Суперкомпиляция для miniKanren

Автор: Екатерина Вербицкая

Лаборатория языковых инструментов JetBrains
Санкт-Петербургский государственный университет

15 декабря 2018

Реляционное программирование

Программа — отношение

$$append^o \subseteq [A] \times [A] \times [A]$$

$$\begin{aligned} append^o \quad = \quad \{ \quad & ([\,], [\,], [\,]); \\ & ([0], [\,], [0]); \\ & ([1], [\,], [1]); \\ & \ldots \\ & ([\,], [0], [0]); \\ & \ldots \\ & ([4], [2], [4, 2]); \\ & \ldots \\ & ([4, 2], [13], [4, 2, 13]); \\ & \ldots \\ \} & \end{aligned}$$

Пример программы на miniKanren

```
appendᵒ x y z =
  (x ≡ [] ∧ z ≡ y)
  ∨ (∃ h t z'
       (x ≡ h : t
       ∧ z ≡ h : z'
       ∧ appendᵒ t y z'))
```

Вычисление в реляционном программировании

$$
\begin{array}{llll}
append^o & [1] & [2,3] & q & \rightarrow & \{ & [1,2,3] & \} \\
append^o & [1] & q & [1,2,3] & \rightarrow & \{ & [2,3] & \} \\
append^o & q & [1] & [2,3] & \rightarrow & \{ & & \}
\end{array}
$$

Вычисление в реляционном программировании

$$append^o \quad q \quad p \quad [1,2] \quad \rightarrow \quad \{ \quad ([\,], [1,2]), \; ([1], [2]), \; ([1,2], [\,]) \quad \}$$

$$append^o \quad q \quad q \quad [2,4,2,4] \quad \rightarrow \quad \{ \quad [2,4] \quad \}$$

$$append^o \quad q \quad p \quad r \quad \rightarrow \quad \{ \quad ([\,], \; \_0, \; \_0), \; ([\_0], \; \_1, \; (\_0 : \_1)), \; ((\_0 : \_1), \; \_2, \; (\_0 : \_1 : \_2)) \; \ldots \quad \}$$

Направление вычислений

$$foo^o \subseteq A \times B$$

- $foo^o\ \alpha\ q : A \to [B]$
- $foo^o\ q\ \beta : B \to [A]$ — в “обратном” направлении
- $foo^o\ q\ p : () \to [(A \times B)]$

Направление вычислений

$$foo^o \subseteq A \times B$$

- $foo^o\ \alpha\ q : A \to [B]$
- $foo^o\ q\ \beta : B \to [A]$ — в “обратном” направлении
- $foo^o\ q\ p : () \to [(A \times B)]$

Время вычисления в разных направлениях часто отличается

Направление вычислений

$$foo^o \subseteq A \times B$$

- $foo^o\ \alpha\ q : A \to [B]$
- $foo^o\ q\ \beta : B \to [A]$ — в “обратном” направлении
- $foo^o\ q\ p : () \to [(A \times B)]$

Время вычисления в разных направлениях часто отличается

$$factorize\ num = mult^o\ [p, q]\ num$$

Цель

Улучшать производительность реляционных программ, не уменьшая декларативности подхода

- Для заданного направления
- Учитывая частично определенные входные данные

Оптимизации: известные данные

```
foo p q ∧ repeat x p
```

```
foo ⊆ [A] × [B]
foo x y =
 (x ≡ [] ∧ heavy y)
 ∨ (∃ h t (x ≡ h : t ∧ light y))

repeat ⊆ A × [A]
repeat x xs =
  ∃r (xs ≡ x : r ∧ repeat x r)
```

Оптимизации: известные данные

```
foo p q ∧ repeat x p
```

```
foo ⊆ [A] × [B]
foo x y =
 (x ≡ [] ∧ heavy y)
 ∨ (∃ h t (x ≡ h : t ∧ light y))

repeat ⊆ A × [A]
repeat x xs =
  ∃r (xs ≡ x : r ∧ repeat x r)
```

```
foo_r q
```

```
foo_r ⊆ [A] × [B]
foo_r y = light y
```

Оптимизации: промежуточные структуры данных

```
map f p q ∧ map g q r
```

```
map f ⊆ [A] × [B]
map f x y =
 (x ≡ [] ∧ y ≡ [])
 ∨ (∃ h t r ( x ≡ h : t
              ∧ y ≡ f h : r
              ∧ map f t r))
```

Оптимизации: промежуточные структуры данных

```
map f p q ∧ map g q r
```

```
map f ⊆ [A] × [B]
map f x y =
 (x ≡ [] ∧ y ≡ [])
 ∨ (∃ h t r ( x ≡ h : t
              ∧ y ≡ f h : r
              ∧ map f t r))
```

```
map_fg f g p r
```

```
map_fg f g ⊆ [A] × [B]
map_fg f g x y =
 (x ≡ [] ∧ y ≡ [])
 ∨ (∃ h t r ( x ≡ h : t
              ∧ y ≡ g (f h) : r
              ∧ map_fg f g t r))
```

Оптимизации: группировка вычислений

```
sum p s ∧ len p l
```

```
sum ⊆ [A] × Int
sum x s = (x ≡ [] ∧ s ≡ 0)
        ∨ (∃ h t r
             (x ≡ h : t
             ∧ sum t r
             ∧ s = r + h))

len ⊆ [A] × Int
len x l = (x ≡ [] ∧ l ≡ 0)
        ∨ (∃ h t m
             (x ≡ h : t
             ∧ len t m
             ∧ l = m + 1 ))
```

Оптимизации: группировка вычислений

```
sum p s ∧ len p l
```

```
sum ⊆ [A] × Int
sum x s = (x ≡ [] ∧ s ≡ 0)
        ∨ (∃ h t r
             (x ≡ h : t
             ∧ sum t r
             ∧ s = r + h))

len ⊆ [A] × Int
len x l = (x ≡ [] ∧ l ≡ 0)
        ∨ (∃ h t m
             (x ≡ h : t
             ∧ len t m
             ∧ l = m + 1 ))
```

```
sum_len p s l
```

```
sum_len ⊆ [A] × Int × Int
sum_len x s l =
  (x ≡ [] ∧ s ≡ 0 ∧ l ≡ 0)
  ∨ (∃ h t r m
       (x ≡ h : t
       ∧ sum_len t r m
       ∧ s = r + h
       ∧ l = m + 1))
```

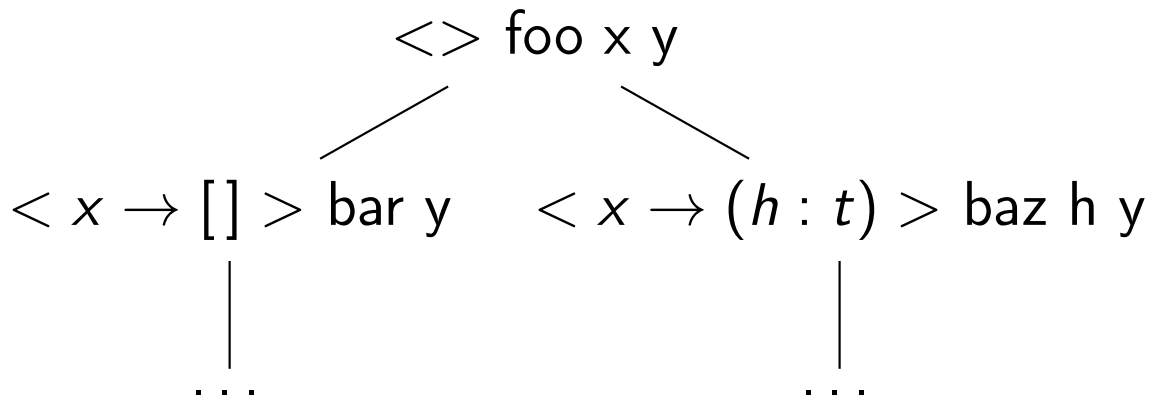
За счет чего можно улучшать производительность

- Статически вычислить все, что можно: специализация
  - Если известны некоторые аргументы
  - Если известно направление вычисления
- Не делать одну и ту же работу дважды
  - Избегать промежуточные значения: deforestation
  - Группировать вычисления, выполняемые во время обхода одной структуры данных: tupling

Все это может делать конъюнктивная частичная дедукция (суперкомпиляция) для логических языков

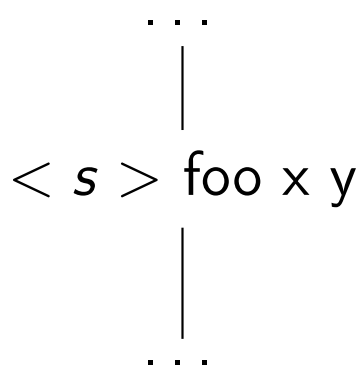
Суперкомпиляция для miniKanren: дерево процессов

```
foo ⊆ [A] × [B]
foo x y =
 (x ≡ [] ∧ bar y)
 ∨ (∃ h t
     (x ≡ h : t
      ∧ baz h y))
```



В какой момент завершать символические вычисления?

Дерево процессов: применение отношения



- Если цель (с точностью до подстановки) встречалась раньше, перестаем исследовать эту ветвь
- Если цель *похожа* на какого-то ее предка, попробовать ее *абстрагировать* и продолжить строить дерево
- Если цель ни на что не похожа, продолжаем строить дерево для применения отношения

Трудности

- Что значит, что цель *похожа* на другую цель?
- Как *абстрагировать*?
- Проверка на похожесть и абстракция связаны между собой существенно теснее, чем в функциональных языках
- Как учитывается связь между переменными?

Ошибка: считать, что решение для функциональных языков применимо для трансформации реляционных программ

Текущее положение дел

- Попробовала реализовать несколько версий суперкомпиляции
  - Некоторые программы успешно преобразовываются
  - Некоторые преобразовываются с ухудшением производительности
  - На некоторых суперкомпиляция не завершаетсяы
- Поняла, что все делала неправильно
- Начала *буквально* адаптировать для miniKanren решения для логического программирования

Дальнейшие планы

- Доведение до работоспособности конъюнктивной частичной дедукции
- “Негативная” суперкомпиляция
- Поддержка отношений высшего порядка
- Адаптация более мощных техник символьных вычислений