КОНФИГУРИРУЕМАЯ ГЕНЕРАЦИЯ ТЕСТОВ ДЛЯ ОПТИМИЗИРУЮЩИХ И РАСПАРАЛЛЕЛИВАЮЩИХ ПРЕОБРАЗОВАНИЙ

Е.В. Алымова

Введение

В статье описывается методика конфигурирования под конкретное преобразование тестов, получаемых автоматически с помощью генератора [1]. Предлагается критерий полноты тестовых наборов в терминах комбинаций операторов целевого языка программирования. Приводится пример конфигурации для преобразования «Разрезание цикла» [4].

Различные подходы к автоматической генерации тестов предлагаются в работах [2, 5, 6, 7]. Методы генерации синтаксически корректных предложений целевого языка по формальным спецификациям описаны в работах [5], [6]. Методы генерации тестов на основе моделей предлагается в работе [7]. Критерий тестового покрытия, в соответствии с которым формируется набор тестов, формулируется в терминах модельного языка. Подход к автоматическому тестированию модулей проверки статической семантики в компиляторах описывается в работе [2], где эти модули рассматриваются как булевы функции.

В данной статье предлагается подход к генерации тестов для оптимизирующих преобразований, основанный на формализации требований применимости преобразования. Методика разработана для тестирования преобразований распараллеливающей системы [3].

Конфигурирование теста под оптимизирующее преобразование

Автоматический генератор тестов на вход получает формальное описание грамматики целевого языка программирования и конфигурационный файл для конкретного преобразования. На выходе генерируется синтаксически и семантически корректная программа, удовлетворяющая условиям выполнения преобразования.

Конфигурационный файл содержит формализованное описание условий применимости преобразования. Как правило, условия накладываются на состав операторов и информационные зависимости преобразуемого фрагмента кода. Файл конфигурации записывается в формате XML и строится по следующему шаблону:

```
<program class="%название преобразования%">
      <params {список параметров программы} />
      <body>
            <stat%имяОператора% {список параметров} /> |
<block m = %целое число% {список параметров}>
<stat%Оператор% {список параметров} />
</block> |
<stat%имяОператора% m = %целое число%   {список параметров}>
<stat%имяОператора% {список параметров} /> |
<block {список параметров}>
<stat%имяОператора% {список параметров} />
</block>
</stat%имяОператора% >
      </body>
</program>
```

Конфигурация задает состав операторов тела программы и позволяет управлять информационными зависимостями блоков операторов. Символ «|» означает, что конструкции, которые он разделяет, могут комбинироваться в зависимости от требований выполнимости преобразования.

Конструкцией «<stat%имяОператора% />» определяется конкретный оператор языка программирования. Если этот оператор сложный, т.е. может быть определен через другие операторы (например, условный оператор), то в списке параметров объявляются имена операторов, через которые этот оператор может быть определен. Список параметров имеет вид: «stat%имяОператора% = %значение%» и определяет операторы, которые генерируются в конкретном фрагменте программы. Значение задается целым числом. Если оно равно единице, то в программе появляется один оператор, иначе генерируется последовательность, в которой этот оператор повторяется столько раз, сколько задано параметром. При этом заголовок оператора копируется, а вся последовательность при генерации логически считается одним оператором. Для операторов присваивания и безусловного перехода список параметров пуст.

Конструкция «<block />» определяет блок операторов, который можно вынести в подпрограмму с сохранением синтаксической и семантической корректности. Данная конструкция позволяет гарантировать

генерацию фрагментов программы, имеющих один вход и один выход. Блок операторов удовлетворяет следующим условиям:

1. ни один оператор из блока не находится в зоне действия условного оператора, который не содержится в этом блоке;
2. ни один оператор из блока не находится в теле цикла, заголовок которого не принадлежит этому блоку;
3. все операторы безусловного перехода ссылаются на метки только тех операторов, которые принадлежат блоку;
4. ни один оператор из блока не имеет метку, на которую ссылаются операторы, не принадлежащие этому блоку;
5. ни одна переменная, участвующая в одном блоке, не появляется в других блоках.

Список параметров для блока аналогичен списку параметров для оператора, имеющего тело.

Конструкция «m = %целое число%» для блоков и операторов, имеющих тело, задает количество операторов этого блока или тела. Список параметров и значение m определяет тело оператора или блока как последовательность длины m операторов из списка параметров.

Критерий полноты тестового набора

Критерий покрытия задается в терминах комбинаций операторов целевого языка программирования. Обозначим множество операторов языка программирования Q = {OP1, OP2, …, OPn}, где OPi – оператор, n – количество операторов. Зафиксировав целое число m, определим множество S как множество всех возможных комбинаций, которые можно составить из m-ок элементов, принадлежащих множеству Q (декартовых произведений элементов множества Q степени m). Мощность множества S равна n в степени m.

Каждой комбинации операторов из множества S ставится в соответствие тестовая программа, содержащая эту комбинацию. Набор тестовых программ ранга k (целое число, большее 1 и меньшее m) полон, если для каждого элемента из множества S в тестовом наборе найдется по крайней мере одна программа, соответствующая этому элементу.

Конфигурационный файл указывает место в программе, где должна быть сгенерирована последовательность операторов из множества S. Конструкция «m = %целое число%» и список параметров для блоков и сложный операторов, задает множество Q операторов и фиксирует целое число m. Эти конструкции указывают, что при генерации тестов необходимо осуществить перебор всех возможных m-ок операторов из списка параметров. Обозначим множество этих m-ок как M. Присутствие в файле конфигурации t блоков с заданным m и списком операторов означает, что заданы множества M1, M2,…, Mt. Тестовый набор для такой конфигурации считается полным, если для каждой последовательности из декартова произведения множеств M1, M2,…, Mt найдется тестовая программа, соответствующая этой последовательности.

Пример конфигурации для преобразования

Преобразование «Разбиение цикла» [4, с. 46-50] заменяет цикл

```
DO 333 I = 1, N
S1
……
Sk
S(k+1)
……
Sm
333   CONTINUE
на фрагмент программы, состоящий из последовательности двух циклов:
DO 111 I = 1, N
S1
……
Sk
111   CONTINUE
DO 222 I = 1, N
S(k+1)
……
Sm
222   CONTINUE
```

Для корректного применения данного преобразования должны быть выполнены следующие условия:

1. каждый из фрагментов программы S1, …, Sk и S(k+1), …, Sm имеют один вход и один выход;
2. не существует такой дуги графа информационных связей (v1, v2), что v1 принадлежит Si, i больше либо равно k+1 и меньше либо равно m, v2 принадлежит Sj, j больше либо равно 1 и меньше либо равно k.

Конфигурационный файл имеет вид:

```
<?xml version="1.0" encoding="UTF-8"?>
<program class="loop_distribution">
      <params … />
      <body>
            <statFor>
                  <block m="3" statAssign="1" statIf="1">
                        <statIf statAssign="1" />
                  </block>
                  <block m="4" statAssign="1" statIf="1">
                        <statIf statAssign="1" statGoTo="1" />
                  </block>
            </statFor>
      </body>
</program>
```

Данная конфигурация определяет программы, состоящие из одного оператора цикла. Тело цикла разбито на два блока, между которыми нет зависимостей, направленных снизу вверх. Телом первого блока является последовательность из множества троек, которые можно составить из операторов присваивания и условного перехода. Телом второго блока является последовательность, принадлежащая множеству четверок, которые можно составить из оператора присваивания и условного оператора. Количество программ в полном тестовом наборе для данной конфигурации равно 128 (произведению 2 в четвертой степени и 2 в третьей степени).

ЛИТЕРАТУРА:

1. Алымова Е.В. Автоматическая генерация тестов на основе конфигурационных файлов для оптимизирующих преобразований компилятора // Известия вузов. Северо-Кавказский регион. Естеств. науки. № 3, 2010 - с. 5-8.
2. Архипова М.В. Генерация тестов для семантических анализаторов // Вычислительные методы и программирование. 2006, том.7, раздел 2, с. 55-70.
3. Открытая распараллеливающая система – http://ops.rsu.ru/
4. Штейнберг Б.Я. Математические методы распараллеливания рекуррентных циклов для суперкомпьютеров с параллельной памятью. Ростов н/Д: Изд-во Рост. ун-та, 2004 – 192 с.
5. A. Kalinov, A. Kossatchev, M. Posypkin, V. Shishkov. Using ASM Specification for automatic test suite generation for mpC parallel programming language compiler. // Proc. Fourth International Workshop on Action Semantic, AS'2002, BRICS note series NS-02-8 (2002) 99-109.
6. A.S. Kossatchev, P. Kutter, M.A Posypkin. Automated Generation of Strictly Conforming Tests Based on Formal Specification of Dynamic Semantics of the Programming Language // Programming and Computing Software. July 2004. Volume 30. Issue 4. 218-229.
7. A.S. Kossatchev, A.K. Petrenko, S.V. Zelenov, S.A. Zelenova. Application of Model-Based Approach for Automated Testing of Optimizing Compilers. In Proceedings of the International Workshop on Program Understanding (Novosibirsk - Altai Mauntains, Russia), July 14-16, 2003, 81-88