

СПОСОБЫ ЗАПИСИ АЛГОРИТМОВ ДЛЯ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Д.А. Поспелов
(Москва)

Одной из актуальных задач, связанных с построением и эксплуатацией вычислительных систем (ВС), является проблема создания такого способа записи алгоритмов, который, с одной стороны, был бы достаточно простым для пользователя, а с другой стороны, содержал бы в удобной форме всю необходимую информацию для организации на ВС процесса вычислений. В настоящее время известно несколько способов использования ВС. При работе в мультипрограммном режиме на ВС одновременно реализуется некоторый поток задач, поступающих от пользователей. В этом потоке может быть задан приоритет, либо такой приоритет устанавливается самой ВС. При работе в режиме разделения времени большое количество пользователей одновременно работает с одной ВС, не обязательно зная о существовании других пользователей. В режиме ответ на запрос ВС выполняет ту совокупность программ из конечного списка программ, хранящихся постоянно в её памяти, на которые в течении определенного времени поступили запросы. Наконец, при параллельном режиме работы ВС решает одну большую задачу, которая по каким-то причинам не может быть реализована на одной вычислительной машине.

По-видимому, для каждого режима работы ВС выгодно (по крайней мере, в настоящее время) создавать свою специализированную структуру. Как следствие этого, и язык записи алгоритмов

для ВС, работающих в различных режимах, также должен быть различным. Такая специализация языка позволит наиболее эффективно организовать вычислительный процесс на ВС и получить простое аппаратное решение устройства управления ВС.

В настоящей работе проблема выбора способа записи алгоритмов для ВС рассматривается только для ВС, состоящих из обычных вычислительных машин, структура которых фактически сохранена, работающих в параллельном режиме.

Предполагая, что язык записи алгоритмов является многоуровневым. Но крайней мере, этот язык имеет три уровня: входной уровень, системный уровень и машинный уровень. В соответствии с этим будем различать входной язык, системный язык и машинные языки.

В качестве входного языка можно использовать любой из известных алгоритмических языков, несколько расширив его за счет введения в его структуру средств описания параллельности процессов, как это, например, сделано в работе [10] для языка АЛГОЛ. Андерсон ввел в АЛГОЛ пять дополнительных операторов, определяемых следующим образом: `< fork -оператор > ::= fork < список меток >; < join - оператор > ::= join < список меток >; < terminate - оператор > ::= terminate < список меток >; < obtain -оператор > ::= obtain < список переменных >; < release -оператор > ::= release < список переменных >; < список меток > ::= < метка >, < метка > | < список меток >, метка ; < список переменных > ::= < переменная >, < переменная > | < список переменных >, < переменная >.`

`fork` - оператор начинает выполнение тех параллельных сегментов, которые указаны в списке меток, `join` -оператор контролирует окончание всех сегментов, начатых `fork` -оператором и после окончания последнего из параллельных сегментов передает управление оператору, следующему за ним.

`terminate` -оператор прекращает выполнение указанных в списке меток операторов, `obtain` -оператор блокирует использование указанных переменных, а `release` -оператор снимает блокировки, поставленные `obtain` -оператором.

Аналогичная работа по расширению языка проделана и для языка ФОРТРАН. Сегментация исходных программ, записанных в алгоритмическом языке, предполагается проделанной человеком. Автоматизация сегментации вызывает большие трудности и первые попытки в этом направлении (например, в GIER АЛГОЛ [11]) пока ещё мало эффективны.

Более эффективным является организация автоматического сегментирования на уровне системного языка, который выступает в роли языка-посредника между входным языком и машинными языками. В работе [12] описываются основные принципы использования языка-посредника алгольного типа в качестве системного языка ВС. В качестве единиц такого языка предлагается использовать язык псевдокоманд, которые схемно или программно интерпретируются на машинах системы в последовательности команд, записанных на машинном языке. Этот путь будет наиболее эффективным, если системный язык будет использоваться не только как язык записи алгоритмов, реализуемых на системе, но и как язык управления вычислительным процессом.

Под управлением вычислительным процессом мы понимаем реализацию следующих функций: определение кусков программы, которые в данный момент времени могут быть реализованы, отбор среди них тех кусков, которые будут фактически реализовываться на машинах системы в ближайший интервал времени, распределение кусков по машинам системы, получение результирующей информации от машин системы и организация обмена информацией между машинами, контроль правильности реализации программы.

В качестве способа записи алгоритмов для вычислительных систем, позволяющего реализовать указанные выше функции управления, в настоящей работе предлагается использовать запись в виде специальной структуры, называемой ярусно-параллельной формой (ЯПФ). В работе [7] язык ЯПФ использовался для классификации алгоритмов, реализуемых на ВС, а в работе [8] числовые характеристики алгоритмов, записанных на языке ЯПФ, использовались для оценки производительности ВС централизованного типа. Ниже дается алгоритм перехода от записи программы на языке псевдокоманд к записи этой программы на языке ЯПФ. Прежде чем перейти к этому алгоритму дается краткое описание ярусно-параллельной формы записи алгоритмов.

Ярусно-параллельная форма записи алгоритмов есть представление процесса вычислений в виде мультиграфа без петель конечной или потенциально конечной длины. Вершины мультиграфа сопоставлены сегментам исходной программы, называемым

функциональными операторами, а дуги, соединяющие эти вершины, бывают двух типов: функциональные и управляющие. Каждая пара вершин соединена между собой не более чем двумя дугами. При наличии двух дуг, дуги принадлежат к различным типам. Если вершина i соединена с вершиной j дугой функционального типа, то это означает, что оператор с номером j функционально зависит от оператора с номером i . Если вершина с номером i соединена с вершиной, имеющей номер j дугой управляющего типа, то это означает, что оператор с номером i управляет оператором с номером j (оператор с номером i при этом является оператором условного перехода на два направления). Вершины мультиграфа взвешены векторами

$\langle t_1^i, t_2^i, \dots, t_r^i \rangle$, где i - номер оператора, а t_j^i

- время реализации (или математическое ожидание времени реализации) оператора i на машине типа j . Каждому оператору сопоставлено два списка: список входных идентификаторов и список выходных идентификаторов. Множества идентификаторов из списка входных идентификаторов взвешивают функциональные дуги, входящие в рассматриваемый оператор, а множества идентификаторов из списка выходных идентификаторов взвешивают дуги, выходящие из рассматриваемого идентификатора. Кроме того, каждому оператору сопоставляется еще два списка: список исходных идентификаторов и список окончательных идентификаторов.

Таким образом, каждая ЯПФ может быть задана с помощью следующих таблиц: главной матрицы $G = \| \epsilon_{ij} \|_{q \times q}$,

где q - число функциональных операторов в ЯПФ, ϵ_{ij} определяются следующим образом:

$$\epsilon_{ij} = \begin{cases} 0, & \text{если операторы с номерами } i \text{ и } j \text{ не связаны,} \\ 1, & \text{если оператор с номером } i \text{ функционально связан} \\ & \text{с оператором с номером } j \text{ (передает ему данные),} \\ 2, & \text{если оператор с номером } i \text{ управляет оператором} \\ & \text{с номером } j, \\ 3, & \text{если оператор с номером } i \text{ функционально связан} \\ & \text{с оператором с номером } j \text{ и кроме того передает} \\ & \text{ему управление,} \end{cases}$$

матрицы времен, $T = \| t_j^i \|_{q \times r}$, где t_j^i определе-

ны выше, а r - число различных типов машин, использованных в

ВС, совокупностей списков входных, выходных, исходных и окончательных идентификаторов для всех операторов системы.

Покажем, что ЯПФ позволяет, исходя из своего представления, организовать реализацию вычислительного процесса.

Для этого мы должны показать, что запись в виде ЯПФ позволяет решить все функции управления, перечисленные нами ранее. Первой функцией управления является определение в данный текущий момент процесса всех операторов, которые могут быть начаты в этот момент времени. Рассмотрим сначала начальный момент времени. Как следует из определения главной матрицы, в этот момент времени могут выполняться только те операторы, для которых в сопоставляемых им столбцах находятся только нулевые элементы. Все операторы, сопоставляемые нулевым столбцам матрицы, образуют множество допустимых для реализации операторов.

При фиксированном числе машин в ВС следует из этого множества отобрать n операторов (n — число машин в ВС). Если множество допустимых операторов состоит из m операторов и $m \leq n$, то для фактической реализации можно отобрать все допустимые операторы. В противном случае надо произвести выбор n операторов из m , а оставшиеся $m - n$ операторов включить в множество допустимых операторов, которое будет формироваться на следующем шаге процесса вычислений. Для отбора n операторов можно воспользоваться следующим правилом приоритета: в первую очередь отбираются операторы, имеющие в соответствующих строках элемент, равный трем (или элементы, равные трем), во вторую очередь выбираются операторы, имеющие в соответствующих строках элементы, равные двум, в третью очередь отбираются операторы с единичными элементами в соответствующих строках, наконец, в последнюю очередь отбираются для реализации операторы, которым в главной матрице сопоставляются нулевые строки. При прочих равных условиях в первую очередь отбираются операторы, содержащие в сопоставленных им строках большее число единиц.

Распределение операторов по машинам системы при синхронном режиме работы системы может происходить, например, по методам, описанным в [1] или [2], [5]. Для применимости этих методов достаточно иметь в распоряжении лишь ту информацию, которая содержится в матрице времен. При асинхронном режиме работы системы распределение операторов происходит по

мере обращения машин системы в память системы из множеств допустимых операторов, используя правило приоритетного выбора, описанное выше. Более полное описание организации допустимого множества операторов при асинхронном режиме дано в работах [3] и [6]. После начала выполнения распределенных операторов в главной матрице обращаются в ноль все ненулевые элементы в строках, соответствующих этим операторам.

Для организации получения результирующей информации от машин системы, организации обмена информацией между машинами, а также для формирования зон числового окружения для реализуемых функциональных операторов используются списки исходных, окончательных, входных и выходных идентификаторов.

Более подробное описание управления вычислительным процессом с помощью ЯПФ содержится в работе [9].

Перейдем к изложению основного материала настоящей работы: описанию алгоритма перевода записи алгоритма в виде программы, состоящей из псевдокоманд в ЯПФ.

Псевдокоманды представляют собой слова следующей структуры:

идентификатор псевдокоманды	идентификатор операции	идентификаторы операндов, их тип и размеры	идентифика- ры результа- тов, их тип и размеры.
--------------------------------	---------------------------	--	--

При идентифицировании всех данных в псевдокомандах используется принцип однозначности идентификации. Другими словами, всякий идентификатор используется только для одной переменной. Это условие не является принципиальным, но существенно облегчает реализацию алгоритма перехода от программы, записанной на языке псевдокоманд, к ЯПФ.

Программа в псевдокомандах просматривается однократно снизу вверх. При этом фиксируется следующая информация: по идентификаторам операций выявляются псевдокоманды начало цикла, конец цикла и условный переход, по идентификаторам операндов и результатов составляются все четыре списка идентификаторов, сечение которых происходит по мере появления указанных выше псевдокоманд, подсчитывается время и объем программной и числовой информации, соответствующей кускам программы, находящимся между указанными выше псевдокомандами (этот подсчет

происходит на основании таблиц интерпретации псевдокоманд машинными командами, времени выполнения машинных команд и размеров операндов и результатов).

За счет выявления точек ветвления программы образуются естественные части программы, которые являются либо линейными участками, либо циклами, не входящими внутрь других циклов. Функциональные связи между ними устанавливаются на основе анализа пересечения списков входных и выходных идентификаторов, а управляющие связи — за счет анализа идентификаторов, результатов в псевдокомандах условного перехода и конца цикла. Этот анализ позволяет сформировать главную матрицу, в которой роль операторов играют естественные части.

Одним из требований эффективной реализации ЯПФ на вычислительных системах является требование использования только оперативного ЗУ машин системы. Это в свою очередь накладывает ограничение на допустимый объем естественных частей. Если естественная часть превышает допустимый объем, то её необходимо разрезать на более мелкие части. Если эта естественная часть представляет собой линейный участок, то для его разрезания можно воспользоваться методом, описанным в [4], если же эта естественная часть есть цикл, то задача несколько усложняется.

Прежде всего проверяется тип цикла (тип цикла отмечается в идентификаторе псевдокоманды "начало цикла"). Если цикл является Б-циклом, т.е. циклом с искусственно упорядоченным прохождением по значениям кратной переменной (например, цикл, возникающий при вычислении таблицы значений функции $y = f(x)$ с шагом по x , равным h на отрезке $[a, b]$), то проверяется, нельзя ли уменьшить объем информации за счет уменьшения области изменения параметров цикла. Если это не удается, то проверяется, нет ли внутри данного цикла другого цикла, т.е. нельзя ли данный цикл представить как совокупность двух (или одного) линейных участков и цикла. Если этой возможности нет, то происходит разрезание цикла по методам разрезания линейных частей.

Если цикл является А-циклом, т.е. циклом с естественно упорядоченным прохождением по значениям кратной переменной (например, цикл вычисления $y = \Phi(x)$ с заданной точностью по итерационной формуле $y_n = F(y_{n-1}, x), y_0 = \Phi(x)$),

то сразу же проверяется возможность представления его в ви-

де совокупности линейных участков и внутреннего цикла.

Если после разрешения цикла его не удастся распараллелить так, чтобы временные объемы получающихся кусков были бы достаточно велики (больше некоторого τ , определяемого скоростью работы устройства управления в ВС и скоростью работы каналов связи [8]), то такой цикл в главной матрице представляет одним столбцом и одной строкой. Если же цикл сам имеет структуру некоторой ЯПФ, то при управлении им в процессе реализации ЯПФ от главной матрицы отчуждается циклическая главная матрица, содержащая такую же информацию, что и часть главной матрицы, сопоставляемая циклическому участку. После каждого прохождения цикла, которое не является еще последним, циклическая главная матрица с помощью главной матрицы восстанавливается в первоначальном виде. Только после окончания цикла происходит обращение в ноль ненулевых элементов в соответствующих строках главной матрицы.

Таким образом, нами описан переход от языка псевдокоманд к языку ЯПФ и показана принципиальная возможность использования языка ЯПФ в качестве системного языка в вычислительных системах.

Л И Т Е Р А Т У Р А

1. Верхотуров Е.П. О распределении кусков программы по машинам вычислительной системы. В сб. "Научные и практические проблемы больших систем, ч. I, II, МДНТП, 1967, стр. 82-86.
2. Голубев-Новожилов Ю.С. Многомашинные комплексы вычислительных средств, М., "Сов. Радио", 1967.
3. Котов В.Е., Нариньян А.С. Преобразование операторных схем в асинхронные программы. В сб. "Вычислительные системы", Труды симпозиума, Новосибирск, 1967, стр. 97-101.
4. Малушински Я., Поспелов Д.А. Об одном методе выделения в схеме программы независимых кусков. Труды МЭИ, вып. 53, 1964, стр. III-II6.
5. Мацящик К., Поспелов Д.А. Оптимальное попарное распределение программы на параллельно работающие вычислительные устройства. В сб. "Сети передачи информации и их автоматизация", М., "Наука", 1965, стр. 75-79.

6. Осипова М.А. Некоторые проблемы построения машинно-ориентированного языка для комплекса вычислительных машин с общей памятью. Диссертация на соискание ученой степени кандидата технических наук, М., 1966.
7. Поспелов Д.А. Классификация структур алгоритмов, реализуемых на вычислительных системах.- Известия АН СССР, Техническая кибернетика, № 5, 1967, стр. 137-144.
8. Поспелов Д.А. Оценка производительности вычислительной системы, состоящей из обычных вычислительных машин.- Известия АН СССР. Техническая кибернетика, № 5, 1966, стр. 98-112.
9. Поспелов Д.А. Управление в больших вычислительных системах, Труды конференции по большим системам, т. I. М., "Наука", 1968 (в печати).
10. Anderson . Program structures for parallel processing. - Commun. of ACM, 1965, № 12, p. 780-788.
11. Naur P. The programming of a system for automatic segmentation of programs within an ALGOL compiler (GIER ALGOL), Výzkumný ústav matematických strojů, Praha, 1964, 14 p.
12. Nuding E. A language designed for communication between computers of different types, - Symbol. Languages Data Process, N.Y. - London, 1962, p. 791-795.