УДК 004.054; 004.582

А.С. ПРИГОЖЕВ

Одесский национальный политехнический университет, Украина

ПОСТРОЕНИЕ КЛАССОВ ЭКВИВАЛЕНТНОСТИ С ИСПОЛЬЗОВАНИЕМ ГРАФА ЗАДАЧ

В работе предлагается алгоритм автоматизированного построения графа задач для программного кода с использованием алгоритмических алгебр. На основе анализа графа сторояться классы эквивалентности для тестирования программы. Сформулированы принципы анализа такого графа. Использование предложенного метода позволяет реализовать автоматизированное построение классов эквивалентных тестов и сократить время, используемое для построения тестов, за счёт уменьшения количества исполняемого программного кода.

Ключевые слова: *тестирование программ, граф задач, автоматизация тестирования*

Введение

Процесс тестирования является важным этапом жизненного цикла программных систем. Тестированию подлежат как результаты каждо шага разработки программной системы, так и вся программная система в целом. Существуют различные критерии завершения тестирования [1], однако в практической деятельности чаще всего применяется принцип «максимальное количество тестов за определённое время». Однако такой подход требует значительных трудозатрат. Поэтому актуальной является задача автоматизации процессов тестирования.

Одним из направлений такой автоматизации является применение описания программ в виде графов[2]. Наиболее часто при тестировании программ применяются графы исполнениия программы, которые позволяют рассмотреть логику исполнния программы, однако анализ таких графов с целью получения классов эквивалентности для тестирования вызывает сложности, в связи с наличием большого количества маршрутов для анализа. Поэтому для автоматизации задачи тестирования перспективным представлется использование графов с иерархическим остовом. Для построения такого графа можно воспользоваться формальными правилами, основанными на алгоритмических алгебрах [3-5]. Использование такого подхода позволяет осуществлять формальный анализ исходного кода с учётом его семантической структуры. Использование формальной алгоритмической алгебры и её синтаксическое сходство с языками программирования позволяет построить граф, соотвествующий структуре кода и структуре задач, решаемых в коде.

Рассмотрим принципы построения такого графа и алгоритм анализа графа с целью построения классов эквивалентности.

1. Построение графа задач для программного кода

Рассмотрим построение графа структуры задач на примере программной реализации алгоритма сортировки массива вставками. Входными параметрами являются массив и размер передаваемого массива.

```
void insertSort(T a[], long size)
{
T x;
long i, j;
for ( i=0; i < size; i++)
{
x = a[i];
for ( j=i-1; j>=0 && a[j] > x; j--) a[j+1] = a[j];
a[j+1] = x;
}
}
```

Данный код можно представить в виде следующего выражения в алгебре Дейкстры:

$$SORT ::= \{i := 0 * \{[i < size]\ x := a[i] * j = i - 1 * \{[j >= 0 \,\&\, a[j] > x]\ a[j+1] = a[j] * j--\} * * a[j+1] = x * i++\}\} \quad (1)$$

Применяя операцию суперпозиции преобразуем схему таким образом, чтобы она содержала только операции композиции, а в теле циклов находилась только одна операторная переменная. В результате получим следующее выражение:

$$SORT ::= \{i := 0 * \{\overline{[i < size]} A\}\} \quad (2)$$

Преобразуя данное выражение, согласно алгебре сценариев будем иметь:

$$SORT ::= \{([1]\{i := 0[1]\}, E) * (\overline{[i < size]}\{A[i < size]\}, E)\} \quad (3)$$

Соответственно в виде графа данная схема представляется как показано на рис. 1:

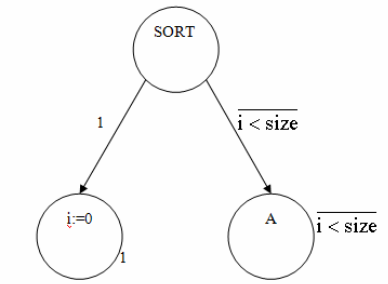

Рис. 1 Граф сценариев для схемы (3)

Далее повторяем эти же действия для схемы ранее обозначенной буквой A. Она имеет вид

$$A ::= \{x := a[i] * j = i - 1 * \{[j >= 0 \,\&\, a[j] > x]B\} * * a[j+1] = x\} \quad (4)$$

Соответствующее представление в алгебре сценариев схемы (4):

$$A ::= \{([1]\{x := a[i][1]\}, E) * ([1]\{j := i - 1[1]\}, E) * *([\overline{j >= 0 \,\&\, a[j] > x}]\{B[\overline{j >= 0 \,\&\, a[j] > x}]\}) * *([1]\{a[j+1] := x[1]\}, E) * ([1]\{i++[1]\}, E)\} \quad (5)$$

Представление в виде графа схемы (5) показано на рис 2.

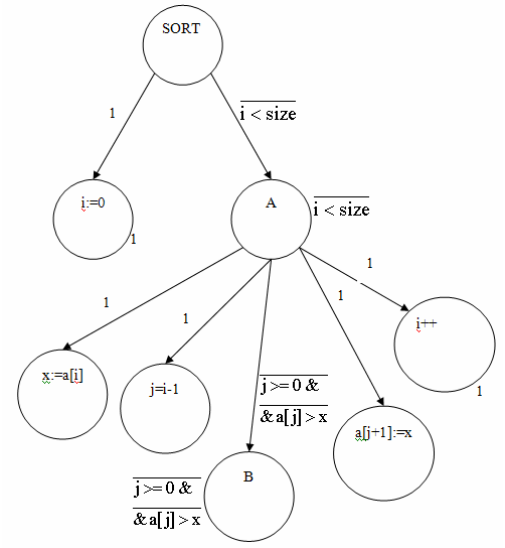

Рис. 2. Граф сценариев для схемы (5)

Схема B в алгебре Дейкстры имеет вид:

$$B ::= \{a[j+1] := a[j] * j--\} \quad (6)$$

Действуя аналогично предыдущим двум случаям, получаем граф задач для листинга программы представленного выше: Для данного алгоритма этот граф представляет собой дерево, однако, в общем случае, например при использовании рекурсивных функций, этого может и не произойти. Поэтому в дальнейшем, для обозначения структуры на рис. 3 будет использоваться термин граф. Рассмотрим далее, каким образом осуществляется анализ указанного графа с целью построения классов эквивалентности.

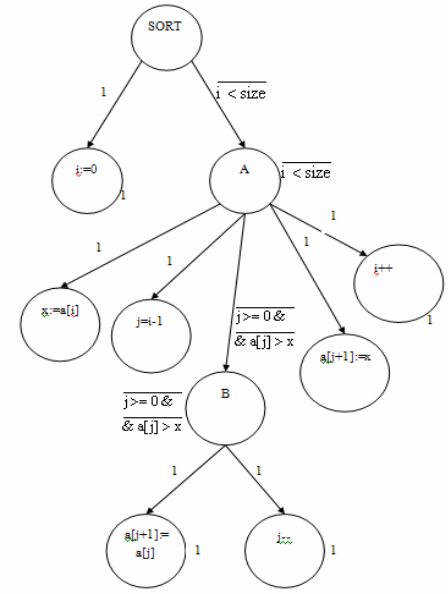

Рис. 3. Граф задач для программы

2. Построение классов эквивалентности

Для построения классов эквивалентности для тестирования программы необходимо рассмотреть структуру дерева с учётом того, что вершины рассматривамого дерева расположены в порядке их исполнения. Как видно из рис. 3, программа может быть успешно выполнена тогда, когда на представленном графе существуют все пути, т.е. все контекстные условия, сопоставленнные рёбрам, истинны.

Таким образом, необходимо установить при каких условиях будут существовать указанные пути. Очевидно, что обход графа в направлениях «сверху-вниз» и «слева-направо» с соблюдением условий, сопоставляемых вершинам и рёбрам, приводит к выполнению указанного алгоритма. Это связано с тем, что данный граф построен с использованием операции суперпозиции и имеет схожую структуру с известными графами операций. Для проведения анализа также необходимо выделить две группы переменных: входные параметры и внутренние переменные алгоритма. В нашем случае входными параметрами являются массив и переменная size.

Учитыая указанную особенность, а также две выделенные группы переменных, проведём анализ построенного графа. Из вершины SORT выходят два ребра. Условия, сопоставленные одному из указанных рёбер всегда истинно, в другом условии используется переменная size и внутренняя переменная i. Учитывая, что одна из переменных является внутренней и между вершинами в графе существует отношение предшествования, можно установить область допустимых значений переменной size. Так как вершина, находящаяся на один уровень выше не изменяет внутреннюю переменную, то для установки истинности условия необходимо отследить операции, находящиеся на одном уровне с вершиной A, но левее её. В данном случае

там находится только одна такая вершина, в которой выполняется оператор i=0. Тогда, очевидно, при старте функции SORT данное условие будет истинно в том случае, если size>0. Поскольку переменная size является входной для программы, то полученное условие задаёт один допустимый (size>0) и один недопустимый (противоположное по значению условие) класс эквивалентности для переменной size.

Рассматривая далее вершины с которыми связана вершина A, можно определить, что при продолжении первого прохода по программе необходимо, чтобы условие $\overline{j >= 0 \& a[j] > x}$ было истинным. Учитывая, что последнее значение j перед проверкой указанного условия на единицу меньше, чем i, то данное условие не выполняется. Поскольку значение переменной j зависит от значения переменной i, то необходимо проверить вершины находящиеся ниже и правее от вершины B на увеличение переменной i в процессе решения задачи. Поскольку условие на данном шаге не выполняется, то необходимо рассмотреть операторы, находящиеся правее на данном уровне. В результате анализа можно заметить, что значение данной переменной в дальнейшем увеличивается на 1, и более никаких преобразований с ней не происходит. Таким образом, можно утверждать, что на следующем шаге условие $j \geq 0$ будет выполнено.

Рассмотрим далее анализ второй половины условия. Учитывая, что ближайшим с левой стороны является операция присваивания $j = i - 1$, то вторую половину условия можно переписать в виде $a_{i-1} > a_i$. Истинность этого условия говорит об упорядоченности массива a. Таким образом, нижележащий уровень выполнится лишь в том случае, если ложно указанное условие. Таким образом, сформулировано условие для ещё двух классов эквивалентности входных данных.

Заключение

В предлагаемой работе расмотрены принципы построения представления программного кода в виде графа задач. Показано, что на основе анализа данного графа возможно построить классы эквивалентности для входных параметров программного кода.

Использование такого подхода позволит в дальнейшем сократить время, используемое для построения тестов, за счёт уменьшения количества исполняемого программного кода, а также автоматизировать построение таких тестов.

Литература

1. Андон Ф.И. Основы инженерии качества программных систем / Ф.И. Андон, Г.И. Коваль, Т.М. Коротун, и др. – К.: Академпериодика, 2007. – 672 с.

2. Липаев В.В. Надежность программных средств. / В.В. Липаев. – М.: СИНТЕГ, 1998. – 232 с.

4. Пригожев А.С. Информационная технология помощи пользователю / А.С. Пригожев // Холодильная техника и технология. – 2007. – № 2. – С. 98-101.

5. Цейтлин Г.Е. Инструментарий конструирования экпертных систем символьной обработки / Г.Е. Цейтлин, Т.К. Терзян, Л.М. Захария // Математические машины и системы. – 1997. – № 1. – С. 14-24.

6. Цейтлин Г.Е. Введение в алгоритмику / Г.Е. Цейтлин — К.: Сфера, 1998. – 310 с.

Поступила в редакцию 3.02.2010

Рецензент: д-р техн. наук, проф. А.В. Дрозд, Одесский национальный политехнический университет, Институт компьютерных систем, Одесса.

ПОБУДОВА КЛАСІВ ЕКВІВАЛЕНТНОСТІ З ВИКОРИСТАННЯМ ГРАФА ЗАВДАНЬ

О.С. Пригожев

У роботі пропонується алгоритм автоматизованої побудови графа завдань для програмного коду з використанням алгоритмічних алгебр. На основі аналізу графа будуються класи еквівалентності для тестування програми. Сформульовані принципи аналізу такого графа. Використання запропонованого методу дозволяє реалізувати автоматизовану побудову класів еквівалентних тестів, а також зменшити час на створення тестів, за рахунок зменшення кількості програмного коду.

Ключові слова: тестування програм, граф завдань, автоматизація тестування.

CONSTRUCTION OF EQUIVALENCE CLASSES WITH USING PROBLEMS GRAPH

A.S. Prigozhev

In the paper we propose the algorithm for automated construction of graph problems for the software package using algorithmic algebra. The equivalence classes which are used to test the program are based on the graph analysis. The analysis principles of this graph was formulated. The use of the proposed method allows to design the automated construction of equivalence classes of tests. Also, it decreases the test creation time due to reducing the number of program code.

Keywords: testing of programs, problems graph, automation of testing.

Пригожев Александр Сергеевич – канд. техн. наук, ст. преп. кафедры сист. прогр. обеспечения Одесского национального политехнического университета, Одесса, Украина, e-mail: prigozhev@rambler.ru.