

## ПРЕОБРАЗОВАНИЕ ОПЕРАТОРНЫХ СХЕМ В АСИНХРОННЫЕ ПРОГРАММЫ

В.Е. Котов, А.С. Нариньяни

(Новосибирск)

В работе [1] был предложен метод программирования (асинхронное программирование), позволяющий решать задачи на вычислительных системах с произвольно меняющимся в процессе решения числом процессоров. Там же изложена общая постановка задачи и возможности применения метода. Содержанием настоящего доклада является изложение некоторых конкретных результатов в этой области, в основном касающихся получения асинхронных программ по исходным операторным схемам.

1. Пусть  $A_i$  - некоторый оператор над полем переменных  $T$ , а  $x_i$  - поставленная в соответствие этому оператору двузначная функция, определенная на том же поле (спуск о в а я ф у н к ц и я). Пару  $\langle x_i, A_i \rangle$  будем называть б л о к о м  $X_i$ . Фиксированную совокупность блоков, заданных над общим полем, будем называть к в а з и п р о г р а м м о й над этим полем  $\langle \{X_i\}, T \rangle$ .

2. Введем гипотетическую систему (которую будем называть а с и н х р о н н о й с и с т е м о й, или  $A$  - с и с т е м о й), реализующую по данной квазипрограмме  $\langle \{X_i\}, T \rangle$  вычислительный процесс над полем  $T$  следующим образом:

Считается, что  $A$  - система работает в дискретном времени  $\tau$ . В каждый конкретный момент  $\tau$ , каждый из блоков находится в одном из двух состояний - "рабочем" или "нерабочем". Каждому блоку, независимо от других, ставится в соответствие монотонная последовательность произвольно выбранных моментов времени, в которые происходит проверка состояния данного блока. Если в момент проверки данный блок находится в "нерабочем" состоянии, причем его спусковая функция имеет значение "1" при

соответствующем данному моменту времени значении поля, то этот блок переходит в "рабочее" состояние, пребывая в нём некоторое произвольное время  $\bar{\tau}_j^1 \div \bar{\tau}_j^2$ , после чего он вновь переводится в "нерабочее" состояние. За это время оператор данного блока применяется к полю  $T$ . ( $i$  - индекс блока,  $j$  - номер применения данного оператора).

Если же спусковая функция данного блока в момент проверки имеет значение "0" или же сам блок находится в "рабочем" состоянии, то состояние блока не изменяется и его оператор не применяется к полю.

Работа  $A$ -системы, реализующей квазипрограмму  $\langle \{x_i\}, T \rangle$ , начинается в момент  $\tau = 0$  при начальном значении поля  $T_{j0}$  и "нерабочем" состоянии всех блоков. Система работу заканчивает, когда какой-либо из блоков подает специальный управляющий сигнал, т.е. когда все блоки будут в "нерабочем" состоянии и его спусковые функции равны "0".

3. Если при работе  $A$ -системы, реализующей некоторую квазипрограмму при некотором начальном значении поля  $T_{j0}$ , зафиксировать для каждого блока конкретные значения последовательности моментов  $\bar{\tau}_1^1, \bar{\tau}_1^2, \bar{\tau}_2^1, \bar{\tau}_2^2, \dots$ , то такой конкретный вычислительный процесс будем называть  $A$ -реализацией  $(R)$  данной квазипрограммы для соответствующего  $T_{j0}$ .

Таким образом, всякая квазипрограмма и  $A$ -система задают для каждого начального значения поля полное множество  $A$ -реализаций этой квазипрограммы.

4. Пусть задано некоторое определение эквивалентности  $A$ -реализаций. Если все  $A$ -реализации некоторой квазипрограммы эквивалентны согласно этому определению для любого начального  $T_j \in \{T\}^*$ , то такую квазипрограмму будем называть асинхронной программой ( $A$ -программой) на множестве начальных значений  $\{T\}^*$  относительно данного определения эквивалентности.

5. Результатом  $r(i, j, k)$  будем называть переменную, значение которой для каждой пары  $\langle k, T' \rangle$  равно значению  $k$ -ой выходной переменной оператора  $A_i$  при  $j$ -ом его применении в  $R$  и начальном значении  $T'$ .

Будем говорить, что данная квазипрограмма является  $A$ -программой по множеству результатов  $\{r\}^*$  на множестве начальных значений  $\{T\}^*$ , если для любого начального  $T' \in \{T\}^*$  значение любого  $r(i, j, k) \in \{r\}^*$  является одним и тем же для всех реализаций этой квазипрограммы.

6. Информационным графом реализации  $R$  назовем граф  $\bar{G}_R = \langle \{A_i^j\}, \Gamma_R \rangle$ , где  $A_i^j$  -  $j$ -ое применение оператора  $A_i$ , а между  $A_i^j$  и  $A_i^k$  находится дуга с индексом  $\langle m, n \rangle$  если в  $R$   $m$ -му выходу  $A_i^j$  и  $n$ -му входу  $A_i^k$  соответствует одна и та же переменная  $t$  из  $T$  и в интервале времени  $(\bar{\tau}_j^i + \bar{\tau}_k^i)$  ни одно из применений операторов не имеет в качестве выходной переменную  $t$ .

Очевидно, что если  $\bar{G}_R$  совпадает с  $\bar{G}_{R'}$ , то  $R'$  и  $R''$  эквивалентны по всем своим результатам.

7. Следующие условия являются достаточными для того, чтобы квазипрограмма  $\langle \{X_i\}, T \rangle$  являлась  $A$ -программой на множестве  $\{T\}$  по всем результатам своих реализаций:

При любой  $A$ -реализации  $R$  для любого начального  $T' \in \{T\}$  справедливо:

а) Спусковая функция  $x_i$  принимает значение "1" только в том случае, когда входной набор  $X_i$  ( $\forall x_i$ ) примет значение, при котором все выходные переменные однозначно определены.

б) Если хотя бы для одной  $A$ -реализации и набора  $i, j, k, l$  выполняется  $\bar{\tau}_j^i \leq \bar{\tau}_k^i \leq \bar{\tau}_l^i$ , то одновременно справедливо и

$$1) \text{ Вых } X_i \cap \text{ Вых } X_k = \Lambda;$$

$$2) \text{ Вых } X_i \cap \text{ Вх } X_k = \Lambda;$$

$$3) \text{ Вх } X_i \cap \text{ Вых } X_k = \Lambda.$$

8. Рассмотрим произвольную операторную схему  $\langle \{A_i\}, \Gamma_0 \rangle$ , где  $\{A_i\}$  - множество операторов, состоящее из преобразователей  $A_i^+$  и распознавателей  $A_i^*$ .

Оператор будем называть *возвратным*, если он лежит на контуре в данной операторной схеме.

9. Каждой схеме можно поставить в соответствие множество  $A$ -программ, эквивалентных ей по всем ее результатам на всем множестве её начальных значений. Однако между собой эти программы могут в значительной степени отличаться множествами своих реализаций, сопоставленных с одними и теми же начальными значениями (т.е. могут быть как бы в различной степени асинхронными).  $A$ -программу можно получать, взяв за исходную непосредственно саму операторную схему, или же операторную схему, полученную из основной с помощью какой-либо системы формальных преобразований. Очевидно, что при соответ-

вующей направленности таких преобразований во втором случае результирующие А-программы будут обладать большей асинхронностью, чем в первом.

Авторами рассматривалась задача получения максимально асинхронной А-программы из исходной операторной схемы при переименовании переменных и расклеивании операторов.

Ю. Решение этой задачи было разбито на два этапа. Результатом первого должен был явиться наиболее простой способ построения асинхронной программы по исходной операторной схеме, не предъявляющий к результирующей А-программе никаких других требований, кроме максимальной асинхронности в вышеупомянутом смысле. С помощью расклеивания вершин исходная операторная схема переводится в эквивалентную ей, но не содержащую контуров; последняя, в свою очередь, с помощью переименования переменных переводится в операторную схему с распределенной памятью [2] (обладающую, очевидно, тем качеством, что никакой её переменной ни в какой реализации не может быть дважды присвоено значение). Каждый оператор этой конечной схемы по определенному закону снабжается спусковой функцией и некоторыми дополнительными операторами над специально вводимыми управляющими переменными. Полученная таким образом асинхронная программа, обладая максимальной асинхронностью, отвечает поставленным требованиям, однако имеет очевидные недостатки — излишне широко использует потенциально бесконечную память и потенциально бесконечное число блоков.

II. В задачу второго этапа входило получение методов улучшения результирующей программы, без уменьшения её асинхронности, т.е. уменьшение числа информационных и управляющих переменных, числа блоков, упрощение спусковых функций и т.д. Алгоритм перевода разделяется здесь на две части: направленное переименование переменных операторной схемы; перевод полученной схемы в А-программу.

Будем говорить, что  $m$ -ый выход  $A_j$  совместен с  $n$ -ым выходом  $A_i$  в некоторой программе, когда для любой реализации  $R$  этой программы справедливо: если  $\tau_1^j > \tau_k^i$  то,  $\Gamma_R(\langle A_i^k, n \rangle) \subset \hat{\Gamma}^{-1}(A_j^1)$ , где  $\Gamma_R(\langle A_i^k, n \rangle)$  — множество вершин в  $G_R$ , в которые заходят дуги, исходящие из  $A_i^k$  и имеющие первый индекс  $n(\langle n, \dots \rangle)$ ;  $\hat{\Gamma}^{-1}$  — транзитивное замыкание отображения  $\Gamma_R^{-1}$ .

Два выхода совместны, если каждый из них совместен со

вторым. Были сформулированы и доказаны достаточные условия совместности двух выходов в операторной схеме и самосовместности выходов возвратных операторов.

Переименование переменных производится следующим образом:

а) Определяется совместность и самосовместность выходов в исходной операторной схеме.

б) Составляется граф несовместности  $G_n$ , вершинами которого являются выходы операторов, причем две вершины соединены ребром, если соответствующие выходы несовместны.

в) Выделяется подграф  $G^*$  графа  $G_n$ , содержащий только несовместные выходы возвратных операторов, и производится раскраска вершин этого подграфа.

г)  $G_n$  с раскрашенными вершинами  $G^*$  докрашивается. Каждому цвету  $G^*$  соответствует потенциально бесконечный вектор переменных, каждому цвету  $G_n \setminus G^*$  — одна отдельная переменная (которая может быть присоединена к любому из векторов). Происходит соответствующая замена переменных в исходной операторной схеме. Все операторы схемы разбиваются на две группы: к первой относятся только возвратные операторы, не имеющие самозависимых выходов, ко второй — все остальные. Каждый оператор первой группы расщепляется, т.е. каждый из них заменяется потенциально бесконечной серией операторов, различающихся только значением константы, запоминающей сопоставленный с каждым из них индекс применения. Операторы второй группы остаются без изменения.

При переводе этой результирующей схемы в А-программу каждый оператор схемы снабжается спусковой функцией, представляющей собой логическую функцию от предикатов, описывающих состояние самого блока, а также его готовность к обработке входных переменных. Поскольку входные и выходные наборы каждого оператора в общем случае меняются в процессе реализации программы (а следовательно, меняются и спусковые функции), то к блокам А-программы добавляются специальные операторы, обеспечивающие эти изменения.

## Л и т е р а т у р а

1. В.Е.Котов, А.С.Нариньяни. Асинхронные процессы над общей памятью. — Кибернетика, 1966, № 3.
2. А.П.Ершов. Об операторных схемах над общей и распределенной памятью. — Тезисы докладов к симпозиуму "Вычислительные системы", Новосибирск, 1966, стр.8.