УДК 681.5.62–5 + 681.5.681.3

ПОСТРОЕНИЕ МНОЖЕСТВА СЕЧЕНИЙ В ЗАДАЧЕ ОПТИМАЛЬНОГО РАЗБИЕНИЯ ПАРАЛЛЕЛЬНЫХ УПРАВЛЯЮЩИХ АЛГОРИТМОВ

Ватутин Э.И., Зотов И.В., Титов В.С.
Курский государственный технический университет

В статье рассматриваются особенности реализации этапа построения множества смежных сечений параллельного управляющего ациклического алгоритма в рамках параллельно-последовательного метода формирования субоптимальных разбиений. Приводятся алгоритмы проведения преобразований над R-выражениями. Даются примеры выполнения таких преобразований.

Some aspects of the construction of a set of adjacent sections for a parallel control acyclic algorithm, a phase of the parallel-sequential suboptimal separation method, are under consideration. Algorithms for converting R-expressions are given. And examples for the convertions are presented.

Одной из эффективных реализаций устройств логического управления функционально сложными объектами являются микроконтроллерные сети [1], известные также как микропрограммные мультимикроконтроллеры (МПММК) [2]. МПММК представляют собой параллельные структуры, формируемые из множества однотипных СБИС – микропрограммных микроконтроллеров (МПМК).

В работе рассмотрена одна из подзадач синтеза МПММК, направленная на оптимальное представление алгоритмов логического управления в виде сети компонентных алгоритмов, распределенных между отдельными МПМК микроконтроллера (выбор разбиения) [3,4,6]. Один из алгоритмов решения этой задачи (названный параллельно-последовательным) представлен в [2]; он учитывает ряд важных ограничений, состав и содержание которых зависят от архитектуры СБИС МПМК, технологических ограничений, структурной организации МПММК, дисциплины взаимодействия микроконтроллеров и выполняет квазипараллельное формирование компонентных алгоритмов.

Суть параллельно-последовательного алгоритма сводится к следующему. Исходная граф-схема алгоритма управления (ПарГСА) G^0, удовлетворяющая набору ограничений, приводится к ациклическому виду с использованием $\overline{\omega}$-преобразования, не изменяющего отношения параллельности [3] между вершинами. Затем ПарГСА ставится в соответствие система выражений Ξ, которая используется для формирования множества смежных сечений (сечением является группа вершин ПарГСА, элементы которой не находятся в отношении связи [3,6]). Первоначально формируется базовое сечение с использованием *R*-алгоритма, после чего начинается перебор смежных сечений “вверх” (получение *u*-сечений) и “вниз” (получение *d*-сечений). Полученное разбиение алгоритма на сечения используется при формировании блоков разбиения (компонентных алгоритмов).

Формирование множества сечений начинается с выделения максимального (базового) сечения ПарГСА, заданной в виде системы выражений Ξ. В результате преобразования исходной системы Ξ по правилам *R*-алгоритма получается система

Ξ^*, содержащая два выражения вида $a_0 \to R_\gamma, R_\gamma \to a_k$, причем R_γ является искомым базовым сечением. Преобразование заключается в применении одного из трех правил (*u*-поглощение, *d*-поглощение, ψ-перегруппировка), в результате чего система видоизменяется. Применение правил *u*- и *d*-поглощения ведет к уменьшению числа выражений на единицу, ψ-перегруппировка изменяет структуру одного из выражений.

Непосредственно после выделения базового сечения возникает необходимость в нахождении множества смежных сечений алгоритма [2]. Для этого организуется процесс построения смежных сечений в двух направлениях (“вверх”, т.е. в сторону начальной вершины a_0, и “вниз”, т.е. в сторону конечной вершины a_k), начиная с базового. Процесс получения множества смежных сечений представляет собой итеративное применение процедур Π_u и Π_d к текущему сечению, что позволяет получить для текущего сечения т.н. *u*- и *d*-сечения. Применение процедур заключается в выделении подмножества выражений системы, являющихся конструктивными подмножествами текущего сечения, и выполнению подстановки. Применение процедур Π_u и Π_d происходит до тех пор, пока сечение, полученное на очередном шаге, не будет включать начальную либо конечную вершины алгоритма.

Хранение и обработка *R*-выражений производится в виде деревьев. Например, *R*-выражение $a_7 \,|\, a_8 \,|\, (a_9 \bullet a_{10})$ будет представлено в виде дерева следующим образом:

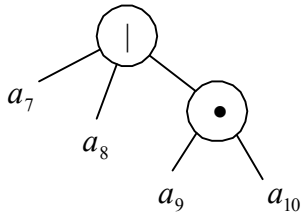

Рис.1 Пример дерева, соответствующего R-выражению

Узлам дерева соответствуют отношения между вершинами (параллельность “•” и альтернатива “|”), листьям – вершины алгоритма. Деревья (*R*-выражения) хранятся в виде массива записей, причем каждая запись содержит информацию о типе вершины, о ссылках на потомков и о ссылке на предка (фактически ссылка представляет собой номер элемента массива). Связи между элементами дерева организованы таким образом, что возможно движение по дереву в двух направлениях: от корня к листьям и от листьев к корню.

Операции над системой выражений (*u*-, *d*-поглощения, ψ-перегруппировка, процедуры Π_u и Π_d) допускают разбиение на более простые операции над *R*-выражениями. К их числу относятся:

- проверка отношения нестрогого включения $R_i [\subseteq] R_j$;
- сравнение субсечений (поддеревьев);
- вычисление ω-мощности сечения;
- раскрытие скобок в *R*-выражениях;
- перегруппировка подвыражения (как прямое, так и обратное преобразование);
- подстановка деревьев;

Вычисление ω-мощности сечения происходит рекурсивно от корня к листьям дерева (сверху вниз), при этом вычисление сводится к проверке типа узла и выполнению определенных действий:

Тип узла дерева	Алгоритм вычисления ω-мощности
$\mid$	Максимальная ω-мощность подвыражения
$\bullet$	Сумма ω-мощностей подвыражений
L	Количество листьев в наборе

Примеры R-выражений и их ω-мощности:

Выражение	ω-мощность
$a_1 \mid a_2$	1
$a_1 \bullet a_2$	2
$a_1 \mid (a_2 \bullet a_3) \mid a_4$	2
$a_1 \bullet (a_2 \mid a_3) \bullet a_4$	3

Сравнение субсечений (поддеревьев) основано на понятии эквивалентности деревьев. Деревья считаются эквивалентными, если они имеют одинаковую структуру, т.е. каждому узлу одного дерева взаимно однозначно соответствует узел другого дерева, причем узлы совпадают по типу и по структуре поддеревьев, корнями которых они являются. Сравнение поддеревьев осуществляется по рекурсивному алгоритму.

Проверка отношения нестрогого включения. В процессе поиска деревьев для *u*- или *d*-поглощения возникает необходимость в выяснении того, является ли одно дерево полностью или частично поддеревом другого, и определении того, какая часть дерева полностью эквивалентна, а какая полностью неэквивалентна. Дерево *A* не полностью эквивалентно дереву *B* в том случае, если дерево *A* является поддеревом дерева *B*, однако у элемента дерева *B*, соответствующего корню дерева *A*, имеются потомки, не входящие в состав дерева *A*. При реализации подстановок деревьев неэквивалентные потомки остаются у корня. Пример деревьев, находящихся в отношении неполной эквивалентности:

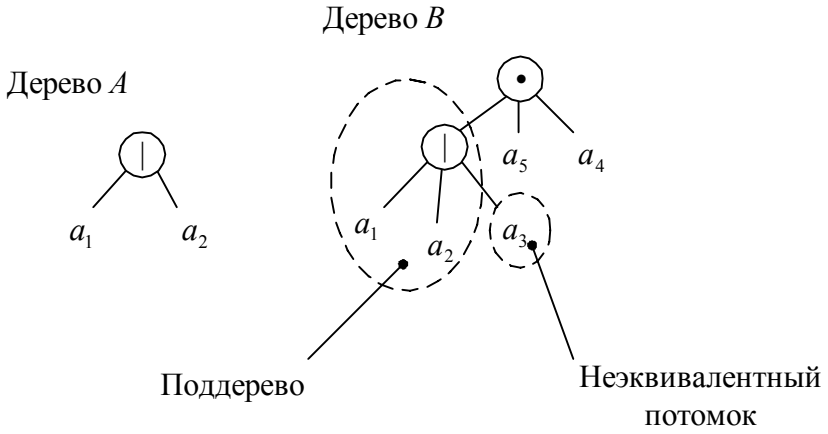

Рис.2. Пример неполной эквивалентности деревьев

Проверка деревьев на предмет неполной эквивалентности производится рекуррентно. Алгоритм проверки в целом похож на алгоритм сравнения деревьев.

Для более эффективного проведения проверки каждой вершине дерева B сопоставляется числовая оценка. Вершину считается кандидатом в эквивалент корня дерева, если оценка вершины и оценка корня дерева *A* совпадают. Алгоритм формирования оценок заключается в восстановлении маршрута от листа к корню в

дереве A в виде последовательности типов пройденных узлов (“•”, “|”, “•”, “•”) в направлении снизу-вверх, после чего в дереве B находятся все аналогичные маршруты, и оценки вершин, в которых происходит окончание маршрута, увеличиваются на единицу. Затем из дерева B выбираются вершины, оценки которых совпадают с оценкой корня дерева A, и для них проводится рекурсивное сравнение деревьев.

Для вершин-кандидатов необходимо проведение дополнительной детальной рекурсивной проверки с целью выяснения структуры дерева. Без рекурсивной проверки обойтись невозможно, т.к. в общем случае возможны варианты деревьев, в которых оценка вершины будет соответствовать оценке корня, однако деревья не будут эквивалентны.

Раскрытие скобок в R-выражениях производится по мере необходимости после подстановок. Например, если в результате выполнения подстановки получается выражение $a_1 \bullet (a_2 \bullet a_3 \bullet (a_4 \mid a_5))$, то оно должно быть преобразовано к виду $a_1 \bullet a_2 \bullet a_3 \bullet (a_4 \mid a_5)$.

С точки зрения преобразования деревьев раскрытие скобок сводится к нахождении пары узлов предок-потомок, имеющих одинаковый тип. Результатом преобразования является поглощение потомка предком.

Подстановка деревьев является одной из наиболее часто используемых операций. В результате проверки пригодности структуры деревьев для проведения какой-либо операции преобразования выясняется, какой узел является эквивалентом корня, находятся ли поддерево и искомое дерево в отношении полной эквивалентности, определяется группа узлов, которые останутся у эквивалента корня после подстановки. Основная сложность данной операции заключается в том, что возможны 8 различных типов подстановок, 6 из которых встречаются при выполнении R-алгоритма и еще 2 при выполнении процедур $П_u$ и $П_d$. Примеры систем выражений и подстановок:

$$\begin{cases} a_1 \to a_2 \\ a_2 \to a_3 \bullet a_4 \end{cases} \Rightarrow \quad a_1 \to a_3 \bullet a_4, \qquad \begin{cases} a_1 \to a_2 \mid a_3 \\ a_2 \to a_4 \bullet a_5 \end{cases} \Rightarrow \quad a_1 \to (a_4 \bullet a_5) \mid a_3 .$$

Перегруппировка подвыражения вызвана неоднозначностью описания фрагментов алгоритмов с параллельными альтернативами. Существует 2 типа выражений (a- и p-форма), причем если при поиске базового сечения возможно появление только преобразования $a \to p$-типа, то при формировании множества смежных сечений возможно наличие и обратного ($p \to a$) преобразования. Осуществление преобразований возможно с учетом как структуры RT- и ST-деревьев, так и анализа матрицы отношений вершин.

R-алгоритм в общем заключается в итеративном применении операций u-, d-поглощения и ψ-перегруппировки к исходной системе с целью преобразования ее к т.н. нередуцируемой ω-форме, которая содержит 2 выражения. На каждом шаге происходит подстановка одного R-выражения в другое, при условии, что полученное выражение будет иметь не меньшую ω-мощность, чем его предшественники. При реализации перебора сечений условий на ω-мощность не накладывается, что фактически приводит к перебору всех вершин алгоритма в составе сечений.

В ходе выполнения преобразования системы происходит запись операций в порядке их применения в текстовой файл. Рассмотрим пример нахождения множества сечений (протокол преобразований):

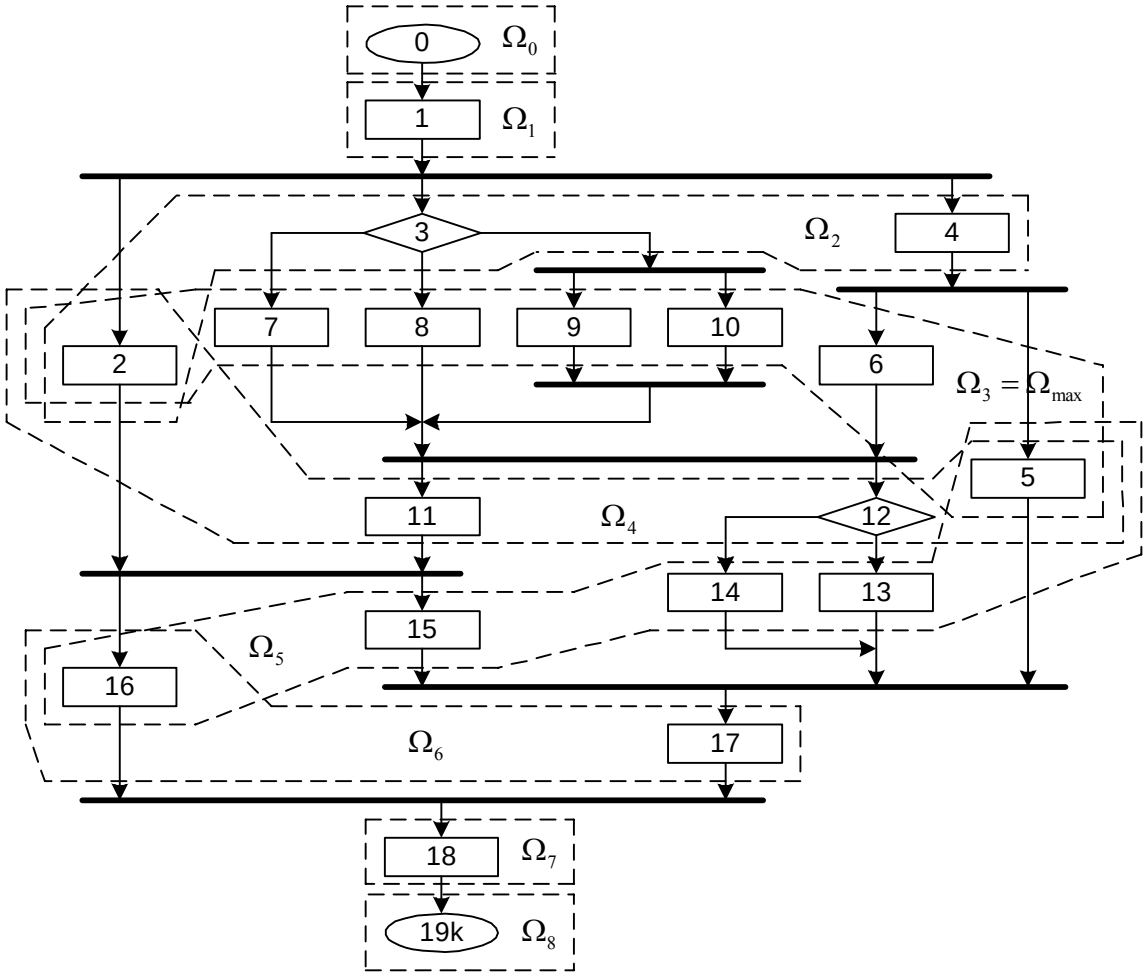

Рис.5. Разбиение ПарГСА на сечения

Исходная система

0: a0 -> a1
1: a1 -> a2*a3*a4
2: a3 -> a7|a8|(a9*a10)
3: a4 -> a5*a6
4: a6*(a7|a8|(a9*a10)) -> a11*a12
5: a2*a11 -> a15*a16
6: a12 -> a13|a14
7: a5*a15*(a13|a14) -> a17
8: a16*a17 -> a18
9: a18 -> a19

———————— Поиск базового сечения ————————

u-поглощение 0, 1:

0: a0 -> a2*a3*a4
1: a3 -> a7|a8|(a9*a10)
2: a4 -> a5*a6
3: a6*(a7|a8|(a9*a10)) -> a11*a12
4: a2*a11 -> a15*a16
5: a12 -> a13|a14
6: a5*a15*(a13|a14) -> a17
7: a16*a17 -> a18
8: a18 -> a19

u-поглощение 0, 1:

0: a0 -> a2*a4*(a7|a8|(a9*a10))
1: a4 -> a5*a6
2: a6*(a7|a8|(a9*a10)) -> a11*a12
3: a2*a11 -> a15*a16
4: a12 -> a13|a14
5: a5*a15*(a13|a14) -> a17
6: a16*a17 -> a18
7: a18 -> a19

u-поглощение 0, 1:

0: a0 -> a2*a5*a6*(a7|a8|(a9*a10))
1: a6*(a7|a8|(a9*a10)) -> a11*a12
2: a2*a11 -> a15*a16
3: a12 -> a13|a14
4: a5*a15*(a13|a14) -> a17
5: a16*a17 -> a18
6: a18 -> a19

u-поглощение 1, 3:

0: a0 -> a2*a5*a6*(a7|a8|(a9*a10))
1: a6*(a7|a8|(a9*a10)) -> a11*(a13|a14)
2: a2*a11 -> a15*a16
3: a5*a15*(a13|a14) -> a17
4: a16*a17 -> a18
5: a18 -> a19

d-поглощение 3, 4:

0: a0 -> a2*a5*a6*(a7|a8|(a9*a10))
1: a6*(a7|a8|(a9*a10)) -> a11*(a13|a14)
2: a2*a11 -> a15*a16
3: a5*a15*a16*(a13|a14) -> a18
4: a18 -> a19

d-поглощение 2, 3:

0: a0 -> a2*a5*a6*(a7|a8|(a9*a10))
1: a6*(a7|a8|(a9*a10)) -> a11*(a13|a14)
2: a2*a5*a11*(a13|a14) -> a18
3: a18 -> a19

d-поглощение 1, 2:

0: a0 -> a2*a5*a6*(a7|a8|(a9*a10))
1: a2*a5*a6*(a7|a8|(a9*a10)) -> a18
2: a18 -> a19

d-поглощение 1, 2:

0: a0 -> a2*a5*a6*(a7|a8|(a9*a10))
1: a2*a5*a6*(a7|a8|(a9*a10)) -> a19

Базовое сечение:

a2*a5*a6*(a7|a8|(a9*a10))

w-мощность:

5

Перебор u-сечений

Множество подходящих выражений: {2,3} u-сечение: a2*a3*a4
Множество подходящих выражений: {1} u-сечение: a1
Множество подходящих выражений: {0} u-сечение: a0

Перебор d-сечений

Множество подходящих выражений: {4} d-сечение: a2*a5*a11*a12
Множество подходящих выражений: {5,6} d-сечение: a5*a15*a16*(a13|a14)
Множество подходящих выражений: {7} d-сечение: a16*a17
Множество подходящих выражений: {8} d-сечение: a18
Множество подходящих выражений: {9} d-сечение: a19

Сечения (в конструктивной форме):

a0
a1
a2*a3*a4
a2*a5*a6*(a7|a8|(a9*a10))
a2*a5*a11*a12
a5*a15*a16*(a13|a14)
a16*a17
a18
a19

Рассмотренная методика позволяет получать разбиение алгоритма управления на сечения в рамках параллельно-последовательного метода формирования оптимального разбиения управляющих алгоритмов. В дальнейшем, при формировании блоков разбиения, в рамках рассматриваемого подхода используется информация о принадлежности вершин какому-либо из сечений.

Литература

1. Харченко В.С., Никольский С.Б., Сазонов А.Е. Один подход к синтезу дискретных микроконтроллерных сетей // А и ВТ. – 1989. – №4. – С. 87–95
2. Организация и синтез микропрограммных мультимикроконтроллеров / Зотов И.В. и др. – Курск: ГУИПП «Курск», 1999. – 368 с.
3. Баранов С.И., Журавина С.Н., Песчанский В.А. Метод представления параллельных граф-схем алгоритмов совокупностями последовательных граф-схем // А и ВТ. – 1984. – №5. – С. 74–81.
4. Баранов С.И., Журавина С.Н., Песчанский В.А. Обобщенный метод декомпозиции граф-схем алгоритмов // А и ВТ. – 1982. – №5. – С. 43–51
5. Харченко В.С., Кальченко С.Б., Сазонов А.Е. Декомпозиция параллельных матричных схем алгоритмов в задачах синтеза микроконтроллерных сетей // А и ВТ. – 1990. – №4. – С. 81–89
6. Зотов И.В., Колосков В.А., Титов В.С. Выбор оптимальных разбиений алгоритмов при проектировании микроконтроллерных сетей // А и ВТ. – 1997. – №5. – С.51–62