ГЕНЕРАЦИЯ СЛУЧАЙНЫХ АЛГОРИТМОВ В ЗАДАЧЕ ПОСТРОЕНИЯ РАЗБИЕНИЙ ПАРАЛЛЕЛЬНЫХ УПРАВЛЯЮЩИХ АЛГОРИТМОВ

Eduard I. Vatutin

Department of Computer Science
Kursk State Technical University
50 Let Oktyabrya, 94, 305040, RUSSIA
Tel: +7(4712) 56-43-13, E-mail: evatutin@rambler.ru

Abstract — *Дается описание алгоритма, позволяющего осуществлять автоматическую генерацию и визуальное отображение выборки случайных алгоритмов логического управления с заданными параметрами. Приводятся примеры сгенерированных алгоритмов.*

1. INTRODUCTION

Одним из перспективных подходов к созданию систем логического управления является их синтез на основе микроконтроллерных сетей (микропрограммных мультимикроконтроллеров) [1]. При синтезе подобных систем в рамках рассматриваемого подхода выделяется целый ряд задач, требующих своего решения. Одной из них является задача выбора оптимального разбиения заданного параллельного алгоритма логического управления на последовательные подалгоритмы ограниченной сложности в соответствии с технологическими и функциональными ограничениями. Задача относится к классу NP-полных и не может быть решена точно для алгоритмов размерности более чем ~15 вершин (приблизительно) ввиду чрезмерных временных затрат. Реальные алгоритмы управления обладают куда большей размерностью. Для ее решения известен ряд эвристических методов (например, [2, 3]), один из которых предложен авторами [4]. При построении решения рассмотренные методы используют различные методики; некоторые из методов не оптимизируют такие важные критерии, как, например, интенсивность межблочных взаимодействий [1], поэтому вызывает интерес проведение сравнения методов между собой с целью выявления наилучших из них по заданным критериям. Для этого необходимо обладать достаточно большим набором исходных данных (в данном случае – алгоритмов логического управления), чтобы провести объективное сравнение. Число реальных примеров алгоритмов управления, также как и число алгоритмов, которые можно придумать в ограниченный интервал времени, существенно ограничено и об объективности сравнения методов не может быть и речи, поэтому вызывает интерес автоматическая генерация заданного количества алгоритмов с установленными параметрами. Следует отметить, что результаты сравнения, полученные на выборке случайных алгоритмов, следует проверить и на известных реальных примерах алгоритмов управления.

2. ПРЕДСТАВЛЕНИЕ АЛГОРИТМА В ВИДЕ ДЕРЕВА ФРАГМЕНТОВ

В составе любого корректного алгоритма можно выделить набор типовых фрагментов, из которых он составлен. К ним относятся:

- начальный фрагмент (BE);
- линейный участок (LW);
- условное (альтернативное) ветвление (ALT);
- параллельный фрагмент (PAR);
- различные типы циклов – с пред- и постусловием, с прерыванием (LB, LA, LM).

(При данном подходе параллельные альтернативы и параллельные циклы не рассматриваются (не участвуют в генерации), т.к. их можно заменить на перечисленные выше последовательные фрагменты в комбинации с параллельным фрагментом, как показано на рис. 1).

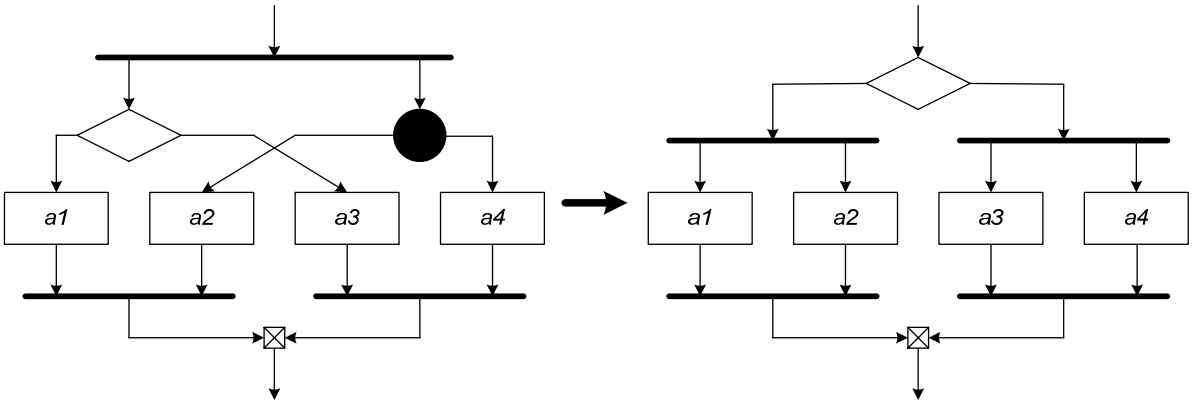

Рис. 1. Замена параллельного альтернативного фрагмента

Любой алгоритм (возможно с небольшими преобразованиями, не меняющими порядка выполнения операторов и логики алгоритма), обычно представляемый в виде графа вершин и дуг алгоритма, может быть представлен в виде дерева перечисленных выше фрагментов (рис. 2).

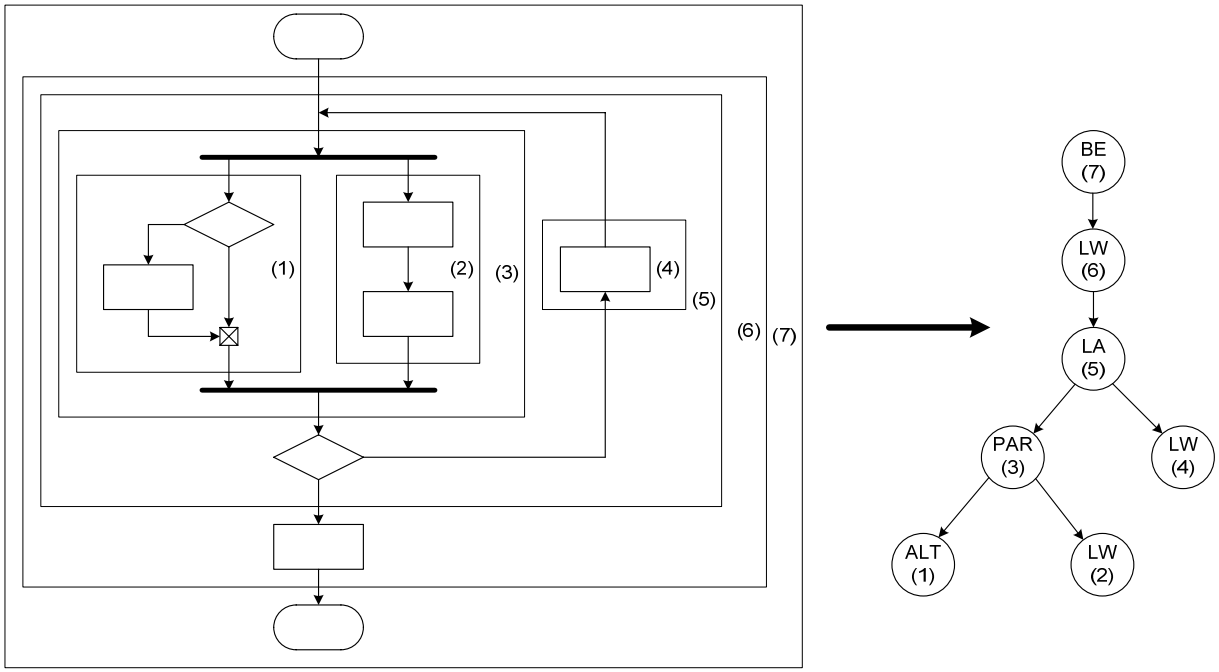

Рис. 2. Представление алгоритма в виде дерева фрагментов

3. АЛГОРИТМ СИНТЕЗА СЛУЧАЙНЫХ УПРАВЛЯЮЩИХ АЛГОРИТМОВ

С учетом возможности представления алгоритма управления в виде дерева фрагментов алгоритм случайной генерации можно представить в виде двух укрупненных этапов:

1. Построение дерева фрагментов сверху вниз (от корня к листьям).
2. Построение случайного алгоритма путем укрупнения фрагментов снизу вверх (от листьев к корню).

Данный специфичный порядок обхода дерева обусловлен тем, что результирующий случайный алгоритм необходимо представлять не просто в виде графа, а в виде набора фигур [5], чтобы сохранить возможность графического отображения сгенерированного алгоритма.

Первый этап генерации алгоритма можно представить в виде следующего алгоритма:

1. Добавить в дерево фрагментов начальный фрагмент (BE).
2. Выбрать среди имеющихся в дереве фрагментов родительский фрагмент случайным образом пропорционально количеству его свободных дуг. (Под свободными понимаются дуги, которые могут участвовать в процессе заменены на дочерний фрагмент. Количество свободных дуг изначально определяется типом фрагмента и его параметрами и уменьшается на единицу при каждом добавлении дочернего фрагмента (рис. 3)).
3. Выбрать тип добавляемого фрагмента случайным образом пропорционально вероятностям встречаемости фрагментов различных типов, определить его параметры (количество ветвей, количество вершин, вероятности активации дуг и т.д.). Добавить фрагмент в дерево.
4. Если выполнено условие завершения генерации (достигнуто заранее заданное количество суммарное вершин или количество фрагментов), конец алгоритма, в противном случае перейти к п. 2.

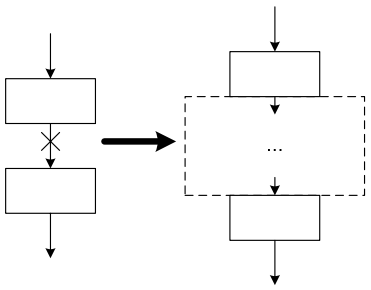

Рис. 3. Замена свободной дуги дочерним фрагментом

В результате выполнения первого этапа построено дерево фрагментов, которое может быть использовано непосредственно для синтеза алгоритма в виде графа. Алгоритм второго этапа выглядит следующим образом:

1. Пометить все фрагменты дерева как нерассмотренные.
2. Выбрать в дереве фрагмент, не имеющий нерассмотренных дочерних фрагментов. Сформировать его геометрическую структуру: подсчитать координаты вершин, дуг и дочерних фрагментов, определить геометрические размеры фрагмента (рис. 4). Настроить связи дуг дочерних фрагментов (если таковые имеются) с вершинами рассматриваемого фрагмента. Определить для дуг дочерних фрагментов значения вероятности активации и интенсивности передачи управления. Определить случайным образом состав микроопераций и логических условий операторных и условных вершин. Пометить фрагмент как рассмотренный.
3. Если рассмотрены все фрагменты, конец алгоритма, в противном случае перейти к п. 2.

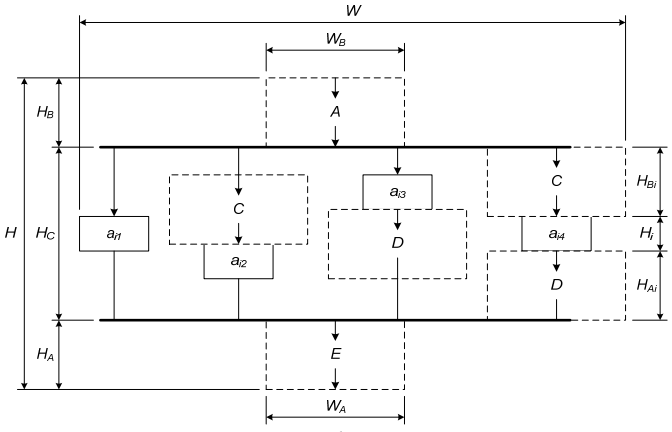

Рис. 4. Определение геометрической структуры фрагмента на примере параллельного фрагмента

Следует отметить, что при определении геометрических параметров фрагментов (рис. 4) в общем случае геометрические параметры дочерних фрагментов не совпадают:

$$H_i \neq H_j,\ W_i \neq W_j,\ \forall i, j \notin \{B, C, A, B_1, ..., B_n, C_1, ..., C_n, A_1, ..., A_n\},$$

что создает определенные затруднения при вычислении расположения ветвей, а также координат вершин и дуг.

После выполнения указанных выше действий получается алгоритм, представленный в виде графа, причем все его вершины и дуги имеют координаты на плоскости и могут быть без особых затруднений изображены на экране.

4. ПРИМЕРЫ СГЕНЕРИРОВАННЫХ СЛУЧАЙНЫХ АЛГОРИТМОВ УПРАВЛЕНИЯ

Для подтверждения работоспособности описанного выше алгоритма приведем несколько примеров ациклических алгоритмов, сгенерированных в соответствии с описанным выше алгоритмом, реализация которого входит в состав программной системы PAE [5].

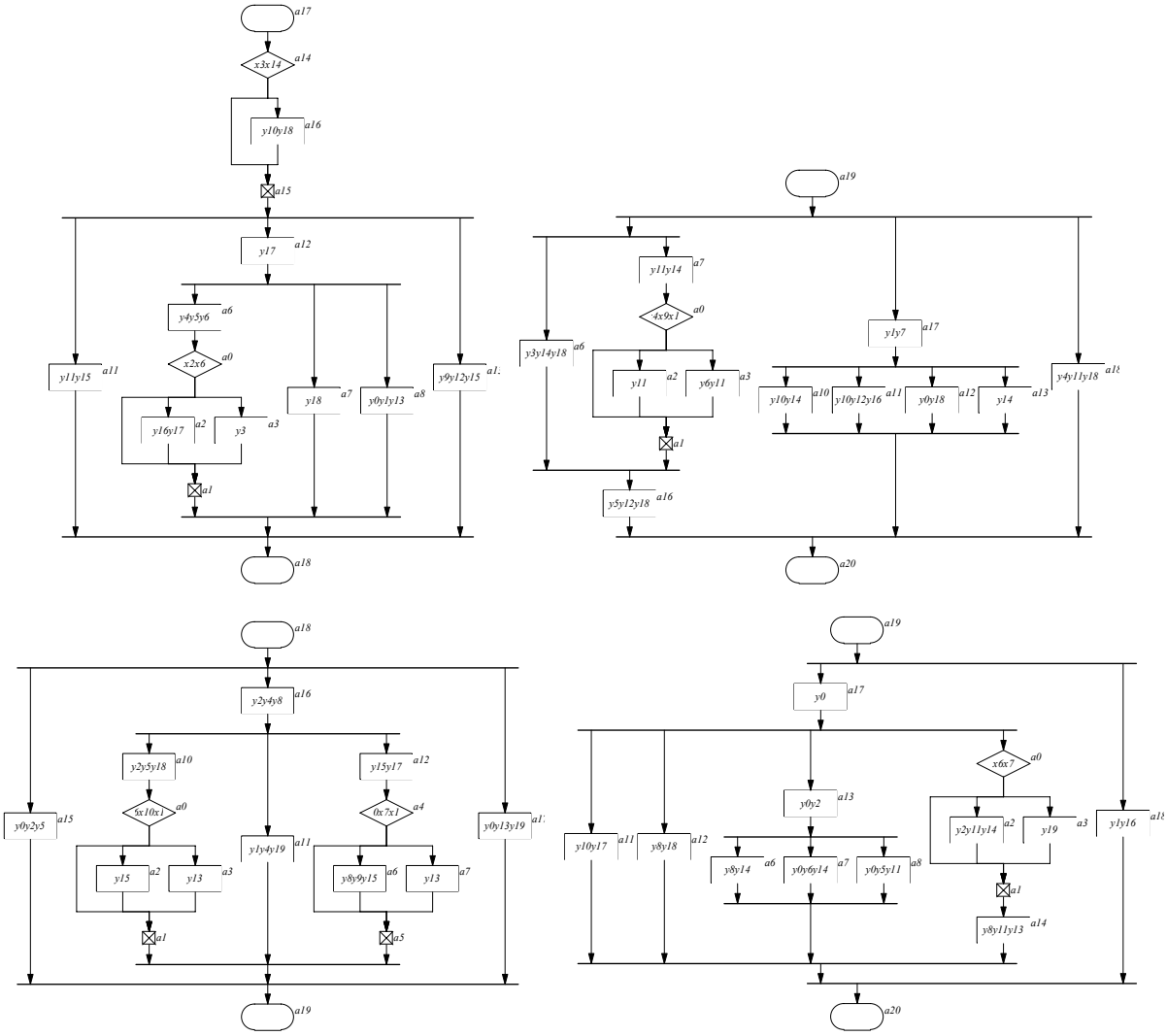

Рис. 5. Примеры сгенерированных случайных алгоритмов

5. ДОПУЩЕНИЯ

Следует отметить, что сгенерированные подобным образом алгоритмы слегка отличаются от реальных алгоритмов из-за некоторых допущений, полагаемых при генерации:

1. Длины линейных участков равновероятны.
2. Количества ветвей в составе альтернативных и параллельных фрагментов равновероятны.
3. Количества микроопераций и логических условий в вершинных равновероятны.
4. Вероятности встреча микроопераций и логических условий в вершинах равны.

Однако в целом принятые допущения не должны оказать серьезного влияния на объективность сравнения методов.

6. CONCLUSION

1. Разработан и протестирован алгоритм генерации выборки случайных алгоритмов управления произвольного объема.
2. Сгенерированный алгоритм представляется не просто в виде графа, а в виде, удобном для последующего графического вывода.
3. На основании разработанного алгоритма возможно проведение как сравнения методов построения разбиений, так и тестирование самих методов на предмет выявления ошибок.
4. В ходе подобной генерации алгоритма можно достаточно просто в вычислительном плане определить такую важную характеристику синтезированного алгоритма управления, как степень параллелизма.
5. Оценены скоростные характеристики алгоритма: генерация ~1500 алгоритмов из ~20 вершин за 1 секунду на компьютере с процессором Intel Celeron 850 МГц (CPUID=068Ah).

7. REFERENCES

[1] Организация и синтез микропрограммных мультимикроконтроллеров / Зотов И.В. и др. Курск: ГУИПП «Курск», 1999. 368 с.

[2] Баранов С.И., Журавина Л.Н., Песчанский В.А. Метод представления параллельных граф-схем алгоритмов совокупностями последовательных граф-схем // Автоматика и вычислительная техника. 1984. №5. С. 74–81.

[3] Закревский А.Д. Параллельные алгоритмы логического управления. Минск. ИТК НАН Б. 1999. 202 с.

[4] Ватутин Э.И., Зотов И.В. Метод формирования субоптимальных разбиений параллельных управляющих алгоритмов // Труды II международной конференции «Параллельные вычисления и задачи управления» PACO ’04 памяти Е.Г. Сухова. М.: Институт проблем управления им. В.А. Трапезникова РАН, 2004. С. 884–917.

[5] Ватутин Э.И., Зотов И.В. Программная система для построения разбиений параллельных управляющих алгоритмов // Труды V международной конференции «Идентификация систем и задачи управления (SICPRO’06)». М.: Институт проблем управления им. В.А. Трапезникова РАН, 2006. С. 2239–2250.