УДК 681.142.2

ПАРАЛЛЕЛЬНЫЕ СТРУКТУРЫ УПРАВЛЕНИЯ ВЫЧИСЛИТЕЛЬНЫМИ ПРОЦЕССАМИ В САПР

О.Ф. Немолочнов, А.Г. Зыков, В.И. Поляков, А.А. Македонский

Рассматриваются вычислительные процессы, описываемые множеством параллельных структур, получаемых предложенным методом структурирования, для верификации программ в САПР.
Ключевые слова: структурирование вычислительных процессов, графо-аналитические модели.

Введение

Любой вычислительный процесс (ВП) должен быть верифицирован, независимо от авторского стиля его проектирования и реализации. Сложность вычислительных процессов определяется не количеством всех операторов программы, а числом безусловных и условных точек ветвления ВП, которые определяют логику принятия решений в программе. Таким образом, для решения задач анализа и тестирования программ целесообразно выделить все условия ветвления ВП в отдельное множество и рассмотреть их независимо от линейных операторов. Каждое условие-предикат можно представить логической переменной, а множество условий-предикатов – логической функцией соответственно. Это позволит перейти от смысловых понятий условий-предикатов к переменным и выражениям на языке алгебры логики и использовать аппарат булевой алгебры [1]. Целью работы является разработка метода структурирования ВП по его графо-аналитической модели (ГАМ).

Структурирование вычислительных процессов

Одним из способов описания алгоритма работы программы является ее ГАМ. В модели каждая команда представляется вершиной графа, а дуги графа – переходами между командами. В таком графе дуги образуются либо в порядке следования операторов, либо с помощью условной и безусловной передачи управления, нарушающей естественный ход выполнения программы. На графе выделяются начальная и конечная вершины, содержащие соответственно точки входа и выхода из программы. Будем называть такой граф неструктурированным графом программы [2]. Более информативным является структурированный граф.

Алгоритм структурирования является многопроходным, состоящим из нескольких итераций. Итерации заканчиваются, когда все команды из анализируемого программного кода покрыты вершинами графа или если какие-либо команды покрыть вершинами графа не удалось. Вторая ситуация означает, что в программе существует так называемый мертвый код, который следует обрабатывать особо.

Под структурированием графа будем понимать извлечение вычислительного процесса из программного кода и его описание в виде ГАМ на основе знания системы команд вычислителя (процессора), для которого написана программа [3]. Вычислитель может быть физическим (реальным) или смоделированным (виртуальным). Вычислительный процесс в виде ГАМ может быть последовательно представлен с различной степенью детализации. Первоначально – в виде линейных или условных вершин и дуг, где линейные вершины реализуют различные алгебраические выражения (формулы) без разветвлений, условные вершины – условия-предикаты, а дуги реализуют связи между вершинами, вырабатываемые устройством управления конкретного процессора. Дальнейшее структурирование сводится к выделению замкнутых подмножеств вершин в виде линейных и параллельных структур (циклических и ациклических) и обращений к процедурам. Под циклом будем понимать некоторое подмножество команд, которое может исполняться два и более раза.

В итоге ГАМ вычислительного процесса представляется в виде композиции линейных и параллельных структур с выделенными процедурами как самостоятельными независимыми единицами, которые, в свою очередь, тоже могут потребовать процесса структурирования. Между структурами устанавливается соотношение включения (вложенности) друг в друга. В основу структурирования положен принцип замкнутости подмножеств соответственно для команд и вершин, т.е. в любое замкнутое подмножество нельзя войти иначе, чем через точку входа (*Tin*), и выйти иначе, чем через точку выхода (*Tout*). В общем случае, любая программа или процедура являются замкнутыми через точки входа и выхода параллельными структурами.

Извлечение ВП из программы путем дешифрации может быть построено на основе анализа машинного кода. Таким образом, объединение условных и линейных вершин в циклы, процедуры, параллельные структуры и другие возможные блоки позволит в конечном итоге структурировать ГАМ ВП и автоматизировать верификацию, контроль и диагностику программного продукта.

Для построения ГАМ ВП используются три уровня структуризации, реализуемые последовательно.

На первом уровне программа представлена в виде машинного кода, расшифровываемого вычислительной машиной, реальной или виртуальной, система команд которой априорно считается известной. Все команды разбиваются на две группы: линейные команды, содержащие в себе, помимо операций пре-

образования информации, неявную микрокоманду передачи управления следующей команде, и команды условной и безусловной передачи управления, содержащие в явном виде микрооперацию передачи управления по заданным в них адресам. Таким образом, команды порождают вершины ГАМ, связанные между собой дугами, задающими переходы между командами-вершинами.

На втором уровне структуризации последовательности команд объединяются в некоторые замкнутые множества в виде линейных и условных вершин графа вычислительного процесса. Вершины образуют граф, в котором линейные вершины порождают одну дугу, а условные вершины порождают две альтернативные дуги. В линейных вершинах осуществляется вычисление значений переменных по итеративным или рекуррентным формулам. В условных вершинах вырабатываются направления передачи управления и, в этом смысле, они являются условиями-предикатами. Построенный таким образом граф является бинарным или булевым графом, в котором дуги графа задают только связи между вершинами и, следовательно, не являются нагруженными.

На третьем уровне структуризации, который фактически является структурированием вычислительного процесса, вершины графа второго уровня объединяются в некоторые замкнутые множества, образующие линейные и параллельные структуры. Отсюда следует, что линейные структуры являются вырожденными, так как содержат только один путь, и, следовательно, совпадают по определению с линейными вершинами второго уровня структуризации.

Для построения математического описания ГАМ вычислительного процесса, порождаемого программой при интерпретации ее команд вычислительной машиной, предложена итерационно-рекурсивная модель. Данная модель является концептуальной дискретно-синхронизированной и отражает один шаг при вычислении значений переменных в прямом направлении (итерация от точек входа к точкам выхода) и обратном направлении (рекурсия) от точек выхода к точкам входа. Для всех переменных, вычисляемых в линейных и параллельных структурах, можно построить итерационно-рекурсивные покрытия специального вида. Такие покрытия содержат в себе кубическую часть, описывающую условия вычисления некоторой заданной переменной, и линейные формулы. Множество линейных и параллельных структур и связей между ними образуют структурированную графоаналитическую модель вычислительного процесса.

Структурирование графа программы позволяет:

- уменьшить его размерность по числу вершин и дуг;
- перейти на основе выделенных условных вершин к построению булева графа;
- построить комплексные кубические покрытия переменных, вычисляемых программой;
- упростить организацию автономного (модульного) и сборочного тестирования программного обеспечения.

Параллельные структуры

Для ациклических и циклических параллельных структур ВП, реализованных программно, наблюдается аналогия с комбинационными и последовательностными логическими схемами, законы функционирования которых описываются кубическими покрытиями булевых функций.

Отличительной особенностью любой параллельной структуры является ее замкнутость по условиям-предикатам, т.е. совокупность условий-предикатов, образующих параллельные структуры, должна порождать тождественно истинное условие вычисления переменных по всем ветвям структуры. Это условие, наряду с условием одной точки входа и одной точки выхода, является основой при поиске и построении параллельных структур ВП программы.

Среди параллельных структур следует выделять особые случаи – циклы и процедуры. Для циклов следует находить линейную структуру – начало цикла, осуществляющую начальную установку параметров цикла, и параллельную структуру, образующую тело цикла, в которой отдельно выделено множество условных вершин, осуществляющих выход из цикла. Процедуры выделяются в самостоятельные структурные единицы, так как к ним возможны обращения из разных точек программы. Структурирование тел циклов и процедур осуществляется по тем же правилам с выделением линейных и параллельных структур.

Выделим два типа вычислений переменных в виде линейных и интервальных формул, которые в данном рассмотрении являются принципиально различными. Линейная формула вычисляет некоторую переменную на безальтернативной основе и реализуется в дальнейшем в виде последовательности операторов (машинных команд в исполнительных программах). Интервальная формула вычисляет одну и ту же переменную по двум или более линейным формулам в зависимости от некоторых заданных условий. Интервальная формула в дальнейшем реализуется в виде параллельной структуры, в которой могут существовать условия компенсации значений вычисляемой переменной. Указанная структура характеризуется одной точкой входа (точкой ветвления) и одной точкой выхода (точкой объединения ветвей). В программах эти точки реализуются с помощью команд безусловной и условной передачи управления, так как исполнительная программа записывается в виде линейной последовательности операторов, если она реализуется на одном вычислителе.

Заметим, что на ГАМ могут существовать параллельные ветви вычислений разных переменных в зависимости от некоторых условий. Но в таких структурах не возникают условия компенсации, и поэтому их анализ при поиске решений сводится к анализу обычных линейных формул в несколько усложненном варианте, связанном с вычислением условий. Также отметим, что циклы и обращения к процедурам сами по себе не создают параллельных структур и сводятся к линейным формулам, если в них, в свою очередь, нет внутренних интервальных формул. Таким образом, их анализ не требует специального рассмотрения.

Для примера рассмотрим ГАМ ациклического ВП с двумя условиями-предикатами, представленную на рисунке.

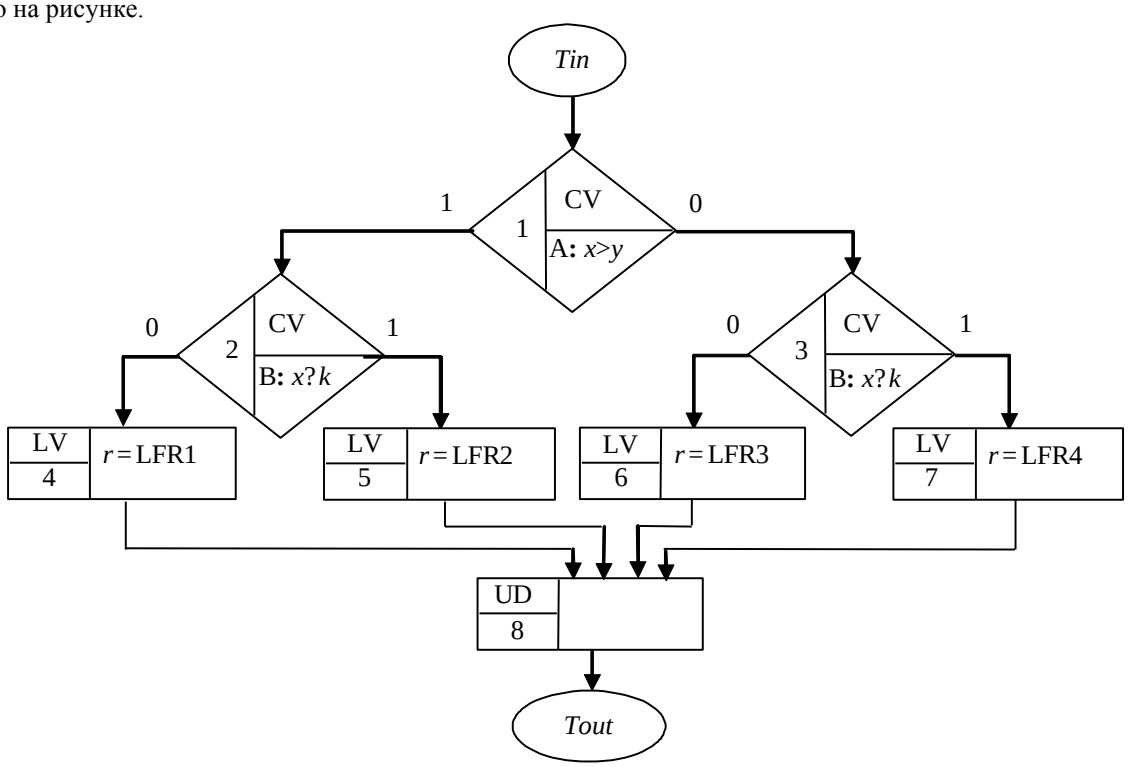

Рисунок. ГАМ с параллельной структурой вычислительного процесса,
где A и B – условия-предикаты, A определяется отношением $x > y$, а B – отношением $x \geq k$

Модель представлена в нотации, предложенной в [4]. Здесь CV – обозначение условных вершин, LV – линейных вершин и UD – объединяющих вершин.

AB	00	01	11	10
	LFR3	LFR4	LFR2	LFR1

Таблица. Карта Карно для ГАМ структуры на рисунке

В таблице приведен вариант вычисления переменной r при различных значениях условий-предикатов, которые определяются отношениями, представленными в следующей системе неравенств:

$$r = \begin{cases} \text{LFR1}, \text{ при } x > y \text{ и } x \leq k; \\ \text{LFR2}, \text{ при } x > y \text{ и } x \geq k; \\ \text{LFR3}, \text{ при } x \leq y \text{ и } x \leq k; \\ \text{LFR4}, \text{ при } x \leq y \text{ и } x \geq k. \end{cases} \quad (1)$$

Из (1) видно, что условия вычисления r по формулам заданы в избыточной форме, и, с учетом перестановок, существует 4 различных сочетания последовательного их вычисления. Соответственно, потенциально существует четыре различные параллельные структуры на графе вычислительного процесса, реализующие одну и ту же интервальную формулу (1). С помощью карты Карно можно, основываясь на конечных значениях условий A и B, перейти к вычислению необходимой формулы. Например: если условия-предикаты A и B являются истиной, то на выходе получим команду $r = \text{LFR2}$, но если условие A будет ложью ($x \leq y$) при том же условии B, то получим на выходе $r = \text{LFR4}$. Изменение порядка следования условий-предикатов A и B приводит к разнообразию программных реализаций.

Заключение

Построение структурированных графо-аналитических моделей программного кода путем выделения линейных и условных вершин с последующим объединением их в параллельные структуры позволяет упростить верификацию и сопровождение программных продуктов, что в итоге дает возможность повысить качество и надежность программного обеспечения. Структурирование ГАМ ВП, в свою очередь, снижает размерность задачи верификации и делает ВП более прозрачным для его системного анализа. Это упрощает процесс организации модульного, сборочного и системного тестирования программ. Синтез тестов по структурированным ГАМ позволяет определить наличие недекларированных возможностей и мертвого кода в программах и принять меры по предотвращению несанкционированного доступа и разного рода закладок в систему.

Литература

1. Немолочнов О.Ф., Зыков А.Г., Поляков В.И. Кубические покрытия логических условий вычислительных процессов и программ // Научно-технический вестник СПбГУ ИТМО. – 2004. – № 14. – С. 225–233.
2. Зыков А.Г., Немолочнов О.Ф. Поляков В.И., Сидоров А.В. Структурирование программ и вычислительных процессов на множество линейных и условных вершин // Научно-технический вестник СПбГУ ИТМО. – 2005. – № 19. – С. 207–212.
3. Немолочнов О.Ф., Зыков А.Г., Поляков В.И. Методы формализации графо-аналитических моделей вычислительных процессов программ // Труды Международных научно-технических конференций «Интеллектуальные системы» (AIS'07) и Интеллектуальные САПР (CAD-2007). – М.: Физматлит, 2007. – Т. 3. – С. 8–96.
4. Немолочнов О.Ф., Зыков А.Г., Поляков В.И., Осовецкий Л.Г., Сидоров А.В., Кулагин В.С. Итерационно-рекурсивная модель вычислительных процессов программ // Изв. вузов. Приборостроение. – 2005. – Т. 48. – № 12. – С. 14–20.

Немолочнов Олег Фомич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, доктор технических наук, профессор, зав. кафедрой, nemolochnov_o_f@mail.ru

Зыков Анатолий Геннадьевич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, кандидат технических наук, доцент, zykov_a_g@mail.ru

Поляков Владимир Иванович – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, кандидат технических наук, доцент, v_i_polyakov@mail.ru

Македонский Алексей Алексеевич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, аспирант, alexeymakedonskiy@gmail.com