления цепи C .

Далее

$$\frac{L(C)}{v_1} \leq m(\bar{C} \setminus C)T. \quad (5)$$

Последнее неравенство справедливо в силу определения величины $L(C)/v_1$. Прибавляя к обеим частям неравенства (5) величину

$m(C)T$, получаем

$$m(C)T + \frac{L(C)}{v_1} \leq m(C)T + m(\bar{C} \setminus C)T,$$

т.е. получаем (4). Таким образом, предложенный алгоритм с применением функции (I) удовлетворяет схеме метода "ветвей и границ" и, следовательно, в результате работы данного алгоритма будет получено оптимальное решение. Оптимальность решения следует из свойств общей схемы метода "ветвей и границ" [8].

Рассмотрим еще один подход к построению оценочных функций. Пусть G - исходный граф. По цепочке C формируется граф $G(C)$, в который включаются ребра (и вершины, которые эти ребра соединяют), не пройденные с рабочей скоростью цепочкой C . Множество вершин графа $G(C)$ обозначаем $X(C)$. Во множестве вершин $X(C)$ выделяем компоненты связности, т.е. такие подмножества вершин $X_i(C)$, $i=1 \dots q$, что для любых двух вершин x_{j_1}, x_{j_2} , $j_1 \neq j_2$, $x_{j_1} \in X_{j_1}(C)$, $x_{j_2} \in X_{j_2}(C)$ не существует пути, соединяющего их в графе $G(C)$. Строится новый полный граф $\bar{G}$, имеющий $q+2$ вершин. q вершин в графе $\bar{G}(C)$ определяются количеством q компонент связности. В граф $\bar{G}(C)$ вносится вершина $x_m(C)$, являющаяся конечной в цепочке C , и вносится вершина x_o , являющаяся базой. Вершины графа $\bar{G}(C)$ обозначаем y_i , $i=0, 1 \dots q+1$, где $y_o = x_m(C)$, $y_{q+1} = x_o$. Длина ребер между вершинами y_i, y_j , $i, j=0, 1 \dots q+1$, графа $\bar{G}(C)$ берется равной кратчайшему расстоянию ρ_{ij} (в графе G) между подграфами $G_i(C)$ и $G_j(C)$, представляющими компоненты связности:

$$(G_o(C) = x_m(C), G_{q+1}(C) = x_o).$$

$$\rho_{ij} = \rho(G_i(C), G_j(C)) = \min_{\substack{z \in X_i(C) \\ s \in X_j(C)}} d_{zs},$$

где d_{zs} - кратчайшее расстояние между вершинами x_{i_1} и x_{j_2} в исходном графе G . Матрица D кратчайших расстояний между всеми парами вершин графа G вычисляется в общем алгоритме поиска.

В качестве оценки f цепочки C принимается величина

$$f(C) = m(C)T + \frac{N(C)}{v_1} + \frac{Z(\bar{G}(C))}{v_2}, \quad (6)$$

где $Z(\bar{G}(C))$ - длина минимальной гамильтоновой цепи с началом в y_o и концом в y_{q+1} . Величина $Z(\bar{G}(C))$ - минимально воз-

можная суммарная длина свободного пробега при последующем обходе графа $G(C)$, оставшегося после прохождения цепочки C .

Поиск минимальной гамильтоновой цепи - довольно трудоемкая задача. Предложим другие оценки, основанные на простых идеях оценивания нижней границы величины минимальной гамильтоновой цепи

$$f(C) = m(C)T + \frac{N(C)}{v_1} + \frac{\sum_{k=0}^{q+1} (\rho_k + \lambda_k)}{2v_2}, \quad (7)$$

где ρ_k - длина кратчайшего ребра, выходящего из вершины y_k .

$$\rho_k = \min_{\substack{i \neq k \\ i=0,1,\dots,q+1}} \rho_{ki},$$

λ_k - длина второго кратчайшего ребра, выходящего из вершины y_k ($\lambda_0 = 0$, $\lambda_{q+1} = 0$). Очевидно, что

$$\frac{1}{2} \sum_{k=0}^{q+1} (\rho_k + \lambda_k) \leq Z(\bar{G}(C)).$$

Более простая по сравнению с (7) оценка имеет вид

$$f(C) = m(C)T + \frac{N(C)}{v_1} + \frac{\sum_{k=0}^q \rho_k}{v_2}, \quad (8)$$

где ρ_k - имеет тот же смысл, что и в (7).

Самая простая оценка, основанная на идее оценивания минимальной гамильтоновой цепи, имеет вид

$$f(C) = m(C)T + \frac{N(C)}{v_1} + \frac{\rho(q+1)}{v_2}, \quad (9)$$

где ρ - кратчайшее ребро в $\bar{G}$. Если y_0 или y_{q+1} принадлежат каким-либо компонентам связности, то $\rho = 0$. В этом случае при поиске кратчайшего ребра необходимо из графа $\bar{G}(C)$ исключить нулевые ребра, а в оценке (9) последний член (если в графе $\bar{G}(C)$ было два нулевых ребра) заменяется на $\rho \cdot q / v_2$.

Очевидно, что

$$\rho(q+1) \leq \sum_{k=0}^q \rho_k \leq \sum_{k=0}^{q+1} \frac{\rho_k + \lambda_k}{2} \leq Z(\bar{G}(C)).$$

Каждая последующая оценка из (6)-(9) является более худшей, но и более простой по сравнению с предыдущей.

Алгоритмы перебора довольно громоздки, дают широкий поиск и быстро приводят к переполнению памяти и большому расходу времени. Поэтому они дополняются правилами отсечения на основе использования пороговых маршрутов полного обхода.

2. Алгоритмы построения порогового маршрута

В качестве порогового значения в алгоритмах поиска принимается оценка некоторым способом построенного маршрута полного обхода, удовлетворяющего всем необходимым условиям задачи. Чем лучше пороговый полный обход, чем он ближе к оптимальному, тем лучшее пороговое значение он дает, тем больше отсекается неперспективных направлений поиска.

Рассмотрим несколько алгоритмов формирования маршрутов полного обхода.

Алгоритм 1. Пусть уже построена цепочка C_n , имеющая n вершин, и вершина $x(C_n)$ является конечной в цепочке C_n . Среди ребер, выходящих из $x(C_n)$ и удовлетворяющих условиям включения их в цепь полного обхода, выбирается ребро максимальной длины, не пройденное с рабочей скоростью. Это ребро присоединяется к цепочке C_n и получается цепочка C_{n+1} . Для цепочки C_{n+1} продельвается аналогичная процедура и т.д. Если все ребра, выходящие из $x(C_n)$, уже пройдены в цепочке C_n с рабочей скоростью, то алгоритм переходит к п.2.

2. Среди ребер графа G , еще не пройденных цепочкой C_n с рабочей скоростью, ищется ребро u' , имеющее наименьшее удовлетворяющее условиям включения в цепь полного обхода расстояние до вершины $x(C_n)$. К цепочке C_n добавляется то ребро, выходящее из $x(C_n)$, которое является началом этого кратчайшего пути, ведущего из $x(C_n)$ до ребра u' . Получается цепочка C_{n+1} . Алгоритм переходит к п.1.

В результате работы алгоритма получается некоторая цепь полного обхода с оценкой φ_0 .

Алгоритм 2. Пусть уже построена цепочка C_n . По алгоритму формирования цепочек, проходимых за один период T из вершины $x(C_n)$ последовательно формируются Q_j , $j=1, 2, \dots$, проходимые за k периодов T (k фиксировано). При этом сначала формируются цепочки для одного T , затем из концов этих цепочек формируются цепочки для второго T и так до тех пор, пока не пройдено k периодов. Цепочки Q_j , $j=1, 2, \dots$, заносятся в некоторый массив B . Для каждой Q_j , $j=1, 2, \dots$, вычисляется значение z_j :

$$z_j = \frac{S(Q_j)}{k \cdot T},$$

являющееся средней интенсивностью прохождения ребер с рабочей скоростью в цепочке Q_j .

Среди цепочек $Q_j, j=1, 2, \dots$, выбирается цепочка Q_{j_0} , для которой

$$z_{j_0} = \max_{j=0, 1, \dots} z_j.$$

Если несколько цепочек имеют одинаковую максимальную оценку, то среди них выбирается та, для которой величина $(l(Q_j) - S(Q_j))$ - минимальна (где $l(Q_j)$ - длина цепочки Q_j).

Цепочка Q_{j_0} принимается за продолжение цепи C_n . Массив B очищается. Процедура повторяется для вершины, являющейся концевой в цепочке Q_{j_0} с использованием того же массива B и т.д., пока не будет построена цепь полного обхода. Алгоритм второй заканчивает работу.

Увеличивая k , получаем пороговые цепи, все более приближающиеся оптимальному маршруту полного обхода.

Алгоритм I по существу является схемой серии алгоритмов, отличающихся друг от друга видом оценочной функции, применяемой на шаге 2 при выборе направления движения в случае, когда все ребра, выходящие из текущей вершины, пройдены с рабочей скоростью. Могут быть использованы любые из функций (I)-(3), (6)-(9).

3. Комбинированные алгоритмы и численный пример

На основе алгоритмов целенаправленного поиска и алгоритмов построения порогового маршрута предложим комбинированные алгоритмы, общая схема которых может быть представлена следующим образом.

Шаг 1. С использованием одной из оценок (I)-(3) или (6)-(9) строятся частичные цепочки обхода по алгоритму целенаправленного поиска. Заполняется матрица частичных цепочек до тех пор, пока она еще не переполняется. Как только вся отведенная матрица частичных цепочек заполнена, алгоритм целенаправленного поиска прекращает работу.

Шаг 2. Каждая частичная цепочка из матрицы, начиная с цепочки с минимальной оценкой, достраивается до полного обхода с использованием алгоритмов построения порогового маршрута полного обхода. Те частичные цепочки не достраиваются для которых

значение оценки больше, чем уже имеющееся значение для лучшего порогового обхода.

Проводились численные эксперименты с вышеизложенными алгоритмами. Критерием служило время прохождения маршрута и длина свободного пробега. Кратко опишем результаты моделирования алгоритмов на графе рис.1, имеющего 13 вершин и 22 ребра. Моделирование проводилось на машинах ЕС. Маршрут, полученный по алгоритму 1 построения порогового маршрута, приведен в табл.1. Последовательность номеров указывает на последовательность прохождения ребер. Ниже указана соответствующая маршруту последовательность чередования рабочей (обозначается 1) и транспортной (обозначается 2) скоростей. Время счета алгоритма 4'54" на ЭВМ ЕС 1020.

В табл. 2 приведены маршрут и последовательность чередования скоростей, полученные по алгоритму 2 построения порогового обхода. Маршрут улучшается по сравнению с алгоритмом 1. Общая длина свободного пробега уменьшилась. Время счета возросло до 15'29" на ЭВМ ЕС 1020.

В табл. 3 приведены маршрут и скорости, полученные по алгоритму 3, который отличается от алгоритма 1 использованием цепочкой функции (7) на шаге 2. Маршрут улучшился. Время счета равно 56" на ЭВМ ЕС 1033.

В табл. 4 приведены лучший маршрут и соответствующее ему чередование скоростей, полученные по комбинированному алгоритму. Целенаправленный поиск осуществлялся на основе оценочной функции (1). Размерность матрицы частичных цепочек бралась равной 5000×10 . Алгоритм 3 использовался для достраивания частичных цепочек до полного обхода. Приведенный в табл. 4 маршрут был получен за время 12'40" на ЭВМ ЕС 1033. Здесь время счета было ограничено заранее. По сравнению с маршрутами, приведенными в табл.1-3, маршрут обхода улучшился.

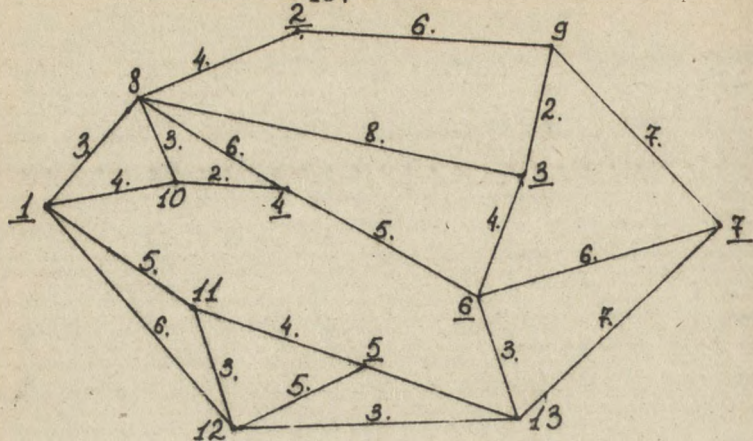


Таблица I

Таблица 2

Таблица 3

Таблица 4

I,II,5,I₂,I₃,6,I₃,6,4,8,I,I₀,4,I₀,4,6,7,9,3,8,I,I₂,II,5,I₃,
3,9,2,8,I₀,I,II,I₂,I₃,6,I₃,7,I₃,I₂,I
I,I,I,I,2,I,2,I,I,I,I,I,2,2,2,I,I,I,I,2,I,2,2,I,2,I,2,I,I,
2,2,I,2,2,2,I,2,2,2

Л и т е р а т у р а

1. Цой С., Цхай С.М. Прикладная теория графов. - Алма-Ата: Наука, 1971. - 500 с.
2. Кристофидес Н. Теория графов. Алгоритмический подход. - М.: Мир, 1978. - 432 с.
3. Altman S., Beltrami E., Rappaport S., Schoepfle G. A nonlinear programming model for household refuse collection, IEEE Trans., SMC-1, 1971, p. 289.
4. Altman S., Bhagat N., Bodin L. Extensions of the Clarke and Wright algorithm for routing garbage trucks, Presented at the 8th International TIMS Meeting, Washington D.C., 1971.
5. Beltrami E.J., Bodin L.D. Networks and vehicle routing for municipal waste collection, Proc. 10th Annual Allerton Conference on Circuit and Systems Theory, University of Illinois, 1972, p. 143.
6. Eilon S., Watson-Gandy, Christofides N. Distribution Management: Mathematical modelling and practical analysis, Griffin, London, 1971.
7. Stricker R. Public sector vehicle routing - the Chinese postman problem, Ph.D. Thesis, Electrical Eng. Dept., M.I.T., 1970.
8. Корбут А.А., Финкельштейн Ю.Ю. Дискретное программирование. - М.: Наука, 1969. - 368 с.

В. И. Исюк

САПР МАТРИЧНЫХ БИС. ОСНОВНЫЕ КОНЦЕПЦИИ

1. Введение

В настоящее время для многих специалистов, занятых проектированием электронного оборудования, характерна острая потребность в полузаказных матричных БИС. Применение матричных БИС позволяет существенно уменьшить размеры печатных плат, обеспечивая размещение большего числа функциональных узлов в расчете на единицу площади, и, следовательно, способствует созданию более эффективных устройств

в расчете на одну печатную плату. Поэтому не удивительно, что в процессе проектирования своих изделий разработчики все чаще используют матричные БИС. При этом первостепенное значение приобретают сроки оставок опытных образцов матричных БИС предприятием - изготовителем.

Существенное уменьшение срока поставок, т.е. промежутка времени от получения заказа на разработку матричной БИС до изготовления опытных образцов, можно достичь, в первую очередь, за счет резкого сокращения времени на проектирование матричной БИС, что возможно при наличии на предприятии-изготовителе высокопроизводительной системы автоматизации проектирования, обеспечивающей сквозной цикл проектирования матричных БИС от подготовки задания на проектирование до получения фотошаблонов и программ для комплексного измерительного оборудования.

2. Диалог и модульность повышает эффективность САПР матричных БИС

При обычном автоматизированном проектировании инженеры, работающие с прикладными программными средствами, сами управляют каждой отдельной стадией процесса проектирования и осуществляют их связь друг с другом: переход от разработки функциональной схемы к составлению логической, от логической схемы к тестам изделия и от тестов к разработке топологии. При этом в ходе работы программ нельзя использовать опыт разработчика. Дело в том, что обычное автоматизированное проектирование в значительной степени ориентировано на па-

кетный режим обработки и решаемые в таком режиме задачи не допускают возможности вмешательства оператора. По этой причине любые попытки инженерного творчества в ходе вычислительных процессов о т ч н о автоматизированного проектирования неосуществимы.

При диалоговом автоматизированном проектировании прикладные программные средства используются для того, чтобы освободить пользователя от тяжелой и нудной работы по управлению отдельными стадиями процесса проектирования и организацией связей между ними, но сохраняют ему возможность активного вмешательства на любом этапе процесса проектирования.

При использовании обычного автоматизированного проектирования часто приходится выполнять повторные прогоны после завершения тех или иных задач, чтобы выявить воздействие различных входных данных на конечный результат проектирования.

При диалоговом автоматизированном проектировании инженер, подготовленный к диалоговой работе, реагирует на результаты счета и запросы ЭВМ в процессе выполнения программ, принимает наиболее критические проектные решения и вводит их в ЭВМ посредством клавиатуры, меняя по своему усмотрению режимы работы программ. В результате, при диалоговом автоматизированном проектировании резко сокращается время выполнения отдельных стадий процесса проектирования, что особенно важно для САПР матричных БИС. Так, перевод электрического расчета фрагментов БИС с программы "Эл-60", работающей в пакетном режиме, в диалоговую систему "Аспект" позволило в 5 раз сократить стадию схемного моделирования БИС [1] .

Существенное сокращение времени проектирования матричных БИС может быть достигнуто при обеспечении полной взаимосвязи отдельных стадий сквозного цикла проектирования с помощью общей базы данных, используемой на каждой стадии проектирования.

Наличие общей базы данных, составляющей основу САПР матричных БИС, системы управления базой данных и единого универсального формата данных является неоспоримым достоинством любой диалоговой системы. Такая структура диалоговой САПР матричных БИС позволяет обеспечить высокий уровень модульности. Система представляется в виде набора совместимых модулей для логического моделирования, генератора тестов и т.д., связанных единым универсальным форматом и общей базой данных.

При этом предполагается возможность разработки и включения в состав САПР других модулей, определяемых проектировщиками БИС.

Процедура включения должна быть построена таким образом, чтобы

Эти модули автоматически оказались совместимыми со всеми остальными модулями диалоговой САПР матричных БИС.

3. Стадии сквозного цикла проектирования матричных БИС

В САПР матричных БИС можно выделить семь стадий, из них первые пять являются основными и определяют сквозной цикл проектирования заказных матричных БИС (таблица).

№ п/п	Стадия проектирования	Средства проектирования
1.	Проектирование функциональной и принципиальной схемы	Язык структурно-функционального описания схем, программы логического синтеза и имитационного моделирования
2.	Генерация тестов	Генераторы тестов, анализаторы тестов, трансляторы тестов для КИО
3.	Временной анализ	Программы логико-временного моделирования
4.	Проектирование топологии	Программы автоматического размещения и трассировки, графические редакторы, программы контроля и проверки на соответствие принципиальной схеме
5.	Генерация фотошаблонов	Программы генерации магнитных лент для МЭУ. Система 15 УТ-4-017
6.	Проектирование электрической схемы и технологии	Схемное моделирование, технологическое моделирование. Система "Аспект"
7.	Оформление документации	Редакторы текста, системы документирования

На первой стадии, стадии проектирования функциональной и принципиальной схемы, осуществляется ввод в ЭВМ информации о проектируемой схеме или ее функция, логический синтез и имитационное модели-

рование схемы с целью проверки выходных реакций проектируемой матричной БИС на заданные наборы входных воздействий.

Процесс ввода информации о проектируемой схеме в ЭВМ осуществляется с помощью интерактивной системы, которая формирует описание схемы в виде унифицированной графической структуры, позволяющей одновременно и автоматически модифицировать все взаимосвязанные фрагменты схемы при изменении конфигурации элементов нижнего уровня [2]. В результате организуется база данных, в которой принципиальная схема представляется в виде вложенных блоков с полным описанием связей, начиная от самого верхнего уровня - структурной схемы, и кончая нижними уровнями компонентов, вплоть до транзисторного уровня. Любые изменения на нижних уровнях отражаются на всех верхних уровнях и влекут за собой коррекцию принципиальной схемы. Аналогичным образом организована библиотека базовых функциональных элементов матричных БИС, в состав которой включены функциональные блоки как нижнего уровня, например, вентили, ключи, триггеры, так и верхних уровней - сдвиговые регистры, счетчики, мультиплексоры, ИЗУ и т.д. Число иерархических уровней, с которыми оперирует система, не ограничено.

Для задания функций цифровых схем или их блоков применяются языки таблиц переходов и выходов, граф-схем алгоритмов, микропрограмм, логических формул, входо-выходных последовательностей и некоторые другие. Логический синтез заключается в преобразовании заданных функций в реализующую их функциональную (в логическом базисе) или переключательную (на уровне транзисторов) схему [2]. Он распадается на ряд исследовательских операций, важнейшими из которых являются: синтез конечного автомата, минимизация числа состояний последнего и их кодирование, построение структурных функций переходов и выходов и их минимизация, синтез комбинационных схем и организация памяти.

Процесс имитационного моделирования осуществляется в соответствии с методологией нисходящего проектирования. Проектируемая схема описывается и отлаживается с помощью моделирования на самом высоком для нее функциональном уровне. Затем последовательно ведется ее разложение на все меньшие функциональные блоки до тех пор, пока не будет достигнут уровень базовых функциональных элементов, состоящих из отдельных транзисторов. Завершающее моделирование на этом уровне позволяет избежать многих проектных ошибок. Проектируемую схему можно моделировать в смешанном режиме. Это означает, например, что одну часть системы можно моделировать на логическом

уровне, подавая на нее сигналы с некоторого блока управления, моделируемого на уровне инструкций. Возможность смешанного моделирования значительно ускоряет решение задач моделирования схем.

Стадия генерации тестов является естественным продолжением стадии проектирования функциональной и принципиальной схем. К сожалению, ни методы генерации тест-программ, ни методы проектирования с ориентацией на тестирование не решают полностью проблему автоматизации тестирования.

В настоящее время в САПР получают все большее распространение системы тестирования, которые включают в себя набор средств, обеспечивающих генерацию и анализ тестов для проверяемой БИС и тестера, с использованием методов моделирования исправной схемы и схем с неисправностями.

„Моделирование логики и логики с ошибками является единственным практически осуществимым способом генерации тест-программ с полным охватом всех возможных ошибок для СБИС при сохранении тест-векторов в разумных пределах“ [3].

Любая система логического моделирования может проверять тесты, созданные разработчиком, однако его полнота оказывается неизвестной. Предполагается с этой целью использовать описанную ранее систему имитационного моделирования, дополненную средствами построения полного списка неисправностей и анализа полноты теста. Традиционно имитация неисправностей базируется на довольно простых моделях ошибок — одиночные контактные неисправности, тождественные 0 или 1 [4]. Было бы естественным для матричных БИС разработать библиотеку моделей неисправностей, учитывающих технологические и топологические особенности типовых функциональных элементов. Дополненная таким образом система имитационного моделирования позволит проектировщикам БИС определять, насколько полно разработанный тест обеспечивает обнаружение всех потенциальных неисправностей схемы и включать дополнительные тестовые наборы для обнаружения не выявленных ранее неисправностей. В результате формируется проверяющий тест — набор входных сигналов и правильные выходные сигналы, применяемые для проверки работоспособности проектируемой матричной БИС. Стадия генерации тестов не будет законченной без специального комплекса трансляторов, которые позволят преобразовывать проверяющий тест в тест-программы для проверки матричных БИС на любом серийно изготавливаемом комплексном испытательном оборудовании, находящемся на предприятии-изготовителе матричных БИС.

На стадии временного анализа выполняется логико-временное мо-

делирование проектируемой матричной БИС. Программы временного моделирования содействуют детальному определению временных характеристик сигналов, поиску критических составянок в схеме. Анализ временных характеристик позволяет убедиться в том, что временные характеристики проектируемой схемы удовлетворяют техническому заданию, и показывает, остаются ли сигналы во время моделирования постоянными или меняются. Все сказанное ранее по методологии имитационного моделирования справедливо и для программ логико-временного моделирования. Для задания временных задержек в программе логико-временного моделирования используются программные средства стадии проектирования электрической схемы.

Эта стадия является не основной для сквозного цикла проектирования матричных БИС, так как используется только при разборке библиотеки типовых функциональных элементов. На этой стадии применяется система автоматизированного проектирования фрагментов БИС "Аспект": Диалоговая интерактивная система "Аспект" осуществляет схемное моделирование, в результате которого выполняется расчет динамических и статических характеристик, анализ напряжений и форм сигналов в проектируемой схеме, моделирование переходных процессов и т.д. Полученные в результате схемного моделирования временные модели используются при формировании библиотеки типовых функциональных элементов матричной БИС. Заложенные в систему "Аспект" технологические модели дают проектировщикам БИС дополнительные возможности по моделированию с помощью ЭВМ различных технологических маршрутов (n-канальных, МОП, КМОП и т.д.). Технологическое и схемное моделирование системы "Аспект" связаны через единый архив данных, что позволяет результаты технологического расчета использовать в качестве исходных параметров для схемного моделирования.

Следующая стадия сквозного цикла проектирования матричных БИС — проектирование топологической структуры. Проектирование топологии матричной БИС выполняется обычно в интерактивном режиме с применением графических редакторов, средств автотрассировки и размещения компонентов. Один из типичных вариантов предлагает следующую последовательность операций: ручное размещение основных макроэлементов и прокладку наиболее критических трасс, дальнейшее автоматическое размещение и трассировку большинства оставшихся макроэлементов и, наконец, ручную трассировку элементов, которые не удалось выполнить в автоматическом режиме.

Исходя из предложенной структуры САПР матричных БИС с единой унифицированной базой данных и в соответствии с методологией нис-

ходящего моделирования, исходной информацией автоматического размещения и трассировки может служить описание схемы на транзисторном уровне, которое к данному моменту уже прошло стадии моделирования и верификации. Такой подход исключает любые теоретические ошибки при проектировании топологии. Однако в качестве дополнительной меры, гарантирующей качество матричной БИС после топологического проектирования, обязательно выполнение топологического контроля и логической проверки топологии на соответствие принципиальной схеме.

Завершающей стадией сквозного цикла проектирования матричных БИС является генерация фотомаски. Топологическая информация, полученная на стадии проектирования топологической структуры, выводится на ленту и передается на диалоговую систему машинного проектирования 15 УТ-4-017 "Кулон".

Система "Кулон", оснащенная скоростным графопостроителем "Планшет", со скоростью вычерчивания до 2 метров в секунду обеспечивает быстрое вычерчивание послойных чертежей спроектированной матричной БИС и генерацию управляющих магнитных лент для микрофотонаборных установок.

Обычно завершающей стадией любого процесса проектирования является оформление документации. Эта стадия является вспомогательной для любого этапа сквозного цикла проектирования матричных БИС и, в первую очередь, вызвана тем, что "инженеры тратят на проектные работы лишь около 30% своего рабочего времени, остальное время уходит на обычную канцелярскую работу" [5].

Для регистрации всей информации, имеющей отношение к проектируемой матричной БИС, необходимы редакторы текста и системы документирования. Редактор текста представляет собой простое программное средство, позволяющее достаточно легко редактировать текстовые файлы, вводя и исключая строки и перемещая отдельные слова до тех пор, пока в файл не будут включены все необходимые данные. Системы документирования выполняют следующий шаг этого процесса, производя форматную запись текстов в файлы и генерируя таблицы и рисунки, в том числе и графическую информацию, поясняющую определенную конструкцию БИС. Какующаяся "несерьезность" этой стадии на практике позволяет освободить инженера-проектировщика от повседневной рутинной работы, что в конечном итоге сказывается на качестве проектируемых изделий.

4. Заказчики БИС — пользователи САПР матричных БИС

В последнее время во взаимоотношениях заказчиков матричных БИС и предприятий-изготовителей наметились новые тенденции, вызванные, в первую очередь, особенностями проектирования заказных матричных БИС. Заказчики матричных БИС стремятся самостоятельно проектировать свои схемы и передавать на предприятие-изготовитель готовую конструкторскую документацию и магнитные ленты с управляющей информацией для изготовления фотошаблонов и опытных образцов матричной БИС. При этом заказчики идут по традиционному пути, связанному с применением на своем предприятии больших ЭВМ и специализированного программного обеспечения САПР матричных БИС, что эффективно только при большом числе реализуемых проектов. Основная же масса заказчиков нуждается в ограниченном числе матричных БИС. Другая негативная сторона этой тенденции заключается в том, что наличие своей собственной САПР на предприятиях-заказчиках приводит к несогласованности действий при проектировании схем, что особенно часто проявляется при тестировании и измерениях опытных образцов матричных БИС на предприятии изготовителя.

Естественный путь решения этой проблемы — предоставление заказчикам со стороны предприятия-изготовителя матричных БИС как своего базового матричного кристалла, так и своей САПР матричных БИС, ориентированной на этот кристалл. При этом программное обеспечение, предназначенное для оформления документации и первичной обработки информации о проектируемой схеме, адаптируется к мини-ЭВМ, например "Электроника 100/25", и передается заказчикам матричных БИС.

Заказчики выполняют у себя на мини-ЭВМ наиболее трудоемкие, но не требующие большой ЭВМ, этапы по формированию описания логической структуры и оформлению документации на схему своей БИС и передают посредством магнитной ленты соответствующие данные на большую ЭВМ предприятия-изготовителя. Дальнейшее проектирование матричной БИС осуществляется на большой ЭВМ силами заказчика или предприятия-изготовителя, где выполняется полное логическое моделирование и проектирование топологии в соответствии со сквозным циклом проектирования матричных БИС.

Рассмотренный подход удобен для проектирования матричных БИС, которые характеризуются средней степенью сложности до 5000 вентилей на кристалле.

Литература

1. Сидоренко В.П., Кривцов В.В., Груданов Н.Б., Гупенко В.Н. „Аспект“-система автоматизированного проектирования фрагментов МДП-ИС в реальном времени. - Электронная техника, серия 3, микроэлектроника, 1986, вып.4(120).
2. Агибалов Г.П., Бузанов В.А., Липский В.Б., Румянцев В.Ф. Логическое проектирование переключательных автоматов. - Томск: Изд-во Том. ун-та, 1983.-154 с.
3. Гнатек Ю. Объединение систем автоматизации проектирования и автоматизации тестирования для решения проблемы контроля СВИС. - Электроника, 1984, № 8, с. 33.
4. Чжен Г., Мэнинг Е., Метц Г. Диагностика отказов цифровых вычислительных систем. - М.: Мир, 1972. - 230 с.
5. Ованчук С. Автоматизированные рабочие места. - Электроника, 1984, № 10, с.35.

Рефераты на опубликованные статьи

УДК 519.7

Агибадов Г.П. Функциональные системы на полурешетках. - В кн.: Алгоритмы решения задач дискретной математики, вып.2.- Томск: Изд-во Том. ун-та, 1987, с.3-39.

Излагаются основные понятия теории функциональных систем на полурешетках, развиваемой с целью адекватного описания функционирования цифровых устройств в динамике - при смене входных воздействий.

Библ. 14, табл.4, ил. 3.

УДК 519.7

Дрягин Ю.С. Некоторые условия существования декомпозиции автоматов. - В кн.: Алгоритмы решения задач дискретной математики, вып.2. - Томск: Изд-во Том. ун-та, 1987, с.40-52.

Рассматривается обобщение известных понятий сохраняемых конечным автоматом разбиений и покрытий и в этих терминах дана характеристика необходимых и достаточных условий существования изоморфного и гомоморфного представления автомата произвольной сетью, в том числе и из автоматов, каждый из которых содержит меньшее число состояний, чем исходный автомат.

Библ. 2, ил.1.

УДК 519.7

Павлов В.Л. Синтез комбинационных схем в произвольном базисе. - В кн.: Алгоритмы решения задач дискретной математики, вып.2. - Томск: Изд-во Том. ун-та, 1987, с.53-58.

Излагается декомпозиционный метод синтеза комбинационных схем в произвольном базисе с параметрическим описанием компонент разложения. Метод может быть использован при синтезе управляющих схем на базе матричных БИС.

Библ.5, табл.3.

УДК 519.7

Панкратова И.А. Синтез комбинационных функциональных схем в многозначной логике. - В кн.: Алгоритмы решения задач дискретной математики, вып.2. - Томск: Изд-во Том. ун-та, 1987, с.59-64.

Описываются метод синтеза комбинационных функциональных схем в многозначной логике и программы синтеза таких схем из элементов ИНЕ в 3-значной логике и из элементов комплементарных ИНЕ, ИЛИНЕ и проводного ИЛИ в 6-значной логике.

Библ.3, табл.2.

УДК 681.3

Липский В.Б. Система логического синтеза интервальных автоматов. - СИНТА. - В кн.: Алгоритмы решения задач дискретной математики, вып.2. - Томск: Изд-во Том. ун-та, 1987, с.65-72.

Приводится описание диалоговой системы логического синтеза схем из элементов И, ИНЕ, ИЛИНЕ, реализующих заданные интервальные автоматы, представляющие алгоритмы их синхронного или асинхронного функционирования.

Библ.1, табл.1, ил.2.

УДК 681.3

Юфит Я.Г. Синтез микропрограммных автоматов на базе микропроцессоров. - В кн.: Алгоритмы решения задач дискретной математики, вып.2. - Томск: Изд-во Том. ун-та, 1987, с.73-81.

Предлагается метод реализации дискретных управляющих устройств, заданных микропрограммным описанием, на базе произвольного микропроцессорного набора.

Библ. 4.

УДК 519.682.1;681.142.2

Левашников А.А. Синтаксический контроль схем на языке СЛОГ. - В кн.: Алгоритмы решения задач дискретной математики, вып.2. - Томск: Изд-во Том. ун-та, 1987, с.81-89.

Предлагаются формальное описание языка СЛОГ и метод синтаксического контроля схем, заданных на этом языке.

Библ. 5, табл.8.

УДК 519.7

Верёвкина И.Е. Имитационное моделирование схем из функциональных элементов. - В кн.: Алгоритмы решения задач дискретной математики, вып.2. - Томск: Изд-во Том. ун-та, 1987, с.90-97.

Описываются алгоритм и программа имитационного моделирования схем с иерархической структурой, построенных из функциональных элементов в многозначной логике.

Библ. 2, табл.5.

УДК 681.3

Николаева Л.А. Имитационное моделирование микропроцессорных систем. - В кн.: Алгоритмы решения задач дискретной математики, вып.2. - Томск: Изд-во Том. ун-та, 1987, с.98-106.

Сообщается об автоматизированной системе разработки имитаторов микропроцессорных систем, предоставляющей пользователю язык описания МПС на уровне систем команд, расширяющийся набор программных моделей операций, выполняемых в МПС, и средства автоматической генерации имитатора конкретной МПС по ее программной модели функционирования.

Библ.2, ил.1.

УДК 519.7

Матросова А.Д., Реморенко Г.А. Уточнение результатов моделирования в алфавите С. - В кн.: Алгоритмы решения задач дискретной математики, вып.2. - Томск: Изд-во Том. ун-та, 1987, с.107-117.

Предлагается процедура уточнения результатов моделирования в алфавите С; основанная на анализе редуцированной эквивалентной нормальной формы. Рассматриваются возможности использования процедуры в схемах с памятью.

Библ. 5.

УДК 519.7

Байда В.Д. К построению множества максимальных устойчивых состояний асинхронного автомата. - В кн.: Алгоритмы решения задач дискретной математики, вып.2. - Томск: Изд-во Том. ун-та, 1987, с.118-122.

Предлагается алгоритм построения множества максимальных устойчивых состояний асинхронного автомата, требующий меньшего объема вычислений по сравнению с известным ранее алгоритмом.

Библ. 3.

УДК 519.7

Черкашин Ю.Д. Минимизация числа обратных связей в функциональных схемах. - В кн.: Алгоритмы решения задач дискретной математики, вып.2. - Томск: Изд-во Том. ун-та, 1987, с.123-126.

Предлагается приближенный алгоритм минимизации числа точек разрыва обратных связей в функциональной схеме, представленной ориентированным графом. Приводятся результаты статистических испытаний на ЭВМ этого алгоритма.

Библ. 2, табл. 1.

УДК 519.7

Бузанов В.А. О вычислении неисправностей в структурном автомате. - В кн.: Алгоритмы решения задач дискретной математики, вып.2. - Томск: Изд-во Том. ун-та, 1987, с.127-129.

Предлагается метод вычисления неисправностей автомата в многозначном структурном алфавите, обнаруживаемых заданной входной последовательностью.

Библ. 2.

УДК 519.7

Евтушенко Н.В. О принадлежности последовательности множеству контрольных последовательностей автомата. - В кн.: Алгоритмы решения задач дискретной математики, вып.2. - Томск: Изд-во Том. ун-та, 1987, с.130-133.

Формулируются необходимые и достаточные условия принадлежности произвольной входной последовательности инициального автомата множеству его контрольных последовательностей.

УДК 519.7

Агибалов Г.П. Задача выбора и декомпозиция схем с размещением и трассировкой в компоновочном пространстве. - В кн.: Алгоритмы решения задач дискретной математики, вып.2. - Томск: Изд-во Том. ун-та, 1987, с.134-153.

Поставлена общая задача физического проектирования схем, включающая декомпозиционный, размещенческий и трассировочный аспекты, предложено решение этой задачи как некоторой задачи выбора и сформулированы алгоритмы решения произвольной задачи выбора. Последний представлен как алгоритм сокращенного обхода дерева поиска по глубине в пространстве систем представителей, содержащий ряд параметров, с помощью которых он может быть настроен на любую конкретную задачу выбора. Указаны выражения для параметров, при которых данный алгоритм решает поставленную задачу физического проектирования.

Библ. 7, табл. I, ил. I.

УДК 681.3

Елисеева Н.А. Алгоритм трассировки электрических цепей. - В кн.: Алгоритмы решения задач дискретной математики. - Томск: Изд-во Том. ун-та, 1967, с.154-160.

Предлагается алгоритм соединения произвольного числа фрагментов электрической цепи минимальной совокупностью связей. Алгоритм может быть использован в системах автоматического проектирования топологии БИС и конструирования печатных плат.

Библ. 2.

УДК 519.7

Дубков Ю.И. Сокращение перебора в задачах комбинаторики. - В кн.: Алгоритмы решения задач дискретной математики. - Томск: Изд-во Том. ун-та, 1987, с.161-164.

Рассматривается задача, частными случаями которой являются известные задачи о покрытиях, о назначениях, коммивояжёра и др. Предлагается параметрический метод поиска ее решения. Формулируется проблема выбора оптимального значения параметра конструируемых на базе этого метода алгоритмов.

Библ. 4.

УДК 681.3

Беляев В.А. Алгоритмы раскраски графа методом сокращенного обхода дерева поиска. - В кн.: Алгоритмы решения задач дискретной математики, вып.2. - Томск: Изд-во Том. ун-та, 1987, с.165-172.

Предлагается улучшенный алгоритм раскраски графа методом сокращенного обхода дерева поиска. Улучшение достигается за счет использования свойств полных разделяющих подграфов и более эффективного алгоритма порождения максимальных полных подграфов.

Библ. 7, ил. 3.

УДК 519.14

Ильин В.А., Катаева С.С. Полный обход графа с ограничениями на структуру маршрута. - В кн.: Алгоритмы решения задач дискретной математики, вып.2. - Томск: Изд-во Том. ун-та, 1987, с.173-185.

Предлагаются алгоритмы поиска минимальных полных обходов графов с ограничениями на структуру маршрута. В основе алгоритма лежит принцип целенаправленного поиска по значениям оценочных функций частично построенных маршрутов с последующим улучшением частичных маршрутов по алгоритмам поиска в глубину.

Библ. 8, табл. 4, ил. 1.

УДК 681.3

Исюк В.И. САПР матричных БИС. Основные концепции. - В кн.: Алгоритмы решения задач дискретной математики, вып.2. - Томск: Изд-во Том. ун-та, 1987, с.186-194.

Обоснованы основные концепции построения САПР матричных БИС: интерактивность, модульность и др. Рассмотрены стадии сквозного цикла проектирования (от разработки задания до выдачи документов) и сформулированы требования к ним.

Библ. 5, табл. 1.

СОДЕРЖАНИЕ

Г.П.Агибалов. Функциональные системы на полурешетках	3
Ю.С.Дрягин. Некоторые условия существования декомпозиции автоматов	40
В.Л.Павлов. Синтез комбинационных схем в произвольном базисе	53
И.А.Панкратова. Синтез комбинационных функциональных схем в многозначной логике	59
В.Б.Липский. Система логического синтеза интервальных автоматов – СИНТА	65
Я.Г.Юфит. Синтез микропрограммных автоматов на базе микропроцессоров	73
А.А.Левашников. Синтаксический контроль схем на языке СЛОГ	82
И.Е.Веревкина. Имитационное моделирование схем из функциональных элементов	90
Л.А.Николаева. Имитационное моделирование микропроцессорных систем	98
А.Ю.Матросова, Г.А.Реморенко. Уточнение результатов моделирования в алфавите С	107
В.Д.Байда. К построению множества максимальных устойчивых состояний асинхронного автомата	118
Ю.Д.Черкашин. Минимизация числа обратных связей в функциональных схемах	123
В.А.Бузанов. О вычислении неисправностей в структурном автомате	127
Н.В.Евтушенко. О принадлежности последовательности множеству контрольных последовательностей автомата	130
Г.П.Агибалов. Задача выбора и декомпозиция схем с размещением и трассировкой в компоновочном пространстве . . .	134

Н.А.Елисеева. Алгоритм трассировки электрических цепей	I54
Ю.И.Дубков. Сокращение перебора в задачах комбинаторики	I61
В.А.Белнев. Алгоритм раскраски графа методом сокращенного обхода дерева поиска	I65
В.А.Ильин, С.С.Катаева. Полный обход графа с ограничениями на структуру	I73
В.И.Исх. САПР матричных БИС. Основные концепции	I86
Рефераты на опубликованные статьи	I95

АЛГОРИТМЫ РЕШЕНИЯ
ЗАДАЧ ДИСКРЕТНОЙ МАТЕМАТИКИ

Выпуск 2

Редактор Л.И.Дюканова

ИБ 1901. Подписано к печати 17.11.1986г. КЗ 06313
Формат 60х84 1/16. Бумага типографская №3. Печать офсетная.
Печ. л. 12,75. Усл. печ. л. 11,85. Уч.-изд. л. 10,5.
Тираж 500 экз. Заказ 97 Цена 1 р. 60 к.

Издательство ТТУ, 634029, Томск, ул. Никитина, 4.
Ротапринт ТТУ, 634029, Томск, ул. Никитина, 4

1-638118
Томский госуниверситет 1878



Научная библиотека 00605452