# Оптимизация специализации функций в Рефале-5$\lambda$

**Д. П. Сухомлинова**
МГТУ имени Н. Э. Баумана

Второе совместное рабочее совещание
ИПС имени А.К. Айламазяна РАН и МГТУ имени Н.Э. Баумана
по функциональному языку программирования Рефал

**11 июня 2019 года**

# Специализация в общем и частном случае

- «Специализация программ — это порождение по универсальной программе с множеством параметров специализированной программы, когда значения части параметров известны и фиксированы» (источник: Андрей В. Климов «Введение в метавычисления и суперкомпиляцию»).

- Мы рассматриваем специализацию отдельно взятого определения функции для случаев её вызова в отдельных точках программы. Предполагается, что функция на Рефале имеет несколько аргументов, часть из которых известны и фиксированы

# Постановка задачи

- Целью данной работы является расширение возможностей оптимизаций компилятора Рефала-5λ добавлением специализаций функций

- Для этого необходимо:
  - определить правила объявления специализации функции по паттерну и ключи компиляции для включения данной оптимизации;

  - реализовать алгоритм специализации;

- В дальнейшем необходимо также оценить выгоду оптимизации специализации наиболее употребительных библиотечных функций языка на примере самоприменимого компилятора языка

# Ограничения специализации функций. Количество аргументов

- Функции на Простом Рефале и Рефале-5λ принимают ровно один аргумент

- Явное описание форматов функций в виде жёстких выражений и обязательной проверки каждого вызова функций на предмет его соответствия формату

# Ограничения специализации функций. Формат аргумента

- Необходимо различать параметры, по которым ведётся специализация (статические), и по которым специализация не ведётся (динамические)

- Определение формата жесткого выражения, в котором статические и динамический параметры промаркированы

# Объявление специализации функции

- `$SPEC ИмяФункции ЖёсткоеВыражение;`

- ЖёсткоеВыражение — образец, не содержащий двух e-переменных на одном скобочном уровне
- Дополнительное ограничение:
    - Образец не должен содержать повторных переменных
- Имена переменных в образце не имеют значения, кроме первого символа:
    - Если имя переменной начинается с заглавной латинской буквы, то соответствующий ей параметр считается статическим параметром специализации

# Семантика специализации

- Для каждой функции допустимо не более одного объявления специализации

- Специализируемые функции должны быть определены в области видимости текущей единицы трансляции

- Если в образце спецификатора есть ссылки на функции, то они должны быть определены (например, если образец содержит АТД-скобки)

- Каждое образцовое выражение в определении функции должны быть уточнениями формата специализации, при этом статическим параметрам могут соответствовать только переменные того же типа

# Формальное описание специализации

- Задаётся функция с несколькими аргументами, где некоторые параметры помечаются статическими

  *f(x, yS, zS) = expr*.

- Для вызова функции *f(a, B(c d), E)* определяются подстановки статических параметров

  *y → B(c d), z→E*.

- Затем для набора подстановок строится специализированная функция

  *f1(x, y1, y2) = expr // {y→B(y1 y2), z→E}*,

  где тело функции образуется путём текстуальной замены параметров на их подстановки

- При этом если параметру соответствовал конструктор с переменными *y → B(c d)*, ему будут соответствовать несколько параметров по числу переменных *y1*, *y2*. Для параметра *z* в конструкторе переменных не было, поэтому в специализированной функции он отсутствует

- Вызов функции, соответственно заменяется на вызов экземпляра специализированной функции:

  *f(a, B(c d), E) → f1(a, c, d)*.

# Алгоритм специализации

- Сопоставление фактических параметров вызова с форматом специализации
- Анализ сопоставления
- Извлечение из этих аргументов переменные точки вызова
- Построение сигнатуры для запоминания просцепиализированных вызовов
- Построение определения специализированной функции и её вызова
- Изменение синтаксического дерева (замена вызова на сформированный и добавление новых определений)

# Пример

```
* исходный код

$SPEC F t.X e.2;

F {
  t.x A e.2 = t.x;
  t.y e.2 B = t.y t.y;
}

Go {
  = <F 10 'abc'> <F () 10 20>;
}
```

```
* после прохода обессахаривателя

$SPEC F t.X#0 e.2#0;

F {
  t.x#1 A e.2#1 = t.x#1;

  t.y#1 e.2#1 B = t.y#1 t.y#1;
}

Go {
  /* empty */ = <F 10 'abc'> <F () 10 20>;
}
```

```
* после проходов оптимизаций дерева

F {
  t.x#1 A e.2#1 = t.x#1;

  t.y#1 e.2#1 B = t.y#1 t.y#1;
}

Go {
  /* empty */ = <F@1 ('abc')> <F@2 (10 20)>;
}

F@1 {
  A e.2#1 = 10;

  e.2#1 B = 10 10;
}

F@2 {
  A e.2#1 = ();

  e.2#1 B = () ();
}
```

# Заключение