Федеральное государственное бюджетное образовательное учреждение
высшего образования «Московский государственный технический университет
имени Н.Э. Баумана (национальный исследовательский университет)»

Автоматическая разметка оптимизируемых функций в компиляторе Рефала-5λ

Работу выполнила: Калинина Е.А.
Группа: ИУ9-81

Научный руководитель: Коновалов А.В.

2020

Постановка задачи

Цель работы: добавление в компилятор Рефала-5λ корректной и безопасной автоматической разметки специализируемых и прогоняемых функций.

Для этого необходимо:

- Выбрать критерии, по которым будет работать разметка и которые обеспечат её корректность и безопасность;
- Реализовать алгоритм, обеспечивающий автоматическую разметку функций, удовлетворяющую выбранным критериям.

Рефал-5λ

Рефал (РЕкурсивных Функций АЛгоритмический язык) – один из старейших функциональных языков программирования, ориентированный на символьные вычисления.

У данного языка есть много диалектов, язык Рефал-5λ – один из них.

Основным отличием диалекта Рефал-5λ является поддержка анонимных функций и функций высшего порядка.

Специфика языка Рефал-5λ

Сопоставление с образцом – метод обработки и анализа структурированных данных, основанный на выполнении заданных инструкций в зависимости от совпадения анализируемого значения с одним из нескольких заданных образцов, описывающих аргумент.

```
BinAdd {
  '0' '0' = '0';
  '0' '1' = '1';
  '1' '0' = <BinAdd '0' '1'>;
  '1' '1' = '10';
}
```

Переменные

В Рефале есть 3 типа переменных s-, t- и e-переменные, в зависимости от типа множества значений, которые эти переменные могут принимать.

```
IsEqual {
  (e.X) (e.X) = 'True';
  (e.X) (e.Y) = 'False';
}
```

Тип переменной

Имя переменной (индекс)

Специализация

f – функция, $f(\text{args})$ – вызов функции.

Специализация – такое порождение новой функции f' и замена вызова $f(\text{args})$ на вызов $f'(\text{args}')$, что в f' учтена часть статически известной информации о её вызове, например, значения некоторых аргументов.

То есть, если функция f принимает два аргумента x, y и можно предположить, что первый аргумент x принимает значение A, то функция может быть специализирована по этому аргументу:

$$f(x,y) \to f_A(y) \to f(A,y)$$

$$Spec(f,A) = f_A = f'$$

$$f(x,y) = Spec(f,A)(y) = f'(y)$$

Прогонка

Прогонка – это один из способов оптимизации программы, при котором берётся вызов функции в правой части предложения и заменяется на стадии компиляции на результат своей работы. При этом возможно расщепление предложения на несколько, имеющих более точные образцы.

```
Inc {
  s.Num = <Add s.Num 1>;
}
```

```
<Inc s.Depth> → <Add s.Depth 1>
```

Критерии автоматической разметки

- Безопасность – помеченная функция не должна приводить к зацикливанию оптимизатора;
- Корректность:
  - метки должны ссылаться на существующие функции;
  - метки не должны конфликтовать с другими метками;
  - шаблон специализации должен согласовываться с определением функции.

Автоматическая разметка специализируемых функций

Алгоритм поиска шаблона специализации:

1. Найти обобщение всех образцов

Обобщение образца P – это образец P_G такой, что существует подстановка, переводящая P_G в P.

Обобщение нескольких образцов P_1, …, P_n – такой образец P_G , что существуют подстановки, переводящие P_G в каждый из исходных образцов.

Автоматическая разметка специализируемых функций

Алгоритм поиска шаблона специализации:

2. Промаркировать в найденном обобщении статические и динамические параметры

Специализация проводится только по статическим параметрам, при этом в теле функции вместо всех вхождений статического параметра подставляется его фактическое значение.

Статические параметры – это такие параметры, которые проецируются в каждом предложении на одну переменную того же типа, динамические – все остальные.

Пример вычисления шаблона специализации

```
Replace {
  (e.From) (e.To) e.Head e.From e.Tail = …;
  (e.F)    (e.T)  e.Text               = …;
}
```

```
1. (e.0) (e.1) e.2
2. (e.STAT1) (e.STAT2) e.dyn3
```

Шаблон специализации:

```
$SPEC Replace (e.STAT1) (e.STAT2) e.dyn3
```

Автоматическая разметка прогоняемых функций

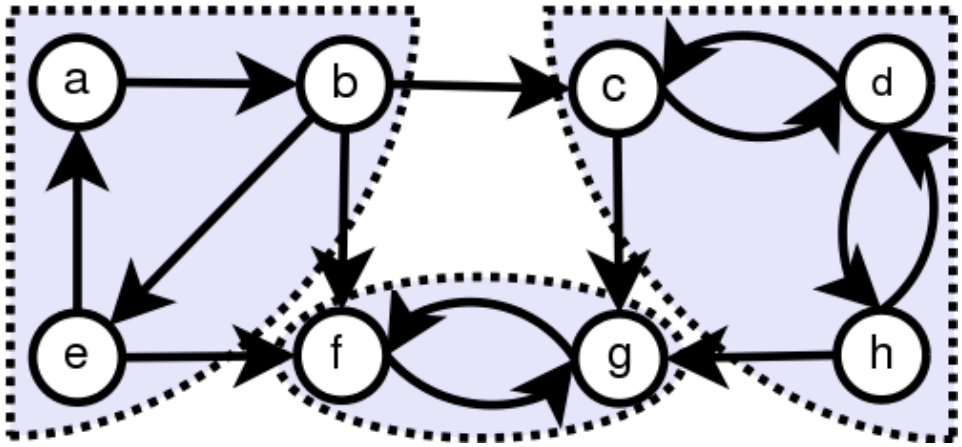
- Нерекурсивной функции можно назначать метку DRIVE.

- Саморекурсивной функции нельзя назначать метку DRIVE – это приведёт к зацикливанию.

- Взаимно-рекурсивной функции можно назначать метку DRIVE только если она вызывается в программе один раз, тогда прогонка просто сожмёт эту компоненту сильной связности на одно звено.

Автоматическая разметка прогоняемых функций

- Представим функции и их вызовы в виде графа;
- Рекурсии – компоненты сильной связности;
- Построим граф конденсации.



Автоматическая разметка прогоняемых функций

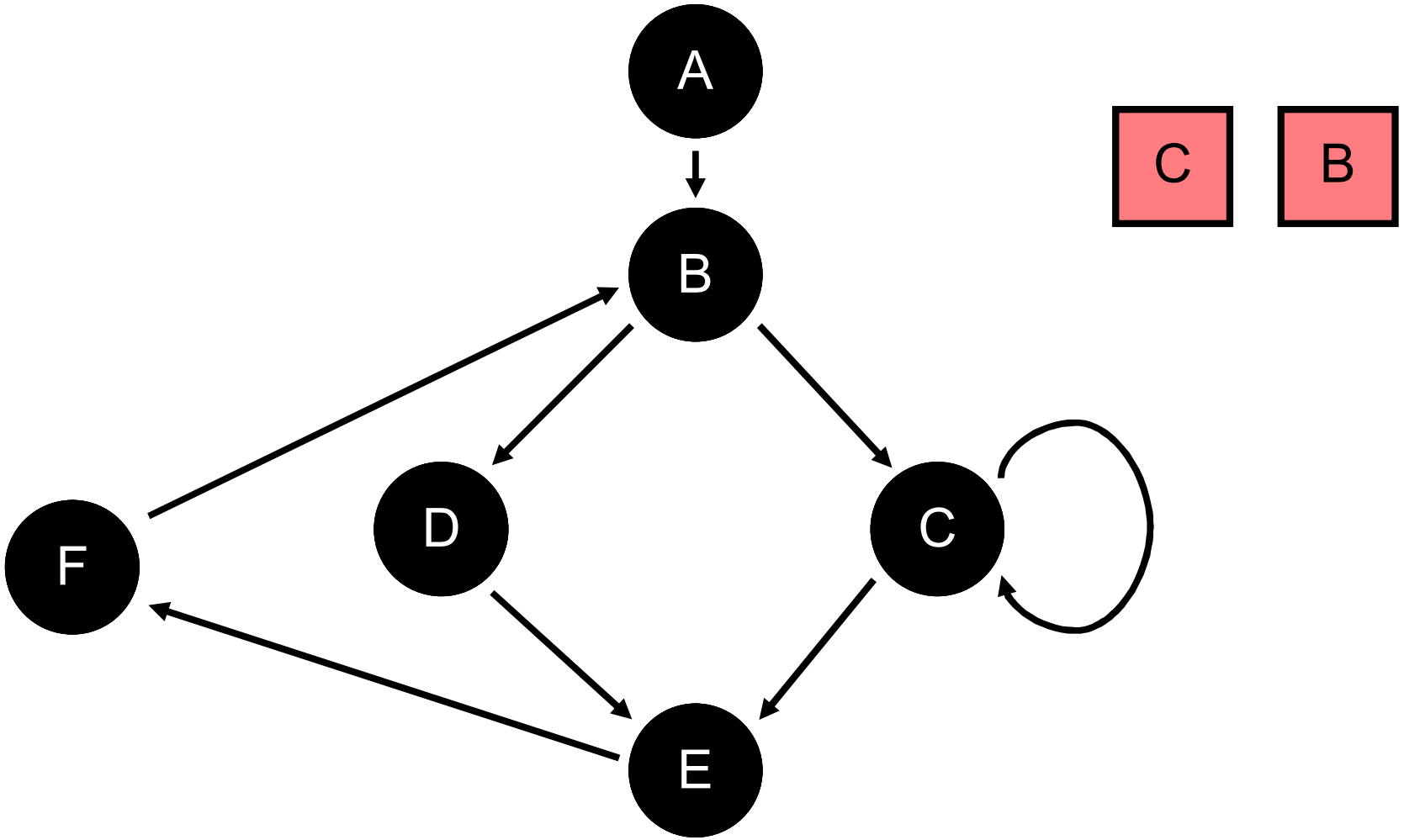
Пусть у нас есть корень R и текущая вершина N. Предки вершины – это путь, по которому мы спустились от R к N, включая R и N:

$$R \equiv P_1 \to P_2 \to \ldots \to P_{k-1} \to P_k \equiv N$$

Если среди потомков N есть хотя бы одна P_i, то от N к P_i ведёт обратная ветка, а вершину P_i нельзя прогонять.

Таким образом, всем функциям, к которым не ведёт обратное ребро при обходе в глубину, можно ставить метку DRIVE.

Обход графа



Тестирование

№ теста	Кол-во шагов	Время выполнения программы, с		
		медиана	I квартиль	IV квартиль
1	20.600.002	25.164	25.087	25.278
2	12.000.002	10.933	10.651	11.159
3	6.500.002	6.892	6.793	7.127

Заключение

В результате работы был реализован алгоритм автоматической разметки специализируемых и прогоняемых функций в компиляторе языка Рефал-5λ.

Наиболее перспективными направлениями дальнейшего исследования представляются:

- Выбор альтернативных критериев проведения автоматической разметки и исследование повышения или понижения времени выполнения программ с использованием альтернативных критериев;
- Добавление автоматической разметки оптимизируемых функций для других видов оптимизации.