П. А. Дортман

ПОДХОДЫ К ОПТИМИЗАЦИИ ПРОГРАММ В СИСТЕМЕ SFP*

1. ВВЕДЕНИЕ

Настоящая работа выполняется в рамках проекта по созданию системы функционального программирования SFP [1]. Эта система позволяет прикладному программисту разрабатывать вычислительные алгоритмы на языке Sisal 3.0 на своей рабочей станции под управлением Windows и генерировать программы для последующего исполнения на суперЭВМ. Система разрабатывается с использованием Microsoft Visual Studio.

Язык Sisal 3.0 — функциональный язык однократного присваивания — был разработан как альтернатива ФОРТРАНу для реализации сложных вычислительных алгоритмов главным образом для научных вычислений [2, 3]. Традиционно для представления Sisal-программ используют представление программы в виде иерархического ациклического потокового графа (например, IF1-представления [4]). В системе SFP внутреннее представление Sisal-программ (будем называть его IR1-представлением) также реализовано как набор классов языка Си++, реализующих средства для работы с IF1-графами.

2. IF1-ПРЕДСТАВЛЕНИЕ

Каждый IF1-граф служит для представления некоторой логически целостной части вычислений, например, для представления в программе функций, отдельных итераций цикла и т.п. А сама программа, таким образом, представляется в виде набора графов функций.

Каждый IF1-граф состоит из *вершин* (операций) и *дуг*, которые представляют собой передачу значения между операциями во время исполнения. Количество различных дуг ограничено, и для каждого типа дуг известно, сколько и какого типа значения вершина принимает. Для удобства упорядоченные места вершин, к которым прикрепляются дуги, будем называть

* Работа выполнена при финансовой поддержке Научной программы «Университеты России» (грант № УР.04.01.027) и Министерства образования РФ (грант № Е02-1.0-42).

входными портами. Места вершин, из которых выходят исходящие дуги с вычисленными значениями, будем называть *выходными портами*. Для вершин каждого типа определено, какие значения на какие входные порты должны приходить, как значения участвуют в вычислении и какие значения формируются на соответствующих выходных портах.

Существуют также специальные константные простые вершины без входных портов, они служат для представления константных значений в вычислениях.

Наряду с простыми вершинами, которые олицетворяют простые операции, такие как арифметические действия, доступ к элементу массива или вызов функции, существуют и составные вершины, состоящие из нескольких подграфов, которые служат для представления циклов и условных конструкций. Так, например, вершина для цикла с предусловием состоит из трех подграфов: граф для представления вычисления условия цикла, граф для вычисления каждой итерации цикла и граф для формирования возвращаемого циклом результата, формируемого по вычисляемым в каждой итерации значениям. Наличие составных вершин в IF1-графах делает их иерархичными.

Ниже приведены примеры графов с простыми и составными вершинами.

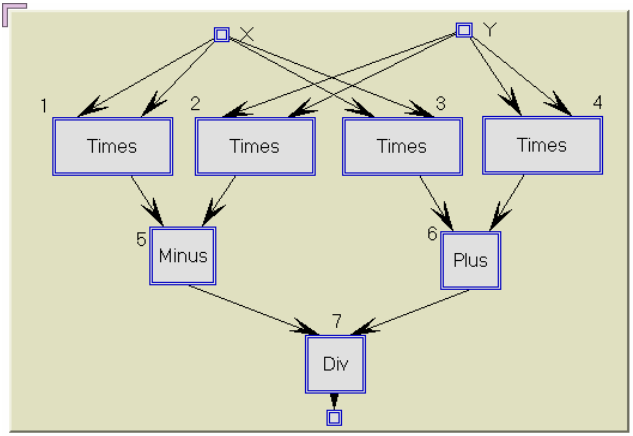

Рис. 1. Граф для вычисления выражения $(x^2 - y^2)/(x^2 + y^2)$

Пример Sisal-функции и его внутреннего представления.

```
function sample(X1, X2, h: real returns real)
 for initial
     X := X1;
     m := 0.0
 while X <= X2 repeat
      X := old X + h;
      m := old m + abs(F(old X) — G(old X))
 returns value of m
end for
end function
```

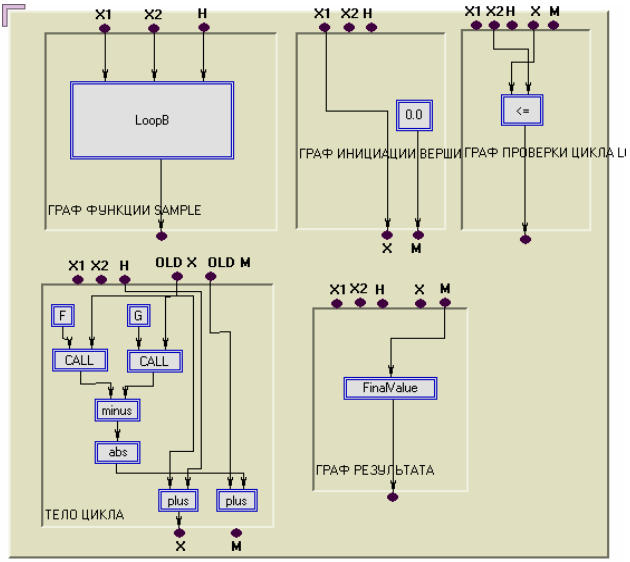

Рис. 2. IF1-граф для примера выше

Большинство традиционных оптимизирующих преобразований над IF1-графами легко программируются просто как преобразования над графами. В настоящей работе мы показываем это на примере ряда преобразований.

Прежде всего отметим, что для ациклического графа, каким является IF1-граф, существует такая нумерация его вершин, которая сохраняет частичный порядок зависимости по данным.

3. ПОДСТАНОВКА ФУНКЦИИ ВМЕСТО ВЫЗОВА

Вызовы функций и процедур зачастую затрудняют автоматический анализ программ в императивных языках. В языке Sisal же вызовы функций не имеют побочных эффектов, их результаты зависят только от своих параметров. В таком случае вызов любой функции может быть заменен кодом с телом этой функции. В IF1-графах подобная подстановка вызова функции на граф тела функции легко алгоритмизируется. И после такой подстановки расширенный граф может быть оптимизирован с помощью других оптимизирующих преобразований, таких как исключение общих подвыражений и вынос инвариантов циклов.

Обычно, когда говорят о подстановке функции, имеют в виду нерекурсивные функции. Однако это вовсе не жесткое правило, раскрытие одного или более рекурсивного вызова может дать дополнительные возможности для проведения других оптимизаций.

4. ИСКЛЮЧЕНИЕ ОБЩИХ ПОДВЫРАЖЕНИЙ

Главной целью преобразования, известного как исключение общих подвыражений, является удаление избыточных вычислений. Отдельные вычисления являются избыточными, если они вычисляются неоднократно при расчете. В IF1-графе избыточные вычисления приобретают форму эквивалентных вершин.

Дадим определение *эквивалентных* вершин. Две простые вершины называются *эквивалентными*, если они представляют одну и ту же операцию (являются вершинами одного типа) и имеют эквивалентные входы. Две составные вершины называются *эквивалентными,* если они являются вершинами одного типа, их входы эквивалентны и их подграфы соответственно изоморфны. Два графа называются изоморфными, если можно установить биективное отображение между их вершинами так, что соответствующие вершины эквивалентны. Будем говорить, что две вершины имеют *эквивалентные входы*, если они имеют одинаковое число входов и для каждой пары входных портов верно одно из следующих утверждений: входные дуги идут от константных вершин с одинаковым значением и типом, входные

дуги идут от одинаковых входных портов границы графа либо входные дуги идут от соответствующих портов эквивалентных вершин.

Понятно, что эквивалентные вершины можно отождествить, так как ведут они себя одинаково, и, таким образом, сократить число вершин-операций в графе. Например, на рис.1 вершины с номерами 1 и 3, а также 2 и 4 — эквивалентны. Исключение общих подвыражений и есть процесс склеивания эквивалентных вершин. Приведем формальный алгоритм исключения общих подвыражений для IF1-графов.

Алгоритм исключения общих подвыражений из графа.
Вход: пронумерованный указанным образом граф G.
Выход: Эквивалентный граф, в котором все эквивалентные вершины склеены.

```
проц ИСКЛЮЧЕНИЕ
    для каждого типа T вершин графа цикл
        Создать пустое множество вершин S(T)
    все
    для каждой вершины N из G в порядке нумерации цикл
        если N — составная то
            для каждого подграфа H из N цикл
                ИСКЛЮЧЕНИЕ(H)
            все
        все
        { поиск эквивалентных вершин }
        пусть S=S(T(N)) — подмножество вершин с таким же типом
                операции T(N), что и у вершины N
        F := 1
        для каждого M из S и  пока F=1 цикл
            если M эквивалентно N то
                Склеить N и M
                F:=0
            все
        все
        если F=1 то S := S+{ N } все
    все
все.
```

Как мы видим, настоящий алгоритм оптимизирующего преобразования графа получился довольно простым и эффективным. Проверка составных вершин теоретически сложна, но на практике не требуется много действий,

чтобы выяснить, что вершины не эквивалентны, так как составные вершины очень отличаются друг от друга.

Некоторые вершины служат для представления коммутативных операций (например сложения или умножения). Однако наш алгоритм не воспринимает, например, 5*G и G*5 как одинаковые выражения. Также настоящий алгоритм не пытается изменить порядок действий для нахождения одинаковых выражений: 5*(G*H) всегда отлично от (5*G)*H.

5. ВЫНОС ИНВАРИАНТОВ ЦИКЛА

Определение выражений, являющихся инвариантом для цикла, и вынос их за цикл, исключают многократное выполнение одного и того же вычисления и являются еще одним традиционным оптимизирующим преобразованием, присущим почти всем оптимизирующим компиляторам. Инвариантом цикла называется выражение, которое вычисляется в одно и то же значение на каждой итерации цикла. Такие выражения могут создаваться программистом с целью повышения читаемости кода или появляться вследствие выполнения каких-либо еще преобразований. Мы сейчас определим, что значит, что IF1-вершина внутри подграфа одной из составных вершин цикла является инвариантом.

Пусть X — вершина внутри подграфа G, содержащегося в графе L цикла. Будем говорить, что X является инвариантом цикла в L, если каждый ее входящий порт принимает значение:

1) либо от константной вершины,

2) либо с порта входных значений L,

3) либо от вершины, являющейся инвариантом цикла.

Предположим, что вершина X является инвариантом цикла или не зависит от каких-либо других вершин, т.е. все ее входящие порты удовлетворяют условиям (1) и (2). В этом случае мы можем вынести X из подграфа G цикла L, последовательно выполняя следующие действия.

1. Вынести X вместе со своими входными дугами из графа G.

2. Поместить вершину X в граф, содержащий L непосредственно перед L в порядке зависимости по данным. Присоединим каждую входную дугу вершины X к соответствующему входному порту L и соединяем выходной дугой исходящие порты с создаваемыми входными портами на L.

3. Соединить вершины, получавшие значения из X, с созданными портами на G.

4. Удалить все входы L, которые использовались исключительно вершиной X.

Понятно, что если мы будем обходить вершины в порядке, сохраняющем зависимость по данным, то проверка условия (3) будет излишней, достаточно проверять только первые два условия. Так как в алгоритме исключения общих подвыражений также использовался обход в порядке, сохраняющем зависимость по данным, то совместное применение этих двух оптимизаций можно естественным образом алгоритмизировать. При этом наиболее удачной нам представляется стратегия, при которой вынос инвариантов цикла выполняется для всех циклов графа, а затем происходит поиск общих подвыражений. Такая стратегия позволяет не упустить шанс склеить только что вынесенные из циклов вершины с другими вершинами в графе.

6. ЗАКЛЮЧЕНИЕ

Семантика IF1-графов позволяет проводить оптимизирующие преобразования программ как эффективные преобразования IF1- графов. И мы продемонстрировали это на примере трех традиционных преобразований.

СПИСОК ЛИТЕРАТУРЫ

1. Глуханков М.П., Дортман П.А., Павлов А.А., Стасенко А.П. Транслирующие компоненты системы функционального программирования SFP // Современные проблемы конструирования программ. — Новосибирск, 2002. — С. 69–87
2. Cann D. Retire FORTRAN? A debate rekindled // CACM. —1992. — Vol. 35, N 8. — P. 81–89.
3. Касьянов В.Н., Бирюкова Ю.В., Евстигнеев В.А. Функциональный язык Sisal 3.0 // Поддержка супервычислений и Интернет-ориентированные технологии. — Новосибирск, 2001. — С. 54–67.
4. Густокашина Ю.В., Евстигнеев В.А. IF1 — промежуточное представление Sisal-программ // Проблемы конструирования эффективных и надежных программ. — Новосибирск, 1995. — С. 70–78.