Реализация склеивания переменных в предикатной программе[1]

Каблуков Иван Владимирович
Институт систем информатики имени А. П. Ершова СО РАН (Новосибирск), Россия
email: vanyok05@mail.ru

Введение

Язык предикатного программирования P [2] - это язык функционального программирования, в котором сочетаются функциональный и операторный (предикатный) стили записи алгоритмов. Он обладает существенно большей выразительностью по сравнению с чисто функциональными языками. По синтаксису, набору операторов и операция язык приближен к стилю языков семейства C. Эффективность предикатных программ достигается применением системы оптимизирующих трансформаций, переводящих программы на язык императивного расширения языка P.

Базовыми трансформациями в языке P являются: склеивание переменных, реализующее замену нескольких переменных одной; замена хвостовой рекурсии циклом; подстановка определения предиката на место его вызова; кодирование структурных объектов низкоуровневыми структурами с использованием массивов и указателей.

Склеивание переменных [1] – это замена (сохраняющая эквивалентность) в тексте программы всех вхождений одной переменной на другую. Значительный эффект достигается при склеивании структурных переменных, таких как массивы и списки, поскольку склеивание обычно позволяет избежать копирования структур.

В отличие от задачи экономии памяти [3], склеиванию в программе подлежат результаты с аргументами, аргументы с локалами и локалы с результатами. Набор склеиваний может быть частично задан пользователем. Необходимо проверить его корректность и дополнить. В языке P правилами языка запрещено присваивание вида: x:=op(x, y). Поэтому в языке P существуют только присваивания x:=op(x1, y). При трансляции x1 склеивается с x. Например, при склеивании переменных c и d оператор c:=d+1 будет преобразован в оператор присваивания c:=c+1, а оператор a:=b при склеивании a и b превратится в оператор a:=a, удаляемый из программы. Типы склеиваемых переменных должны совпадать. В дополнение к этому, переменная параметрического типа T(n+1) может быть склеена с переменной типа T(n) в случае возрастания n на 1.

1. Общая схема реализации

Система предикатного программирования включает: front-end (реализующий синтаксический и семантический анализ), потоковый анализатор, генератор условий корректности при дедуктивной верификации, оптимизирующий трансформатор и др. компоненты. Оптимизирующий трансформатор реализует базовые трансформации: склеивание переменных, замену хвостовой рекурсии циклом, подстановку определения предиката на место его вызова, кодирование структурных объектов низкоуровневыми структурами. Потоковый анализатор строит слои программы, а также для каждого оператора: регионы склеивания и *пост-аргументы*. Пост-аргументами оператора являются переменные, используемые при дальнейшем исполнении после завершения исполнения оператора. Склеивание переменных реализуется в оптимизирующем трансформаторе с помощью алгоритмов уточнения регионов.

Регион склеивания для оператора G есть набор аргументов и результатов одного типа <x: y>, где x – список аргументов и y – список результатов оператора. **Пример**.

[1] Работа выполнена при финансовой поддержке РФФИ, грант № 12-01-000686.

Пусть имеется оператор F с аргументами a, b, c, d, e и результатами f, g, h. Пусть переменные a, b, d и g, h имеют натуральный тип, а переменные c, e и f – массивы. Тогда для оператора F регионы склеивания таковы: <a, b, d: g, h> и <c, e: f>. Регион склеивания вида <a**:** b> , где a и b –переменные, является *командой склеивания*, определяющей замену переменной a на b.

Полная программа делится на *слои*: слой первого уровня содержит программы, не вызывающие другие программы, слой второго уровня содержит программы, вызывающие только программы из слоя первого уровня и т.д. Также существуют рекурсивные кольца – набор программ, каждая из которых может быть вызвана по цепочке вызовов программ из этого кольца, начиная с вызова внутри себя. Каждое рекурсивное кольцо содержится в отдельном слое. Рекурсивный вызов – это вызов программы из того же рекурсивного кольца, в котором находится вызывающая программа. Для таких вызовов в случае, когда программ в рекурсивном кольце больше одной, нужно отдельное рассмотрение. Построение регионов упорядочено по уровням слоев, начиная с первого, и реализуется обходами дерева программы сверху вниз и снизу вверх.

2. Построение множеств пост-аргументов

Для произвольного оператора D определим рекурсивную программу DF(D, $ArgP_D$**:** Arg_D), вычисляющая аргументы Arg_D оператора D, а также аргументы и пост-аргументы для всех вложенных операторов; $ArgP_D$ обозначает ранее вычисленные пост-аргументы оператора D.

Для оператора суперпозиции A = B; C программа DF(A, $ArgP_A$**:** Arg_A) представляется следующей последовательностью операторов:

$ArgP_C = ArgP_A$;
DF(C, $ArgP_C$**:** Arg_C)
$ArgP_B = ArgP_A \cup Arg_C$;
DF(B, $ArgP_B$**:** Arg_B)
$Arg_A = Arg_B \cup Arg_C$;

Для условного оператора A = if(E) B else C и параллельного оператора A = B || C программа DF(A, $ArgP_A$**:** Arg_A) представляется следующей последовательностью операторов:

$ArgP_B = ArgP_A$;
$ArgP_C = ArgP_A$;
DF(B, $ArgP_B$**:** Arg_B)
DF(C, $ArgP_C$**:** Arg_C)
$Arg_A = Arg_B \cup Arg_C$;

3. Построение регионов склеивания

Для построения регионов склеивания программа обходится снизу вверх. В дереве предикатной программы нижними операторами являются операторы присваивания и вызовов предикатов, в которых вычисляются значения их результатов. Для этих операторов строятся регионы склеивания определенным ниже образом. После обходятся объемлющие операторы суперпозиции, параллельные и условные операторы, регионы которых строятся на основе уже определенных регионов подоператоров.

Для оператора **присваивания** регион строится с набором его аргументов в левой части и переменной присваивания в правой.

Предполагается, что для предиката, вызываемого в операторе **вызова** предиката,

уже построены итоговые регионы склеивания результатов с аргументами. Подставляя соответствующие аргументы и результаты вызова в эти регионы, получаются регионы данного оператора вызова. Такой способ построения регионов для вызова реализуется при условии открытой подстановки тела предиката или при подстановке параметров по ссылке. В ином случае для оператора вызова предиката строится регион с набором аргументов вызова в левой части и набором результатов в правой.

Пример

```
Foo (int a, b: int a`, b`)
{ ... }
```

При подстановке тела предиката на место вызова Foo (p1+p2, p3+p4**:** r1, r2) получается оператор суперпозиции: a = p1+p2; b = p3+p4; {…} r1 = a; r2 = b. Переменные a и b становятся локалами предиката. Строятся следующие регионы: <p1, p2**:** a>, <p3, p4**:** b>, <a**:** r1> и <b**:** r2>.

В подоператорах **параллельного оператора** все результаты различны, а аргументы делятся на уникальные – участвующие только в одном подоператоре и общие – участвующие в более чем одном подоператоре. Уникальные аргументы возможно склеить с результатами, но общие аргументы склеивать нельзя, т.к. они используются в других подоператорах. Однако с использованием замены параллельного выполнения на последовательное иногда можно склеить общие аргументы.

Алгоритм. Если в некотором регионе подоператора в левой части есть и уникальные и общие аргументы, то склеивание производится с уникальным аргументом. Если в регионе подоператора есть только уникальные аргументы, а общих нет, то склеивание можно провести, сохраняя параллельное исполнение подоператора с остальными. Для каждого общего аргумента из регионов считается, во скольких подоператорах он участвует. Выбирается наиболее редкий аргумент, выбираются все подоператоры, в которых он участвует, кроме одного и выполняются параллельно до исходного параллельного оператора. В оставшемся подоператоре этот аргумент станет уникальным и его можно будет склеить. Таким образом, обрабатываются все регионы с общими аргументами.

В **условном операторе** если в некоторых регионах с разными результатами есть одинаковый аргумент, то этот аргумент удаляется из данных регионов. Если в некотором регионе не осталось аргументов, то этот регион удаляется. Далее, оставшиеся регионы с одинаковым результатами объединяются, образуя один регион с данным результатом в правой части и объединением аргументов в левой.

Для **оператора суперпозиции** производится склеивание через локалы:

Обходятся регионы подоператоров в порядке их исполнения. Если в регионе в правой части стоит локал, то все аргументы из левой части региона подставляются в последующие регионы вместо локала. Получившиеся регионы приписываются оператору суперпозиции.

4. Построение команд склеивания

Команды склеивания строятся в оптимизирующем трансформаторе уточнением регионов склеивания, построенных потоковым анализатором. Сначала регионы уточняются набором *априорных склеиваний*, заданных в исходной программе. Пользователь может задать в заголовке предиката склеивание результата с аргументом путем именования результата именем аргумента со штрихом на конце. Команда склеивания может быть также задана прагмой. Проверяется, что априорные склеивания

могут быть получены уточнением регионов. В противном случае, выдаются сообщения о некорректном задании априорных склеиваний.

Все получившиеся регионы с одним результатом и аргументом считаются командами склеивания. Для регионов с несколькими аргументами и одним или несколькими результатами произвольным образом выбираются команды склеивания с одним результатом и одним аргументом.

После построения полного набора команд склеивания программа обходится сверху вниз, склеивая переменные исходя из регионов склеивания. Параллельные операторы при необходимости переставляются в последовательные, в операторах суперпозиции производятся склеивания с локалами.

5. Пример

Приведем пример предикатной программы, которая находит целочисленный квадратный корень:

```
sq1(nat x, k, n : nat m)
{       nat p = n + 2* k + 1;
        if (x < p) m = k else  sq1(x, k + 1, p: m)
}
```

Для операторов присваивания строятся регионы <n, k: p> и <k: m>. Для рекурсивного вызова регионы не строятся. Далее на условном операторе объединяются регионы с его ветвей и получается <k: m>. Для оператора суперпозиции, содержащего оператор присваивания и условный оператор из регионов его подоператоров <n, k: p> и <k: m> получаются регионы <n: p> и <k: m> для наибольшего количества склеиваний. Эти регионы приписываются предикату sq1. Они являются командами склеивания, программа обходится сверху вниз заменяя переменные из правых частей на переменный из левых.

Итоговая программа, в которой поизведены склеивания <n: p> и <k: m>:

```
sq1(nat x, k, n)
{       n = n + 2* k + 1;
        if (x < n) k = k else  sq1(x, k + 1, n: k)
}
```

Список литературы

1. Петров Э.Ю. Склеивание переменных в предикатной программе // Методы предикатного программирования. Новосибирск, 2003. С. 48-61.
2. Шелехов В.И. Введение в предикатное программирование. - Новосибирск, 2002. - 82с. - (Препр. / ИСИ СО РАН; N 100).
3. Ершов А.П. Введение в теоретическое программирование - М.: Наука, 1977. - 288с.