ОДНОРОДНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ ИЗ МИКРО-ЭВМ
(Вычислительные системы)
1983 год Выпуск 97

УДК 681.322

АНАЛИЗ АСИНХРОННОЙ ИНТЕРПРЕТАЦИИ ПАРАЛЛЕЛЬНЫХ МИКРОПРОГРАММ

С.М.Ачасова

Работа является продолжением исследований асинхронных интерпретаций параллельных микропрограмм с точки зрения их детерминированности, начатых в [I]. Здесь описаны классы параллельных микропрограмм, порождающих детерминированные асинхронные вычисления, моделирующие такие вычисления сети Петри с их особенностями в каждом классе, сформулированы необходимые и достаточные условия детерминированности параллельных микропрограмм в случае их асинхронного выполнения, и в заключение записан алгоритм проверки параллельной микропрограммы на детерминированность.

I. Используемые понятия

Все понятия и определения, данные в этом разделе, можно найти в более формализованном или развернутом виде в работах [I-3]. Здесь они даются кратко и только для того, чтобы представить язык, на котором ведется изложение материала.

Под параллельной микропрограммой понимают систему параллельных подстановок (микрокоманд) [2]. Свойство параллельности такой системы проявляется в том, что микрокоманды могут выполняться в произвольном порядке. Объектом применения параллельных микропрограмм является конечное множество - слово $W = \{(a_i, m_i)\}$, $i = \overline{0, p}$, где (a_i, m_i) - клетка с именем m_i и в состоянии a_i, $a_i \in A$, $m_i \in M$, $W \subset A \times M$. В слове W не может быть двух клеток (a_i, m_i), (a_j, m_j) с одинаковыми именами, т.е. $m_i \neq m_j$. Слово W можно представить двумя проекциями - векторами $Pr_1(W) = (a_0, \ldots, a_p)$, $Pr_2(W) = (m_0, \ldots, m_p)$.

Множество слов $S = \{W_j\}$, имеющих одинаковые первые проекции, а вторые вида $Pr_2(W_j) = \{\varphi_0(m_j),\ldots,\varphi_p(m_j)\}$, называется конфигурацией и обозначается $S(m) = \{(a_0,\varphi_0(m)),\ldots,(a_p,\varphi_p(m))\}$. В конфигурации все функции $\varphi_i(m)$ попарно различны. Задавая m, можно выписать конкретное слово, принадлежащее конфигурации. Удобно одну из именующих функций $\varphi_i(m)$ конфигурации $S(m)$ иметь равной m, пусть $\varphi_0(m) = m$.

Выражение вида $\theta: S_1 * S_2 \to S_3$ называется подстановкой или микрокомандой. Левая часть микрокоманды - конфигурация $S_1 * S_2$ составлена из двух конфигураций: базы $S_1 = \{(a_0,m),(a_1,\varphi_1(m)),\ldots,(a_p,\varphi_p(m))\}$ и контекста $S_2 = \{(b_0,\psi_0(m)),(b_1,\psi_1(m)),\ldots,(b_q,\psi_q(m))\}$, в которых все функции φ_i,ψ_j попарно различны, $S_1 * S_2 = \{(a_0,m),\ldots,(a_p,\varphi_p(m)),(b_0,\psi_0(m)),\ldots,(b_q,\psi_q(m))\}$. В микрокоманде вторые проекции правой части S_3 и базы S_1 совпадают, т.е. $Pr_2(S_1) = Pr_2(S_3)$. Микрокоманда θ применима к слову W, если в нем найдется хотя бы одна клетка (a_i,m_i) такая, что $S_1(m_i) * S_2(m_i) \subseteq W$. Применение микрокоманды θ к слову W состоит в изъятии из него подслов $W_i = S_1(m_i)$, $i = i_1,\ldots,i_k$, и присоединение к нему подслов $W_i' = S_3(m_i)$. Контекст S_2 входит в условие применимости микрокоманды, но в результате выполнения микрокоманды сам не изменяется. Применение микрокоманды θ_i к слову W для конкретной клетки с именем m_j называется микрооперацией и обозначается через θ_{im_j}.

Поясним все введенные понятия на примере.

ПРИМЕР I. Пусть в алфавите $A = \{a,b,c,d\}$ и множестве имен $M = \{1,7\}$ имеет слово $W = \{(a,1),(b,2),(c,3),(d,4),(a,5),(b,6),(c,7)\}$. Его проекции равны $Pr_1(W) = (a,b,c,d,a,b,c)$, $Pr_2(W) = (1,2,3,4,5,6,7)$. Применим к слову W микрокоманду θ: $(b,x) * \{(a,x-1),(c,x+1)\} \to (d,x)$. В конфигурацию $\{(b,x),(a,x-1),(c,x+1)\}$ входят два подслова слова W: $\{(a,1),(b,2),(c,3)\}$ и $\{(a,5),(b,6),(c,7)\}$. Применение микрокоманды θ к W состоит в выполнении двух микроопераций θ_2 и θ_6, т.е. микрокоманда θ выполняется для клеток с именами 2 и 6. Результатом применения θ к W является слово $W' = \{(a,1),(d,2),(c,3),(d,4),(a,5),(d,6),(c,7)\}$

* * *

Параллельной микропрограммой является множество микрокоманд $\Psi = \{\theta_i\}$, $i = \overline{1,v}$, в котором нет двух микрокоманд с одинаковыми левыми частями. Микропрограмма Ψ применима к слову W, если к нему применима хотя бы одна микрокоманда $\theta_i \in \Psi$.

Различаются два способа преобразования слова с помощью параллельной микропрограммы или две интерпретации микропрограммы: синхронная и асинхронная.

С и н х р о н н а я интерпретация $\bar{\Psi}$ параллельной микропрограммы Ψ есть следующая итерационная процедура. Пусть слово W^{i-1} есть результат применения к W микропрограммы Ψ i-1 раз. Если ни одна микрокоманда $\theta_l \in \Psi$ не применима к W^{i-1}, то слово W^{i-1} является результатом применения Ψ к W. Если же множество применимых к W^{i-1} микрокоманд непусто и равно $\{\theta_{l_1},\ldots,\theta_{l_r}\}$, то результатом i-й итерации является слово W^i, полученное из W^{i-1} путем выполнения всех микроопераций $\{\theta_{l_1 m_{11}},\ldots,\theta_{l_1 m_{1k_1}},\theta_{l_2 m_{21}},\ldots,\theta_{l_2 m_{2k_2}},\ldots,\theta_{l_r m_{rk_r}}\}$ над словом W^{i-1}, где $\{m_{j1},\ldots,m_{jk_j}\}$ - множество имен тех клеток, относительно которых выполняется микрокоманда θ_{l_j}.

А с и н х р о н н а я интерпретация $\bar{\Psi}$ параллельной микропрограммы Ψ также является итерационной процедурой и состоит в следующем. Пусть слово W^{i-1} есть результат применения Ψ к W i-1 раз. Если Ψ не применима к W^{i-1}, то процесс преобразования W закончился словом W^{i-1}. Если множество применимых микрокоманд непусто и равно $\{\theta_{l_1},\ldots,\theta_{l_r}\}$, то результат i-й итерации выбирается из множества слов $\{W^{i_j}\}$, каждое из которых получено из слова W^{i-1} путем применения к нему только о д н о й микрооперации из множества $\{\theta_{l_1 m_{11}},\ldots,\theta_{l_1 m_{1k_1}},\ldots,\theta_{l_r m_{r1}},\ldots,\theta_{l_r m_{rk_r}}\}$. Мощность множества $\{W^{i_j}\}$ равна $\sum_{j=1}^{r} k_j$.

Как в синхронной, так и в асинхронной интерпретациях слова (i-1)-го и i-го итерационных шагов находятся в отношении следования. В синхронном случае слово W^i следует за словом W^{i-1}, если W^i получено из W^{i-1} путем выполнения всех применимых к W^{i-1} микроопераций. В асинхронном случае слово W^{i_j} следует за словом W^{i-1}, если W^{i_j} получено из W^{i-1} путем выполнения для одной из клеток слова W^{i-1} некоторой микрооперации из числа применимых к нему. Совокупность слов, полученных из W путем применения к W параллельной микропрограммы Ψ (синхронным или асинхронным способом), вместе с отношением следования слов называется вычислением (синхронным $\bar{\Psi}_W$ или асинхронным $\bar{\bar{\Psi}}_W$).

Вычисление изображают ориентированным графом, вершинам которого поставлены в соответствие слова; каждые два слова, связанные отношением следования, соединены дугой, подходящим образом ориентированной. Граф имеет одну начальную вершину (вершина без входящей в нее дуги) и множество конечных (вершины без исходящих дуг). Начальной вершине соответствует исходное слово W, конечной - слово - результат. Граф, представляющий конечное вычисление, не имеет циклов. Путь в графе от начальной вершины к конечной соответствует одной реализации вычисления. Граф, представляющий синхронное вычисление Ψ_W, является цепью. Действительно, такое вычисление имеет единственную реализацию, так как каждый шаг вычисления дает единственный результат. Асинхронное вычисление Ψ_W может иметь множество реализаций и не единственный конечный результат (а граф - не единственную конечную вершину) из-за множественности результата каждого шага вычисления. Проиллюстрируем на примере вновь введенную группу понятий.

ПРИМЕР 2. Пусть в алфавите A = {a,b,c,d,h} и множестве имен M = $\overline{1,5}$ имеется слово W= {(c,1),(d,2),(h,3),(a,4),(b,5)}. Применим к W микропрограмму

$$\Psi = \begin{cases} \theta_1\colon (b,x) * (a,x-1) \to (d,x), \\ \theta_2\colon \{(c,x),(d,x+1)\} \to \{(h,x),(a,x+1)\}, \\ \theta_3\colon (h,x) * (a,x+1) \to (a,x), \end{cases}$$

микрокоманды которой можно представить в виде следующих геометрических образов: $\boxed{a}\boxed{b} \to \boxed{\ }\boxed{d}$, $\boxed{c}\boxed{d} \to \boxed{h}\boxed{a}$, $\boxed{h}\boxed{a} \to \boxed{a}\boxed{\ }$.

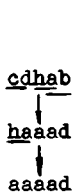

Рис.I

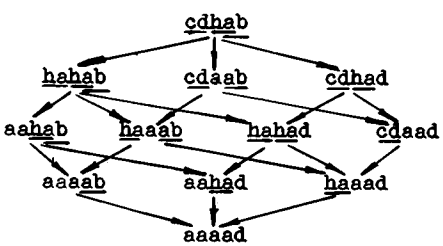

Рис. 2

Графы вычислений синхронного $\bar{\Psi}_W$ и асинхронного $\tilde{\Psi}_W$ представлены соответственно на рис.I и 2 (вершины графов помечены геометрическими образами соответствующих слов, в словах подчеркнуты левые части применимых подстановок).

2. Классы параллельных микропрограмм

Вычисление называется д е т е р м и н и р о в а н н ы м, если все его реализации сходятся к одному и тому же результату и, следовательно, граф такого вычисления имеет единственную конечную вершину. Синхронная (асинхронная) интерпретация параллельной микропрограммы является д е т е р м и н и р о в а н н о й, если синхронное (асинхронное) вычисление $\bar{\Psi}_W(\tilde{\Psi}_W)$ детерминировано для любого слова $W \subset A \times M$.

Понятно, что асинхронное вычисление может оказаться недетерминированным из-за множественности результата на каждом шаге. Вопрос о недетерминированности синхронных вычислений рассмотрен в работе [3]. Ситуация, в которой возникает недетерминированность, состоит в следующем. На некотором шаге вычисления $\bar{\Psi}_W$ клетка (a,x) оказывается в поле действия двух микроопераций, одна из которых стремится (a,x) заменить на (b,x), другая - на (c,x). Возникает противоречие, и результат этого шага вычисления оказывается неопределенным, а само вычисление недетерминированным. Микропрограмма, способная породить такое вычисление, называется противоречивой. В [3] сформулированы необходимые и достаточные условия непротиворечивости для некоторого класса параллельных микропрограмм. В этот класс входят микропрограммы, в которых для записи конфигураций используются только функции сдвига на константу: $\varphi(m) = m \pm C$. Это достаточно широкий класс параллельных микропрограмм для практических вычислений. В данной работе исследование особенностей асинхронных интерпретаций выполняется именно для такого класса микропрограмм.

Следующий пример иллюстрирует явление противоречивости, возникающее при синхронной интерпретации микропрограмм. Заметим, что при асинхронной интерпретации это явление не возникает, поскольку в одной реализации асинхронного вычисления на каждом шаге любая клетка испытывает на себе воздействие только одной микрооперации, независимо от того, в зонах действия скольких микроопераций эта клетка находится.

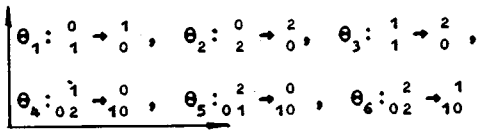

Рис. 3

ПРИМЕР 3. К слову $W = \{(a,1),(b,2),(c,3)\}$ применим микропрограмму

$$\Psi = \begin{cases} \theta_1\colon (b,x) * (a,x-1) \to (d,x), \\ \theta_2\colon (b,x) * (c,x+1) \to (e,x) \end{cases}$$

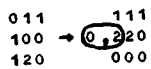

Рис.4

синхронным способом. Результатом выполнения на первом же шаге обеих микрокоманд является множество $\{(a,1),(d,2),(e,2),(c,3)\}$, которое не является клеточным, так как в нем есть две клетки с одинаковыми именами и разными состояниями $(d,2)$ и $(c,2)$. Следовательно, микропрограмма Ψ противоречива.

* * *

Заметим, что классы параллельных микропрограмм, порождающих детерминированные асинхронные и детерминированные синхронные вычисления, не совпадают и не являются подклассами один другого. Покажем это на контрпримерах для обратных утверждений.

ПРИМЕР 4. Микропрограмма сложения n троичных чисел, написанная по аналогии с микропрограммой сложения n двоичных чисел [2, с. 23] с той лишь разницей, что в ее микрокомандах отсутствуют контексты, состоит из шести микрокоманд, геометрические образы которых в целочисленной двумерной решетке изображены на рис. 3. Эта микропрограмма при синхронной интерпретации противоречива. Пример ситуации, в ко-

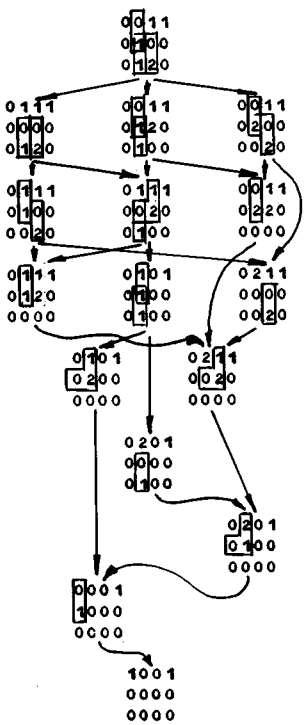

Рис. 5

торой проявляется ее противоречивость, дан на рис. 4. Асинхронная интерпретация данной микропрограммы является детерминированной. Граф сложения трех четырехразрядных чисел изображен на рис. 5.

• • •

ПРИМЕР 5. Микропрограмму

Рис. 6

$$\Psi = \begin{cases} \theta_1: \boxed{b}\boxed{g} \to \boxed{a}\boxed{c}, \\ \theta_2: \boxed{b}\boxed{c} \to \boxed{\ }\boxed{d}, \\ \theta_3: \boxed{a}\boxed{c}\boxed{b} \to \boxed{\ }\boxed{q}\boxed{a}, \\ \theta_4: \boxed{b}\boxed{d} \to \boxed{a}\boxed{q} \end{cases}$$

применим к слову $W = \boxed{b}\boxed{g}\boxed{d}\boxed{e}$.

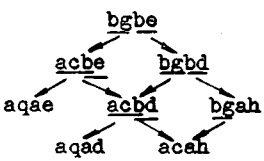

Рис. 7

В синхронном случае получим детерминированное вычисление, граф которого изображен на рис.6. Асинхронное вычисление является недетерминированным, его граф изображен на рис. 7.

• • •

Рассмотрим подклассы параллельных микропрограмм, порождающих детерминированные асинхронные вычисления. Существенным моментом при классификации микропрограмм является проверка микрокоманд на пересечение. Определим понятие пересечения конфигураций. Две конфигурации $S_1 = \{(a_0,m),(a_1,\varphi_1(m)),\ldots,(a_p,\varphi_p(m))\}$ и $S_2 = \{(b_0,m),(b_1,\phi_1(m)),\ldots,(b_q,\phi_q(m))\}$ пересекаются, если найдется такое k, что в конфигурациях S_1 и $S_2' = \{(b_0, m+k),(b_1,\phi_1(m)+k),\ldots,(b_q,\phi_q(m)+k)\}$ есть одинаковые клетки, т.е. $S_1 \cap S_2' \neq \emptyset$. В общем случае, когда имена $m_i \in M$ интерпретируются узлами n-мерной целочисленной решетки и представляются n-разрядными векторами (совокупностью координат) $m_i = (m_i^1,\ldots,m_i^n)$, k также является n-разрядным вектором, $k = (k^1,\ldots,k^n)$, k^j – целые числа.

• ПРИМЕР 6. Даны две конфигурации

$$S_1 = \{(a,(x,y)),(b,(x+1,y)),(c,(x,y+1))\},$$

$$S_2 = \{(b,(x,y)),(c,(x+1,y)),(a,(x+1,y+1)),(b,(x,y+1))\}$$

и константы сдвига $k^X = 1$, $k^Y = -1$, $k = (1,-1)$. Сдвинем на k име -

нующие функции второй конфигурации, получим $S_2' = \{(b,(x+1, y-1)), (c,(x+2, y-1)),(a,(x+2,y)), (b,(x+1,y))\}$. В S_1 и S_2' есть одна общая клетка $(b,(x+1,y))$, значит, $S_1 \cap S_2' = (b,(x+1, y))$. Но это не единственный вариант пересечения данных конфигураций:

при $k = (1,0)$ $S_2'' = \{(b,(x+1,y)),(c,(x+2,y)),(a,(x+2,y+1)),$
$(b,(x+1,y+1))\},$

$S_1 \cap S_2'' = (b,(x+1,y)),$

при $k = (-1,1)$ $S_2''' = \{(b,(x-1,y+1)),(c,(x,y+1)),(a,(x,y+2)),$
$(b,(x-1\ y+2))\},$

$S_1 \cap S_2''' = (c,(x,y+1)),$

при $k = (-1,-1)$ $S_2^{IV} = \{(b,(x-1,y-1)),(c,(x,y-1)),(a,(x,y)),$
$(b,(x-1,y))\},$

$S_1 \cap S_2^{IV} = (a,(x,y)).$

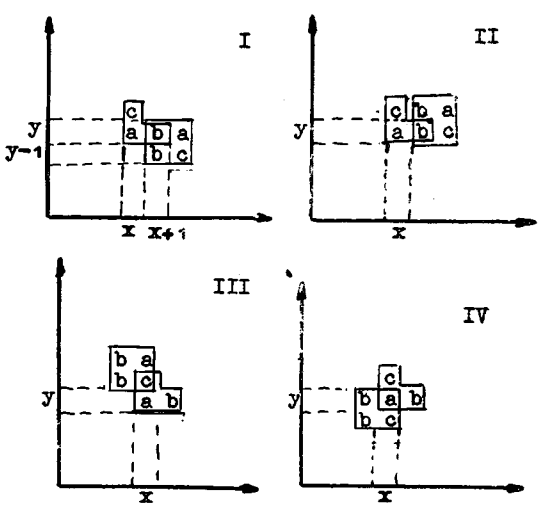

Рис. 8

Геометрические образы всех пересечений изображены на рис.8.

***

Будем говорить, что две микрокоманды Θ_i и Θ_j пересекаются, если пересечение их левых частей непусто, т.е. $S_{i1} * S_{i2} \cap S'_{j1} * S'_{j2} \neq \emptyset$.

Разобьем совокупность возможных вариантов пересечения двух микрокоманд Θ_i: $S_{i1} * S_{i2} \to S_{i3}$ и Θ_j: $S_{j1} * S_{j2} \to S_{j3}$ на три группы. К первой группе отнесем пересечение по базовым

конфигурациям, $S_{i1} \cap S'_{j1} = \emptyset$, причем такое, что множество $S_{i3} * S'_{j3}$ не является конфигурацией (в нем есть по крайней мере две клетки с одинаковыми именами и различными состояниями). Такое пересечение назовем п р о т и в о р е ч и в ы м, так как микропрограмма, в которой есть таким образом пересекающиеся микрокоманды, является противоречивой. Пример противоречивого пересечения микрокоманд приведен на рис.4. Ко второй группе отнесем два варианта пересечения. Во-первых, это - непротиворечивое пересечение по базам, причем такое, что пересечению принадлежит только часть базовой конфигурации, по крайней мере одной из пересекающихся микрокоманд, $S_{i1} \cap S_{j1} \subset S_{i1}(S'_{j1})$. В примере 5 таким образом пересекаются микрокоманды Θ_1 и Θ_3. Во-вторых, это - пересечение базы одной микрокоманды с контекстом другой, $S_{i1} * S_{i2} \cap S'_{j1} * S'_{j2} = S_{i1} \cap S'_{j2}(S_{i2} \cap S'_{j1})$. В примере 5 таким образом пересекаются микрокоманды Θ_2 и Θ_3. Пересечения второй группы будем называть о п а с н ы м и. Это название обусловлено тем, что микропрограмма, содержащая таким образом пересекающиеся микрокоманды, может порождать недетерминированные вычисления. Причиной недетерминированности является нарушение условий применимости одной из пересекающихся микрокоманд в случае выполнения другой, что может привести к тому, что процесс вычисления, разветвившись, не придет к единственному конечному результату. Пример 5 иллюстрирует такую ситуацию.

И наконец, определим пересечения третьей группы. Во-первых, это - непротиворечивое пересечение по полным базам, т.е. пересечение микрокоманд, имеющих одинаковые первые проекции базовых конфигураций, $S_{i1} * S_{i2} \cap S'_{j1} * S'_{j2} = S_{i1} = S'_{j1}$. Во-вторых, это - пересечение по контекстам, $S_{i1} * S_{i2} \cap S'_{j1} * S'_{j2} = S_{i2} \cap S'_{j2}$. Такие пересечения будем называть б е з о п а с н ы м и в смысле возникновения недетерминированности вычисления. Таким образом, пересекающиеся микрокоманды либо не нарушают условий применимости одной в случае выполнения другой (пересечение по контекстам), либо одинаковы по воздействию на клеточное множество (пересечение по полным базовым конфигурациям). В последнем случае микрокоманды изменяют состояния одних и тех же клеток и при этом одинаковым образом, так что после выполнения одной микрокоманды выполнение другой уже не требуется. Пример пересечения по контекстам дают микрокоманды Θ_1 и Θ_3 в примере 2. Непротиворечивое пересечение по полным базам имеют микрокоманды

$$\theta_1\colon \{(a,x),(b,x+1)\} * (c,x+2) \to \{(d,x),(e,x+1)\},$$
$$\theta_2\colon \{(a,x-1),(b,x)\} * (h,x-2) \to \{(d,x-1),(e,x)\}$$

при выполнении сдвига $k = I$ для θ_2 ($S'_{21} * S'_{22} = \{(a,x),(b,x+1)\} * (h,x-1)$).

Заметим, что о безопасных пересечениях микрокоманд θ_i и θ_j имеет смысл говорить только в том случае, если эти микрокоманды ни при каких сдвигах не могут пересечься опасно или противоречиво.

Обозначим через $\tilde{D}$ класс микропрограмм, порождающих только детерминированные асинхронные вычисления (в отличие от $\bar{D}$, обозначающей класс микрокоманд, дающих только детерминированные синхронные вычисления). Обратимся к подклассам класса $\tilde{D}$. Рассмотрим два подкласса - подкласс Q однозначных микропрограмм и вложенный в Q подкласс Y устойчивых микропрограмм.

Для описания подкласса Q используется понятие истории клетки [I]. В каждой реализации асинхронного вычисления можно выделить последовательность клеток, имеющих одно и то же имя. В этой последовательности могут оказаться рядом одинаковые клетки (т.е. имеющие не только одно и то же имя, но и одинаковые состояния). Если из последовательности удалить все стоящие рядом одинаковые клетки, кроме одной, и выписать соответствующую последовательность состояний, то такую последовательность можно назвать историей клетки (как это сделано в [I]). История клетки с именем x обозначается через $h(x)$.

ПРИМЕР 7. Выпишем последовательность клеток с именем 2 в реализации bgbe → acbe → acbd → aqad асинхронного вычисления, данного в примере 5. Получим $(g,2),(c,2),(c,2),(q,2)$. Удалим одну клетку $(c,2)$ и запишем историю $h(2) = g,c,q$.

• • •

Асинхронное вычисление называется о д н о з н а ч н ы м, если каждая клетка имеет одну и ту же историю во всех реализациях этого вычисления. (Очевидно, что такое вычисление детерминировано, так как при одинаковых историях одноименных клеток все реализации вычисления завершаются одним и тем же результатом). Пример однозначного вычисления дан на рис.2, здесь в любой реализации клетки имеют следующие истории: $h(1) = c,h,a$, $h(2) = d,a$, $h(3) = h,a$, $h(4) = a$, $h(5) = b,d$.

Параллельная микропрограмма, которая для любого исходного слова порождает однозначное асинхронное вычисление, называется о д н о з н а ч н о й. Очевидно, что классу однозначных не могут принадлежать противоречивые микропрограммы, поскольку любые две реализации асинхронного вычисления, в каждой из которых выполняется одна из двух противоречиво пересекающихся микрокоманд, имеют разные истории тех клеток, относительно которых проявилось противоречие. Напомним, что противоречивые микропрограммы могут порождать как недетерминированные, так и детерминированные асинхронные вычисления, это верно и для микропрограмм, в которых имеются опасные пересечения, однако в отличие от противоречивых опасные микропрограммы могут быть однозначными. Вот подходящий

ПРИМЕР 8. В микропрограмме

$$\Psi = \begin{cases} \theta_1: \boxed{b}\boxed{c} \to \boxed{\ }\boxed{d}, \\ \theta_2: \boxed{c}\boxed{d}\boxed{e} \to \boxed{\ }\boxed{h}\boxed{k}, \\ \theta_3: \boxed{d}\boxed{d} \to \boxed{\ }\boxed{h}, \\ \theta_4: \boxed{h}\boxed{e} \to \boxed{\ }\boxed{k} \end{cases}$$

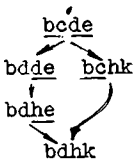

Рис. 9

имеются опасные пересечения (θ_1 и θ_2), но она является однозначной, пример ее применения приведен на рис. 9.

• • •

Опасные пересечения в микропрограммах не являются опасными при синхронной интерпретации. Микропрограммы с такого рода пересечениями всегда дают детерминированные синхронные вычисления. Следовательно, подкласс $\mathcal{Q}$ является собственным подмножеством класса $\mathcal{D}$, $\mathcal{Q} \subset \mathcal{D}$.

Совпадение истор клеток во всех реализациях асинхронного вычисления означает, что микропрограмма, по которой ведется вычисление, устроена следующим образом. Если на i-м шаге вычисления выполнялись условия применимости некоторой микрооперации θ_{kx}, предписывающей замену (a,x) на (b,x), и на следующем шаге эти условия были нарушены в результате выполнения другой применимой микрооперации, то клетка (a,x) на одном из дальнейших шагов вычисления попадет в зону действия некоторой микрооперации $\theta_{k'x'}$, совершающей замену (a,x) на (b,x). А до тех пор (a,x) либо находится вне зоны действия любой другой микрооперации, либо участвует в микрооперациях только как контекстная клетка.

Асинхронное вычисление называется у с т о й ч и в ы м, если в нем условия выполнения каждой микрооперации сохраняются до тех пор, пока эта микрооперация не выполнится. Иными словами, вычисление устойчиво, если на i-м шаге выполнились условия применимости некоторой микрооперации θ_{kx}, предписывающей замену (a,x) на (b,x), то именно микрооперация θ_{kx} и произведет эту замену. Это определение аналогично определению устойчивости параллельных схем программ [4], где устойчивой считается схема, в которой каждый оператор сохраняет готовность к включению до тех пор, пока такая готовность не реализуется. Примером устойчивого является вычисление на рис.2.

Параллельная микропрограмма, порождающая устойчивое вычисление для любого исходного слова, называется у с т о й ч и в о й. Классу устойчивых не могут принадлежать микропрограммы с опасными пересечениями. Из класса устойчивых по чисто формальному признаку выпадают и микропрограммы, имеющие безопасные пересечения по полным базам, поскольку из двух пересекающихся таким образом и применимых в одном слове микроопераций выполниться может быть только одна. Таким образом, класс устойчивых составляют микропрограммы, в которых микрокоманды могут пересекаться только по контекстам, и в таком случае выполнение одной микрокоманды не может нарушить условия применимости другой.

Обозначим через $\tilde{\tilde{D}}$ класс параллельных микропрограмм, порождающих только детерминированные вычисления и в синхронной, и в асинхронной интерпретациях, $\tilde{\tilde{D}} = \tilde{D} \cap \bar{D}$. Имеют место отношения

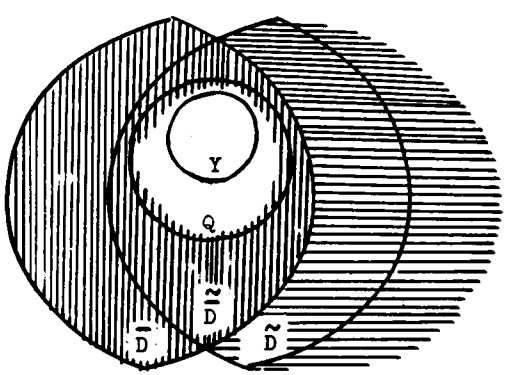

Рис. I0

$$\tilde{D} \supset \tilde{\tilde{D}} \supset Q \supset Y. \qquad (1)$$
$$\bar{D} \supset$$

Выделим подклассы микропрограмм в соответствии с типами пересечения микрокоманд. Обозначим: P - класс микрокоманд с противоречивыми пересечениями (микропрограммы этого класса могут иметь опасные

пересечения), H - класс микропрограмм с опасными пересечениями, но без противоречивых, U - класс микропрограмм, если и имеющих пересечения, то только безопасные. Эти подклассы попарно не пересекаются и находятся в следующих отношениях с подклассами $\bar{D}, \tilde{D}, \tilde{\bar{D}}, Q, Y$:

$$\bar{D} = H \cup U,$$

$$Y \subset U \subset Q, \qquad (2)$$

$$D \setminus \tilde{\bar{D}} \subset P.$$

Диаграмма на рис.10 отображает отношения (1) и (2). На ней область, заштрихованная вертикально, соответствует подклассу H, горизонтально - подклассу P, внутренняя область без штриховки соответствует подклассу U.

3. Параллельные микропрограммы и сети Петри

Кроме графа вычисления, для представления асинхронных вычислений удобно использовать сети Петри [5]. Заметим, что сеть Петри, моделирующая некоторое асинхронное вычисление, относится к графу этого вычисления так же, как она относится к своему графу достижимых маркирований. Другими словами, граф вычисления и граф достижимых маркирований соответствующей сети Петри изоморфны.

Поставим в соответствие клетке (состояние, имя) позицию P сети Петри, а микрооперации - переход t. Входными для перехода t, представляющего микрооперацию θ_{j2}, являются позиции, соответствующие клеткам слова $S_{j1}(z) * S_{j2}(z)$, выходными - позиции, соответствующие клеткам слова $S_{j3}(z) * S_{j2}(z)$. Видно, что клетки, входящие в контекстную часть $S_{j2}(z)$ микрооперации, соответствуют позициям, являющимся для перехода t и входными, и выходными. Позиции, соответствующие клеткам исходного слова, являются входными для сети, и метки в них определяют начальное маркирование сети. Выходными для сети являются те позиции, которые не имеют входа ни в один переход, кроме, может быть, тех переходов, из которых они вышли.

На рис. 11 и 12 даны две сети Петри, моделирующие асинхронные вычисления, представленные на рис. 2 и 7 соответственно. Каждая из сетей в этих простых случаях наглядно демонстрирует детерминированность (N_1) или недетерминированность (N_2) соответствующих вычислений. При любой из возможных последовательностей срабатывания

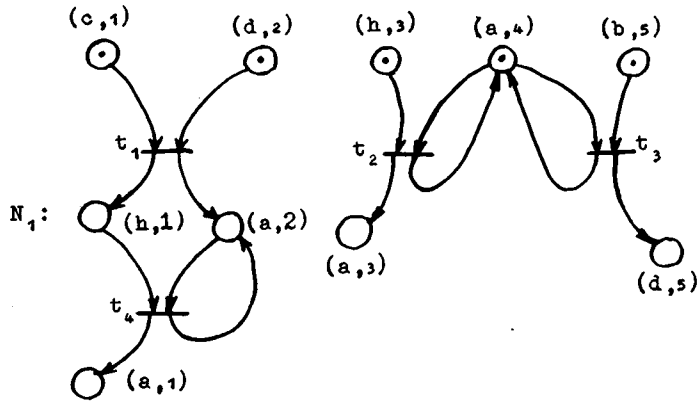

Рис. II. Сеть Петри для вычисления из примера 2.

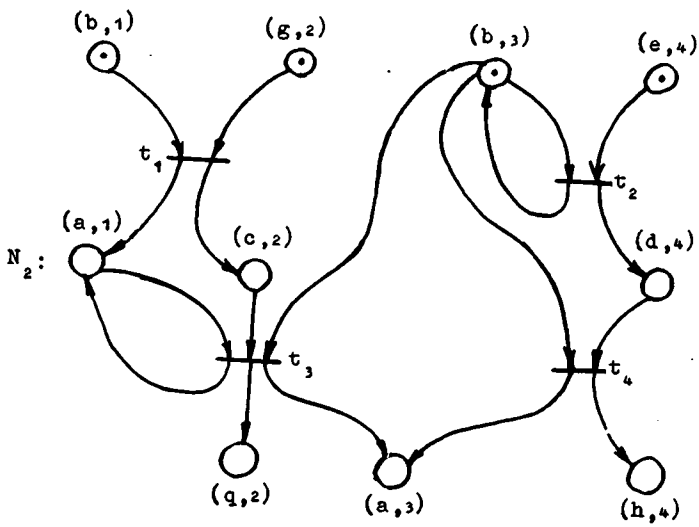

Рис.I2. Сеть Петри для вычисления из примера 7.

переходов сеть N_1 закончит свою работу, имея метки в выходных позициях (a,1),(a,2), (a, 3), (a, 4), (d,5) - это единственный результат вычисления. В N_2 имеется два альтернативно срабатывающих перехода t_3 и t_4, и, кроме того, t_2 не может сработать после t_3. Сеть N_2 может закончить свою работу при одном из трех вариантов размещения меток: {(a,1), (q,2), (a,3), (e,4)}, {(a,1), (q,2), (a,3), (d,4)}, {(a,1), (c,2), (a,3), (h,4)} - это значит, что вычисление не имеет единственного результата.

Сети Петри, моделирующие асинхронные вычисления по параллельным микропрограммам, относятся к классу о р д и н а р н ы х сетей [I,5] (в ординарных сетях нет ограничений на типы связей между позициями и переходами, кратность дуг не более единицы, вектор начального маркирования состоит только из единиц и нулей). Для любого вычисления сеть является б е з о п а с н о й, т.е. при любом маркировании, достижимом из начального, любая позиция сети имеет не более одной метки. Это действительно так: позиция - это пара (состояние, имя), а клетка с данными именем может либо иметь определенное состояние (метка равна единице), либо не иметь его (метка равна нулю), и перенесение меток в результате срабатывания перехода соответствует изменению состояний клеток с определенными именами.

Рассматриваемые сети Петри являются с т р о г о к о н с е р в а т и в н ы м и [5], т.е. при любом маркировании, достижимом из начального, число меток в сети точно такое же, что и при начальном маркировании. Это действительно так, поскольку в сети каждый переход имеет столько же входных позиций $I(t_i)$, сколько и входных $O(t_i)$, $I(t_i) = O(t_i)$.

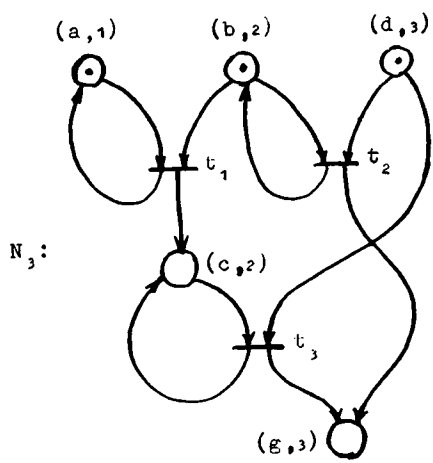

Рис. I3

Рассмотрим особенности сетей Петри, моделирующих однозначные вычисления. Начнем с определения устойчивых сетей Петри [6]. Сеть называется у с т о й ч и в о й, если в ней срабатывание од-

ного из двух переходов, возбужденных при некотором маркировании, достижимом из начального, не может лишить возможности сработать другой переход. Сеть N_1 на рис.II устойчива, а сеть N_2 на рис.I2 не является устойчивой, поскольку переходы t_3 и t_4, возбужденные при наличии меток в позициях (a,1), (c, 2), (b,3), (d,4), не могут оба реализовать свою готовность к срабатыванию.

Построим сеть Петри N_3 (рис.I3), моделирующую асинхронное вычисление для исходного слова $W = \{(a,1),(b,2),(d,3)\}$ по параллельной микропрограмме

$$\Psi = \begin{cases} \theta_1\colon (b,x) * (a,x-1) \to (c,x), \\ \theta_2\colon (d,x) * (b,x-1) \to (g,x), \\ \theta_3\colon (d,x) * (c,x-1) \to (g,x). \end{cases}$$

Сеть N_3 не является устойчивой. При начальном маркировании в N_3 возбуждены переходы t_1 и t_2, но после того, как сработает t_1, переход t_2 сработать не может. Однако в этой ситуации функцию перехода t_2 по передвижению метки из позиции (d,3) в (g,3) берет на себя переход t_3. Таким образом, относительно передачи метки из (d,3) в (g,3) переходы t_2 и t_3 в сети N_3 являются эквивалентными: обязательно сработает один из них и выполнит передачу метки. И если на устойчивую сеть Петри можно смотреть как на устойчивую относительно срабатываний возбужденных переходов, то на сеть Петри типа N_3 можно смотреть как на устойчивую относительно передачи меток из одних позиций в другие, и в отличие от устойчивых сетей, которые можно называть t - у с т о й ч и в ы м и, сети типа N_3 удобно называть p - у с т о й ч и в ы м и.

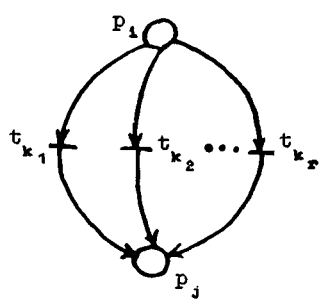

Рис. I4

Дадим более строгое определение p -устойчивых сетей Петри. Две позиции p_i и p_j будем называть смежными, если имеется группа переходов $t_{k_1},\dots,t_{k_r}$, для которых одна из позиций является общей входной, а другая общей выходной (рис.I4). Эти переходы можно назвать эквивалентными для позиций p_i и p_j. Переход назовем неустой-

чивым, если он, будучи возбужденным, имеет риск утратить готовность к срабатыванию, не реализовав ее. Понятно, что упомянутые выше переходы $t_{k_1},\ldots,t_{k_r}$ могут быть неустойчивыми. Сеть Петри называется p-устойчивой при начальном маркировании M_0, если в ней для входных и выходных позиций любого перехода t_q, возбужденного при некотором достижимом из M_0 маркировании M', выполняются условия: I) для каждой входной позиции p_i и некоторого непустого подмножества выходных позиций $P^i = \{p_{j_1},\ldots,p_{j_k}\}$ в любой последовательности маркирований, достижимых из M', найдутся два соседних маркирования M_l и M_{l+1} таких, что $M_l = (\ldots\underset{p_i}{1}\ldots\underset{p_{j_1}}{0}\ldots\underset{p_{j_k}}{0}\ldots)$, $M_{l+1} = (\ldots\underset{p_i}{0}\ldots\underset{p_{j_1}}{1}\ldots\underset{p_{j_k}}{1}\ldots)$; 2) $|UP^i| = u$ - числу выходных позиций перехода t_q. Иными словами, даже если возбужденный неустойчивый переход t_q не сработает, то его выходные позиции получат, а входные потеряют метки в результате срабатывания других переходов сети. С точки зрения структурных свойств для p-устойчивости требуется, чтобы любая входная позиция p_i неустойчивого перехода t_q имела бы смежную с ней выходную позицию. И тогда сеть будет p-устойчивой, если в любой последовательности срабатываний переходов, порождаемой начальным маркированием M_0, будет иметь место один из эквивалентных переходов для каждой пары смежных позиций. Рассматриваемые здесь консервативные сети таковы, что каждая входная позиция неустойчивого перехода имеет свою выходную смежную с ней, и эти позиции соответствуют одноименным клеткам перерабатываемого слова. На рис.I5,I6 приведена p-устойчивая сеть N_4, моделирующая однозначное вычисление, граф которого изображен на рис.9.

Заметим, что эпизоду вычислительного процесса, в котором выполняются условия применимости двух непротиворечиво пересекающихся по полным базам микрокоманд, соответствует фрагмент p-устойчивой сети, например, приведенный на рис.I7.

Устойчивые вычисления моделируются t-устойчивыми сетями Петри (рис.II). В таких сетях ни один из двух возбужденных при некотором маркировании переходов не может помешать сработать другому, т.к. эти переходы с общими для них позициями образуют петли, что соответствует тому, что в устойчивых микропрограммах микрокоманды могут пересекаться только по контекстам. В сетях, моделирующих вычисления по устойчивым микропрограммам, могут быть и

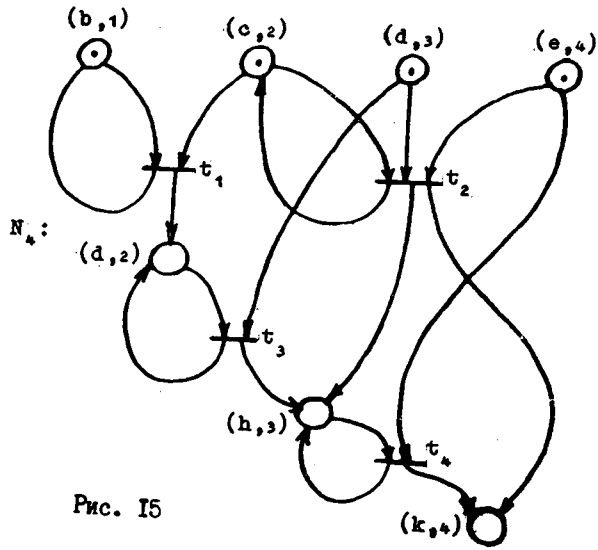

Рис. I5

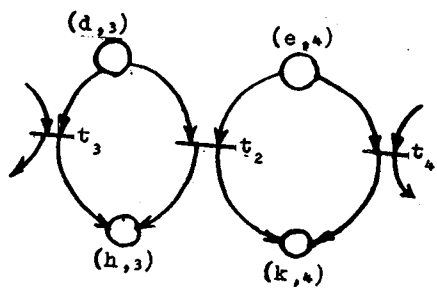

Рис. I6

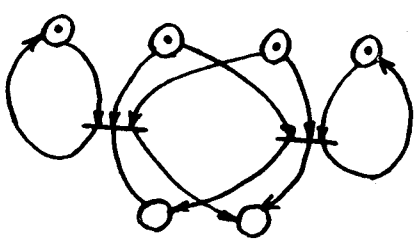

Рис. I7

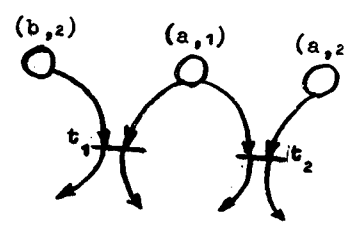

Рис. I8

такие фрагменты, в которых два перехода разделяют одну входную позицию, но петель с ней не образуют. В таком случае эти переходы не могут быть оба возбуждены ни при каком достижимом маркировании. Пример такого фрагмента приведен на рис. I8, где не может иметь места маркирование, при котором во всех трех позициях есть метки, так как ни в одном слове не могут быть вместе клетки (a,2) и (b,2), а значит, и переходы t_1 и t_2 не могут быть возбуждены при одном маркировании.

4. Распознавание недетерминированности параллельных микропрограмм

Параллельная микропрограмма может интерпретироваться сетью автоматов [2]. Асинхронное выполнение микропрограммы соответствует работе сети автоматов без принудительной синхронизации. В таком случае можно считать, что поведение сети имеет сложный характер, сочетающий синхронный и асинхронный режимы работы. Два автомата в такой сети могут сработать синхронно в том смысле, что моменты начала срабатывания того и другого отличаются меньше, чем на некоторый данный отрезок времени τ. В отличие от асинхронной интерпретации в широком смысле (отсутствие принудительной синхронизации) будем называть о р д и н а р н о й асинхронную интерпретацию, понимаемую как такой способ выполнения параллельной микропрограммы, при котором в любой момент времени выполняется только одна микрооперация.

Таким образом, условия детерминированности асинхронной интерпретации параллельной микропрограммы должны включать в себя условия детерминированности как синхронной, так и ординарной асинхронной интерпретации. А это значит, что распознавание детерминированности асинхронной интерпретации состоит в проверке принадлежности микропрограммы классу $\tilde{D}$.

Задача распознавания недетерминированности синхронной интерпретации параллельной микропрограммы решена в [3] как задача распознавания противоречивости микропрограммы по ее тексту. При этом предлагается преобразовать данную микропрограмму в эквивалентную ей неймановскую (такую, в микрокомандах которой конфигурации S_{i_1} и S_{i_2} являются одноклеточными вида (a,x)). Попарное сравнение микрокоманд неймановской микропрограммы дает возможность выявить противоречивость исходной микропрограммы.

Выявление возможной противоречивости в параллельной микропрограмме является первым этапом решения вопроса о детерминированности асинхронной интерпретации этой микропрограммы. Поскольку неединственность конечного результата асинхронного вычисления вызывается опасным пересечением микрокоманд, то естественным подходом к распознаванию недетерминированности ординарной асинхронной интерпретации соответствующей микропрограммы является проверка работы микропрограммы на опасных словах, т.е. на словах, в которых имеется опасное пересечение микрокоманд. Опасные слова конструируются из левых частей пересекающихся микрокоманд для всех возможных сдвигов, т.е. для всех возможных значений вектора k. Для двух микрокоманд опасное слово конструируем из левых частей при некотором сдвиге k_1: $S_{i1} * S_{i2} * S_{j1}^{(k_1)} * S_{j2}^{(k_1)}$, и таких слов может быть сконструировано столько, при скольких значениях вектора k пересечение этих микрокоманд непусто. Для пары (θ_i, θ_i) также могут быть сконструированы опасные слова. Число всех опасных слов для всех пар микрокоманд, в том числе и (θ_i, θ_i), может быть велико. Так, микропрограмма сложения многих троичных чисел (рис. 3) имеет 6 микрокоманд, общее число опасных слов, конструируемых из пар микрокоманд при всех возможных сдвигах, равно 19.

Зададимся вопросом, на каких опасных словах достаточно проверить работу микропрограммы, чтобы сделать вывод о детерминированности или недетерминированности ординарной, асинхронной интерпретации. Оказывается (ниже это будет доказано), что достаточно это сделать на всех опасных словах, образованных только для пар пересекающихся микрокоманд. Обозначим через Δ множество всех опасных слов для всех пар пересекающихся микрокоманд. Имеет место

ТЕОРЕМА I. Для того чтобы ординарная асинхронная интерпретация параллельной микропрограммы была детерминированной, необходимо и достаточно, чтобы были детерминированы вычисления по этой микропрограмме для всех слов $W_i \in \Delta$.

ДОКАЗАТЕЛЬСТВО. Необходимость очевидна. Достаточность докажем от противного. Пусть вычисления по микропрограмме Ψ для всех опасных слов $W_i \in \Delta$ детерминированы. Предположим, нашлось слово $W \notin \Delta$ такое, к которому применимы более двух попарно опасно пере -

секающихся микрокоманд, и вычисление $\tilde{\Psi}_W$ недетерминировано. Это означает, что в одной из вершин графа вычисления процесс разветвился (т.е. оказались применимы по меньшей мере две опасно пересекающиеся микрокоманды) и не пришел к единственному результату. Это противоречит предположению о том, что вычисления по микропрограмме Ψ для всех опасных слов для любой пары пересекающихся микрокоманд детерминированы.

• • •

ТЕОРЕМА 2. Асинхронная интерпретация параллельной микропрограммы однозначна, если и только если однозначны вычисления по этой микропрограмме для всех $W_i \in \Delta$.

Ход доказательства этой теоремы аналогичен ходу доказательства теоремы I. Очевидным является утверждение типа теоремы 2 для устойчивой асинхронной интерпретации параллельной микропрограммы.

Таким образом, для того чтобы установить, является ли детерминированной ординарная асинхронная интерпретация параллельной микропрограммы, следует сконструировать все опасные слова для каждой пары микрокоманд и для каждого слова построить граф вычисления по данной микропрограмме. Вместо построения графа вычисления можно исследовать моделирующую это вычисление сеть Петри.

В сети Петри переход t называется живым при заданном маркировании M, если для любого M', достижимого из M, существует последовательность срабатывания, которая содержит t. Сеть Петри называется живой при заданном маркировании M, если при этом маркировании все переходы являются живыми. Маркированием M, относительно которого будет устанавливаться живость сетей, моделирующих асинхронные вычисления, является начальное маркирование, т.е. маркирование, определяемое исходным словом. Моделирующую сеть Петри преобразуем следующим образом. Все выходные позиции выводим на один дополнительно вводимый переход, и этот переход соединяем с входными позициями. На рис.I9 показана таким образом дополненная сеть N_2 (рис.II), в ней переход t_6 и позиция p присутствуют только для удобства изображения сети.

ТЕОРЕМА 3. Асинхронное вычисление детерминировано, если и только если моделирующая его сеть, замкнутая от выходных позиций к входным, жива.

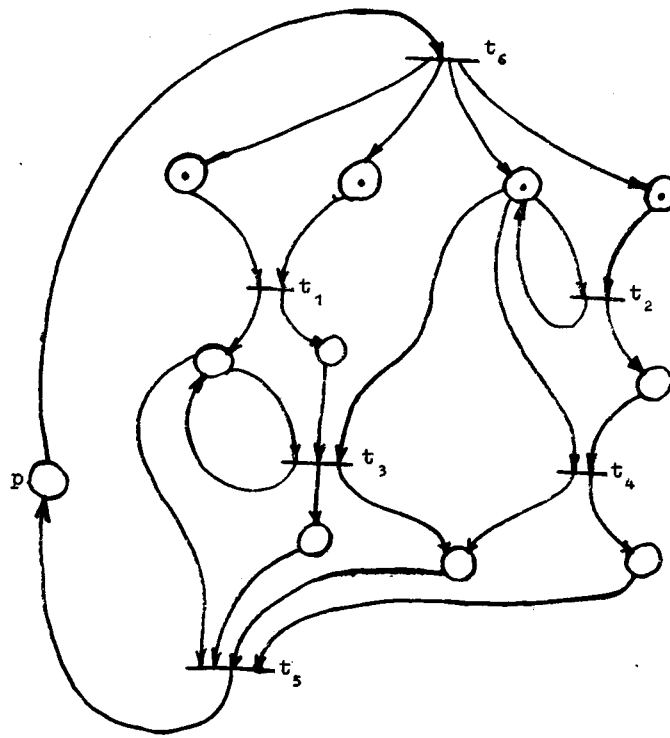

Рис. I9

ДОКАЗАТЕЛЬСТВО. Прямое утверждение. Если сеть жива, значит, имеет место маркирование, при котором метки стоят только в выходных позициях, следовательно, вычисление, моделируемое такой сетью, имеет единственный результат. Обратное утверждение. Предположим, что вычисление детерминировано, а сеть оказывается неживой, значит, работа сети окончилась при маркировании, не соответствующем единственному результату вычисления - метки только в выходных позициях, а это противоречит предположению о детерминированности вычисления.

* * *

Из всего изложенного следует алгоритм распознавания детерминированности асинхронной интерпретации параллельной микропрограммы $\Psi = \{\theta_i\}$, $i = \overline{1,v}$.

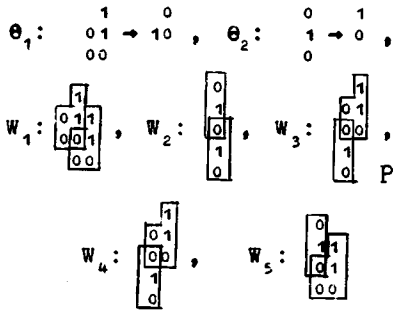

Рис. 20. Микропрограмма из примера 9 и опасные пересечения ее микрокоманд.

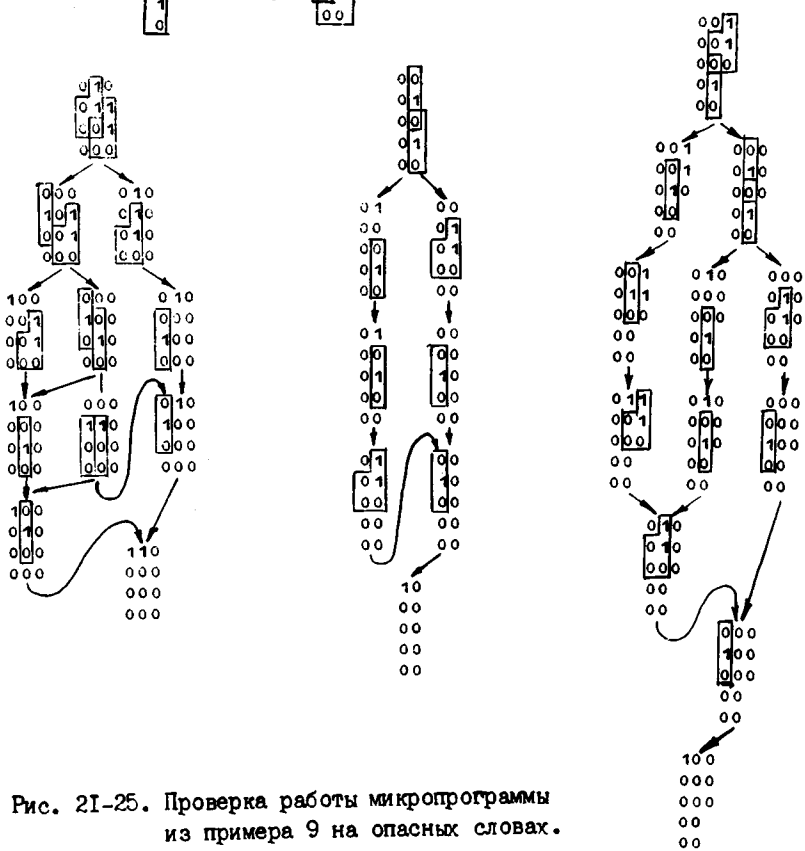

Рис. 21-25. Проверка работы микропрограммы из примера 9 на опасных словах.

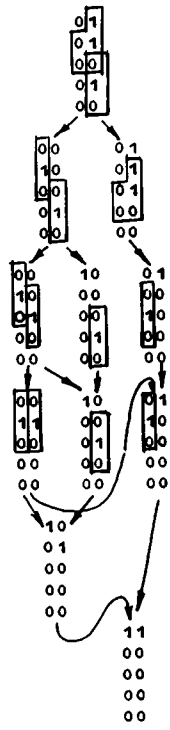

Рис. 24

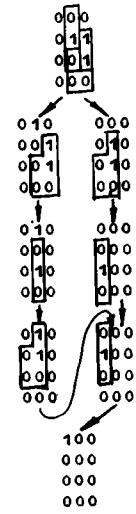

Рис.25

I. Каждая пара микрокоманд θ_i, θ_j, $i,j = \overline{1,v}$, анализируется с точки зрения противоречивого пересечения базовых конфигураций. Если для некоторой пары микрокоманд такое пересечение обнаруживается, то делается вывод, что микропрограмма недетерминирована, и выполнение алгоритма заканчивается.

2. Каждая пара микрокоманд θ_i, θ_j, $i,j=\overline{1,v}$, анализируется с точки зрения опасного пересечения. Если для некоторой пары такое пересечение обнаруживается (напомним, что для одной пары микрокоманд может быть несколько вариантов опасных пересечений), то конструируется соответствующее опасное слово W. Затем строится либо граф вычисления $\tilde{\Psi}_W$, либо сеть Петри, моделирующая вычисление Ψ_W. Если граф имеет более одной конечной вершины или сеть оказывается неживой, то делается вывод о недетерминированности микропрограммы, и выполнение алгоритма заканчивается.

3. Если выполнение алгоритма не прервалось ни в первом, ни во втором пунктах, то делается вывод, что микропрограмма детерминирована, и выполнение алгоритма заканчивается.

Работу алгоритма проиллюстрируем на примере микропрограммы сложения многих двоичных чисел [2].

ПРИМЕР 9. Параллельная микропрограмма Ψ сложения многих двоичных чисел состоит из двух микрокоманд

$$\Psi = \begin{cases} \theta_1\colon \{(1,(x,y)),(1,(x,y+1)),(0,(x-1,y))\} \bullet \{(0,(x-1,y-1)), \\ \quad (0,(x,y-1))\} \to \{(0,(x,y)),(0,(x,y+1)),(1,(x-1,y))\}, \\ \theta_2\colon \{(1,(x,y)),(0,(x,y+1))\} \bullet (0,(x,y-1))\} \to \\ \quad \to \{(0,(x,y)),(1,(x,y+1))\}, \end{cases}$$

геометрические образы которых показаны на рис. 20, там же изображены все варианты пересечений микрокоманд. Противоречивых пересечений среди всех возможных не оказалось, считаем, что благополучно пройден первый пункт алгоритма, и приступаем ко второму. Все пересечения относятся к классу опасных. Слова $W_1,\ldots,W_5$ (рис. 20) являются теми опасными словами, на которых нужно проверить работу микропрограммы Ψ. Графы вычислений $\tilde{\Psi}_{W_1},\ldots,\tilde{\Psi}_{W_5}$ изображены на рис. 2I-25. Каждый из них имеет одну конечную вершину, вычисления $\tilde{\Psi}_{W_1},\ldots,\tilde{\Psi}_{W_5}$ детерминированы, следовательно, асинхронная интерпретация параллельной микропрограммы сложения многих двоичных чисел детерминирована.

Литература

I. БАНДМАН О.Л. Асинхронная интерпретация параллельных микропрограмм. -Кибернетика, I983, № 5, с. I7-23.

2. Методы параллельного микропрограммирования /Под ред. Бандман О.Л.-Новосибирск: Наука, I98I. - I8I с.

3. СЕРГЕЕВ С.Н. Распознавание непротиворечивости алгоритмов стационарных подстановок. -В кн.: Архитектура вычислительных систем с программируемой структурой (Вычислительные системы, вып.82). Новосибирск, I980, с. I8-25.

4. КАРП Р.М., МИЛЛЕР Р.Е. Параллельные схемы программ. - Кибернетический сборник № I3, I976, с. 5-6I.

5. PETERSON J.L. Petri-Net Theory and the Modeling of Systems.-Prentice-Hall,1981.- 290 p.

6. LANDWEBER L.H., ROBERTSON E.L. Properties of Conflict-Free and Persistent Petri Nets.- J.of the Association for Comp. Mach.,1978,July,v.25,№ 3,p.352-364.

Поступила в ред.-изд.отд.
20 мая I983 года