



Суперкомпиляция для miniKanren

Автор: Екатерина Вербицкая

Лаборатория языковых инструментов JetBrains
Санкт-Петербургский государственный университет

15 декабря 2018

Реляционное программирование

Программа — отношение

$$\mathit{append}^o \subseteq [A] \times [A] \times [A]$$

$$\mathit{append}^o = \{ \begin{array}{l} ([], [], []); \\ ([0], [], [0]); \\ ([1], [], [1]); \\ \dots \\ ([], [0], [0]); \\ \dots \\ ([4], [2], [4, 2]); \\ \dots \\ ([4, 2], [13], [4, 2, 13]); \\ \dots \end{array} \}$$

Пример программы на miniKanren

```
appendo x y z =  
  (x ≡ [] ∧ z ≡ y)  
  ∨ (∃ h t r  
    ( x ≡ h : t  
      ∧ z ≡ h : r  
      ∧ appendo t y r))
```

Вычисление в реляционном программировании

$append^o$	$[1]$	$[2, 3]$	q	\rightarrow	$\{$	$[1, 2, 3]$	$\}$
$append^o$	$[1]$	q	$[1, 2, 3]$	\rightarrow	$\{$	$[2, 3]$	$\}$
$append^o$	q	$[1]$	$[2, 3]$	\rightarrow	$\{$		$\}$

Вычисление в реляционном программировании

$$\text{append}^o q p [1, 2] \rightarrow \{ \begin{array}{l} ([], [1, 2]), \\ ([1], [2]), \\ ([1, 2], []) \end{array} \}$$

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

$$\text{append}^o q p r \rightarrow \{ \begin{array}{l} ([], \alpha, \alpha), \\ ([\alpha], \beta, (\alpha : \beta)) \\ ((\alpha : \beta), \gamma, (\alpha : (\beta : \gamma))) \\ \dots \end{array} \}$$

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

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

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

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

Время вычисления в разных направлениях часто отличается

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

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

Время вычисления в разных направлениях часто отличается

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

Улучшать производительность реляционных программ, не уменьшая декларативности подхода

- Для заданного направления
- Учитывая частично определенные входные данные

Оптимизации: известные данные

$\text{foo } p \ q \wedge \text{repeat } x \ p$

$\text{foo} \subseteq [A] \times [B]$
 $\text{foo } x \ y =$
 $(x \equiv [] \wedge \text{heavy } y)$
 $\vee (\exists h \ t \ (x \equiv h : t \wedge \text{light } y))$

$\text{repeat} \subseteq A \times [A]$
 $\text{repeat } x \ xs =$
 $\exists r \ (xs \equiv x : r \wedge \text{repeat } x \ r)$

Оптимизации: известные данные

$\text{foo } p \ q \wedge \text{repeat } x \ p$

$\text{foo} \subseteq [A] \times [B]$
 $\text{foo } x \ y =$
 $(x \equiv [] \wedge \text{heavy } y)$
 $\vee (\exists h \ t \ (x \equiv h : t \wedge \text{light } y))$

$\text{repeat} \subseteq A \times [A]$
 $\text{repeat } x \ xs =$
 $\exists r \ (xs \equiv x : r \wedge \text{repeat } x \ r)$

$\text{foo_r } q$

$\text{foo_r} \subseteq [A] \times [B]$
 $\text{foo_r } y = \text{light } y$

Оптимизации: промежуточные структуры данных

```
map f p q  $\wedge$  map g q r
```

```
map f  $\subseteq$  [A]  $\times$  [B]  
map f x y =  
  (x  $\equiv$  []  $\wedge$  y  $\equiv$  [])  
   $\vee$  ( $\exists$  h t r ( x  $\equiv$  h : t  
                   $\wedge$  y  $\equiv$  f h : r  
                   $\wedge$  map f t r))
```

Оптимизации: промежуточные структуры данных

$$\text{map } f \text{ p } \mathbf{q} \wedge \text{map } g \text{ q } r$$
$$\begin{aligned} \text{map } f &\subseteq [A] \times [B] \\ \text{map } f \text{ x } y &= \\ & (x \equiv [] \wedge y \equiv []) \\ & \vee (\exists h \text{ t } r \ (x \equiv h : t \\ & \qquad \qquad \wedge y \equiv f \ h : r \\ & \qquad \wedge \text{map } f \text{ t } r)) \end{aligned}$$
$$\text{map_fg } f \text{ g } p \text{ r}$$
$$\begin{aligned} \text{map_fg } f \text{ g} &\subseteq [A] \times [B] \\ \text{map_fg } f \text{ g } x \text{ y} &= \\ & (x \equiv [] \wedge y \equiv []) \\ & \vee (\exists h \text{ t } r \ (x \equiv h : t \\ & \qquad \qquad \wedge y \equiv \mathbf{g} \ (f \ h) : r \\ & \qquad \wedge \text{map_fg } f \text{ g } t \text{ r})) \end{aligned}$$

Оптимизации: группировка вычислений

$\text{sum } p \ s \wedge \text{len } p \ 1$

$\text{sum} \subseteq [A] \times \text{Int}$
 $\text{sum } x \ s = (x \equiv [] \wedge s \equiv 0)$
 $\vee (\exists h \ t \ r$
 $(x \equiv h : t$
 $\wedge \text{sum } t \ r$
 $\wedge s = r + h))$

$\text{len} \subseteq [A] \times \text{Int}$
 $\text{len } x \ l = (x \equiv [] \wedge l \equiv 0)$
 $\vee (\exists h \ t \ m$
 $(x \equiv h : t$
 $\wedge \text{len } t \ m$
 $\wedge l = m + 1))$

Оптимизации: группировка вычислений

$\text{sum } p \ s \wedge \text{len } p \ l$

$\text{sum} \subseteq [A] \times \text{Int}$
 $\text{sum } x \ s = (x \equiv [] \wedge s \equiv 0)$
 $\vee (\exists h \ t \ r$
 $(x \equiv h : t$
 $\wedge \text{sum } t \ r$
 $\wedge s = r + h))$

$\text{len} \subseteq [A] \times \text{Int}$
 $\text{len } x \ l = (x \equiv [] \wedge l \equiv 0)$
 $\vee (\exists h \ t \ m$
 $(x \equiv h : t$
 $\wedge \text{len } t \ m$
 $\wedge l = m + 1))$

$\text{sum_len } p \ s \ l$

$\text{sum_len} \subseteq [A] \times \text{Int} \times \text{Int}$
 $\text{sum_len } x \ s \ l =$
 $(x \equiv [] \wedge s \equiv 0 \wedge l \equiv 0)$
 $\vee (\exists h \ t \ r \ m$
 $(x \equiv h : t$
 $\wedge \text{sum_len } t \ r \ m$
 $\wedge s = r + h$
 $\wedge l = m + 1))$

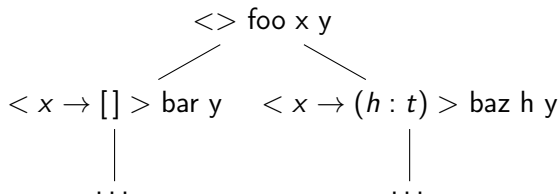
За счет чего можно улучшать производительность

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

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

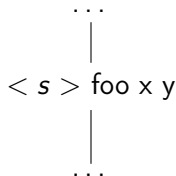
Суперкомпиляция для miniKanren: дерево процессов

$$\begin{aligned} \text{foo} &\subseteq [A] \times [B] \\ \text{foo } x \ y &= \\ & (x \equiv [] \wedge \text{bar } y) \\ & \vee (\exists h \ t \\ & \quad (x \equiv h : t \\ & \quad \wedge \text{baz } h \ y)) \end{aligned}$$



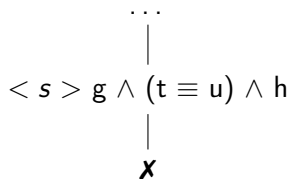
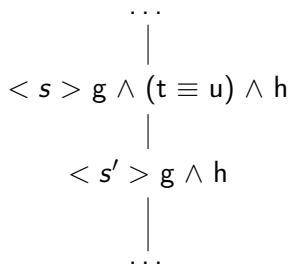
В какой момент завершать символические вычисления?

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

Дерево процессов: конъюнкция



- Все унификации проталкиваем вверх и вычисляем в подстановки
- Остается конъюнкция применений отношений
 - Проверяем на совпадение с предками
 - Проверяем на *похожесть* с предками
 - Иначе разворачиваем применения отношений

Ошибка: слишком агрессивное разворачивание вызовов

- Определить, что значит, что цели *похожи*
- Определить, как *абстрагировать*
- Определить, как учитывать связи между переменными
- Проверка на похожесть и абстракция связаны между собой существенно теснее, чем в функциональных языках

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

Примеры, которые уже работают

```
appendo x y z =  
  (x  $\equiv$  []  $\wedge$  y  $\equiv$  z)  
   $\vee$  ( $\exists$  h t r (x  $\equiv$  h : t  $\wedge$  z  $\equiv$  h : r  $\wedge$  appendo t y r))
```

```
appendo [] y z
```

Примеры, которые уже работают

$$\begin{aligned} \text{appendo } x \ y \ z = \\ & (x \equiv [] \wedge y \equiv z) \\ & \vee (\exists \ h \ t \ r \ (x \equiv h : t \wedge z \equiv h : r \wedge \text{appendo } t \ y \ r)) \end{aligned}$$
$$\text{appendo } [] \ y \ z$$
$$\text{empty_appendo } y \ z$$
$$\text{empty_appendo } y \ z = y \equiv z$$

Примеры, которые уже работают

```
appendo x y z =  
  (x  $\equiv$  []  $\wedge$  y  $\equiv$  z)  
   $\vee$  ( $\exists$  h t r (x  $\equiv$  h : t  $\wedge$  z  $\equiv$  h : r  $\wedge$  appendo t y r))
```

```
appendo [1,2,3] y z
```

Примеры, которые уже работают

```
appendo x y z =  
  (x  $\equiv$  []  $\wedge$  y  $\equiv$  z)  
   $\vee$  ( $\exists$  h t r (x  $\equiv$  h : t  $\wedge$  z  $\equiv$  h : r  $\wedge$  appendo t y r))
```

```
appendo [1,2,3] y z
```

```
app_123 y z
```

```
app_123 y z =  
  z  $\equiv$  1 : (2 : (3 : y))
```


Примеры, которые уже работают

```
appendo x y z =  
  (x ≡ [] ∧ y ≡ z)  
  ∨ (∃ h t r (x ≡ h : t ∧ z ≡ h : r ∧ appendo t y r))
```

```
appendo x [1,2,3] z
```

Примеры, которые уже работают

```
appendo x y z =  
  (x ≡ [] ∧ y ≡ z)  
  ∨ (∃ h t r (x ≡ h : t ∧ z ≡ h : r ∧ appendo t y r))
```

```
appendo x [1,2,3] z
```

```
app_123 x z
```

```
app_123 x z =  
  (x ≡ [] ∧ z ≡ [1,2,3])  
  ∨ (∃ h t r ( x ≡ h : t  
                ∧ z ≡ h : r  
                ∧ app_123 t r))
```

Примеры, которые уже работают

$$\begin{aligned} \text{appendo } x \ y \ z = \\ (x \equiv [] \wedge y \equiv z) \\ \vee (\exists h \ t \ r \ (x \equiv h : t \wedge z \equiv h : r \wedge \text{appendo } t \ y \ r)) \end{aligned}$$
$$\begin{aligned} \text{appendo } x \ y \ i \\ \wedge \text{appendo } i \ z \ r \end{aligned}$$

Примеры, которые уже работают

```
appendo x y z =  
  (x ≡ [] ∧ y ≡ z)  
  ∨ (∃ h t r (x ≡ h : t ∧ z ≡ h : r ∧ appendo t y r))
```

```
appendo x y i  
  ∧ appendo i z r
```

```
double_app x y z r
```

```
double_app x y z r = ∃ h t s  
  (x ≡ [] ∧ y ≡ [] ∧ z ≡ r)  
  ∨ ( x ≡ []  
    ∧ y ≡ h : t  
    ∧ r ≡ h : s  
    ∧ appendo t z s)  
  ∨ ( x ≡ h : t  
    ∧ r ≡ h : s  
    ∧ double_app t y z s)
```

- Попробовали реализовать несколько версий суперкомпиляции
 - Некоторые программы успешно преобразовываются
 - Некоторые преобразовываются с ухудшением производительности
 - На некоторых суперкомпиляция не завершается
- Стало понятно, что мы делали неправильно
- Решено сначала *буквально* адаптировать для miniKanren технику конъюнктивной частичной дедукции для логических языков

Дальнейшие планы

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