Преобразование типизированных функций в реляционную форму

В рамках проекта лаборатории JetBrains

Петр Алексеевич Лозов, 646 группа
Научный руководитель: к.ф.-м.н., доцент Д.Ю. Булычев
Рецензент: к.ф.-м.н. В.А. Павлов

Санкт-Петербургский Государственный Университет
Кафедра системного программирования

24 мая 2018г.

Представление программы в виде отношения

Реляционное программирование

Представление программы в виде отношения

Исполнение программ в различных направлениях

Реляционное программирование

Представление программы в виде отношения

Исполнение программ в различных направлениях

Реляционный язык - `miniKanren`

OCanren — реляционное расширение OCaml

```
type nat = O | S of nat

val add^o : nat → nat → nat → 𝔊

let rec add^o x y z =
  (x ≡ O ∧ y ≡ z) ∨
  (fresh (x' z') (
    (x ≡ S x') ∧
    (z ≡ S z') ∧
    (add^o x' y z')))
```

OCanren — реляционное расширение OCaml

```
type nat = O | S of nat

val add^o : nat → nat → nat → 𝔊

let rec add^o x y z =
  (x ≡ O ∧ y ≡ z) ∨
  (fresh (x' z') (
    (x ≡ S x') ∧
    (z ≡ S z') ∧
    (add^o x' y z')))
```

Запрос			Поток
add^o '3' '4'	q	$\rightsquigarrow$	$[q =$ '7'$]$

OCanren — реляционное расширение OCaml

```
type nat = O | S of nat

val add^o : nat → nat → nat → 𝔊

let rec add^o x y z =
  (x ≡ O ∧ y ≡ z) ∨
  (fresh (x' z') (
    (x ≡ S x') ∧
    (z ≡ S z') ∧
    (add^o x' y z')))
```

Запрос				Поток
add^o	'3'	'4'	q	$\rightsquigarrow$	$[q =$ '7'$]$
add^o	q	'4'	'7'	$\rightsquigarrow$	$[q =$ '3'$]$

OCanren — реляционное расширение OCaml

```
type nat = O | S of nat

val add^o : nat → nat → nat → 𝔊

let rec add^o x y z =
  (x ≡ O ∧ y ≡ z) ∨
  (fresh (x' z') (
    (x ≡ S x') ∧
    (z ≡ S z') ∧
    (add^o x' y z')))
```

Запрос					Поток
add^o	'3'	'4'	q	$\rightsquigarrow$	$[q =$ '7'$]$
add^o	q	'4'	'7'	$\rightsquigarrow$	$[q =$ '3'$]$
add^o	q	r	'2'	$\rightsquigarrow$	$[q =$ '0', $r =$ '2'; $q =$ '1', $r =$ '1'; $q =$ '2', $r =$ '0'$]$

Реляционное программирование даром

Реляционное преобразование $[\![\bullet]\!]$ создает реляционные программы из функциональных

Реляционное программирование даром

Реляционное преобразование $[\![\bullet]\!]$ создает реляционные программы из функциональных

Не требует знаний о реляционном программировании

Реляционное программирование даром

Реляционное преобразование $[\![\bullet]\!]$ создает реляционные программы из функциональных

Не требует знаний о реляционном программировании

Позволяет исполнять функциональные программы реляционно

Цель и задачи

Разработать метод преобразования функциональных программ в реляционную форму

Задачи:

- Разработать формальные модели функционального и реляционного языков
- Разработать преобразование типизированных функций в реляционную форму
- Доказать статическую и динамическую корректность преобразования
- Провести опробацию преобразования

Подмножество OCaml для реляционного преобразования

x	(переменная)
$\lambda x.A$	(абстракция)
$A\ B$	(применение)
$C^n(A_1, \ldots, A_n)$	(конструктор)
$A = B$	(сравнение)
$\texttt{let}\ x = A\ \texttt{in}\ B$	(let-связывание)
$\texttt{let rec}\ f = \lambda x.A\ \texttt{in}\ B$	(рекурсивное let-связывание)
$\texttt{match}\ A\ \texttt{with}\ \{p_i \rightarrow B_i\}$	(сопоставление с образцом)

Преобразование типов

$$
\begin{aligned}
[\![g]\!] &= g \to \mathfrak{G} \\
[\![t_1 \to t_2]\!] &= [\![t_1]\!] \to [\![t_2]\!]
\end{aligned}
$$

Преобразование типов

$$
\begin{aligned}
[\![g]\!] &= g \to \mathfrak{G} \\
[\![t_1 \to t_2]\!] &= [\![t_1]\!] \to [\![t_2]\!]
\end{aligned}
$$

Примеры:

$$
\begin{aligned}
[\![int]\!] &= int \to \mathfrak{G} \\
[\![string \to int]\!] &= (string \to \mathfrak{G}) \to (int \to \mathfrak{G})
\end{aligned}
$$

Преобразование выражений: операторы λ-исчисления

$$
\begin{array}{rcll}
[\![\mathtt{x}]\!] & \rightsquigarrow & \mathtt{x} & [\text{Var}] \\
[\![\lambda\ \mathtt{x}.\ A]\!] & \rightsquigarrow & \lambda\ \mathtt{x}.\ [\![A]\!] & [\text{Abst}] \\
[\![A\ B]\!] & \rightsquigarrow & [\![A]\!]\ [\![B]\!] & [\text{App}] \\
[\![\mathtt{let}\ \mathtt{x} = A\ \mathtt{in}\ B]\!] & \rightsquigarrow & \mathtt{let}\ \mathtt{x} = [\![A]\!]\ \mathtt{in}\ [\![B]\!] & [\text{Let}] \\
[\![\mathtt{let\ rec}\ \mathtt{x} = A\ \mathtt{in}\ B]\!] & \rightsquigarrow & \mathtt{let\ rec}\ \mathtt{x} = [\![A]\!]\ \mathtt{in}\ [\![B]\!] & [\text{LetRec}]
\end{array}
$$

(структура выражений не изменяется)

Преобразование выражений: конструкторы

$$\llbracket \texttt{Nil} \rrbracket \quad \rightsquigarrow \quad (\equiv \texttt{Nil})$$

$$\llbracket \texttt{Cons}\ A\ B \rrbracket \quad \rightsquigarrow \quad \begin{array}{l} \lambda\ \texttt{r}. \\ \quad \texttt{fresh}\ (\texttt{p}\ \texttt{q}) \\ \quad\quad (\llbracket A \rrbracket\ \texttt{p}) \ \wedge \\ \quad\quad (\llbracket B \rrbracket\ \texttt{q}) \ \wedge \\ \quad\quad (\texttt{r} \equiv \texttt{Cons}\ \texttt{p}\ \texttt{q}) \end{array}$$

Преобразование выражений: сравнение

$$[\![A = B]\!] \quad \rightsquigarrow$$

```
λ r.
  fresh (p q) (
    ([[A]] p) ∧
    ([[B]] q) ∧ (
      ((p ≡ q) ∧ (r ≡ true)) ∨
      ((p ≢ q) ∧ (r ≡ false))
    ))
```

Преобразование выражений: сопоставление с образцом

$$[\![\ \texttt{match}\ A\ \texttt{with} \mid \texttt{Nil} \to B \mid \texttt{Cons x xs} \to C\ ]\!] \rightsquigarrow$$

$$\begin{array}{l}
\lambda\ \texttt{r}. \\
\quad \texttt{fresh (a) (} \\
\quad\quad ([\![A]\!]\ \texttt{a}) \wedge (\\
\quad\quad ((\texttt{a} \equiv \texttt{Nil}) \wedge ([\![B]\!]\ \texttt{r})) \vee \\
\quad\quad (\texttt{fresh (x' xs') (} \\
\quad\quad\quad (\texttt{a} \equiv \texttt{Cons(x',xs')}) \\
\quad\quad\quad ((\lambda\ \texttt{x xs q} \to [\![C]\!]) \\
\quad\quad\quad (\equiv \texttt{x'})\ (\equiv \texttt{xs'})\ \texttt{r})) \\
\quad\quad))))
\end{array}$$

Доказательство корректности

Статическая корректность:

$$\forall P : t \Rightarrow [\![P]\!] : [\![t]\!]$$

Доказательство корректности

Статическая корректность:

$$\forall P : t \Rightarrow [\![P]\!] : [\![t]\!]$$

Частичная динамическая корректность:

$$\forall P : g, P \rightarrow^* v \Rightarrow \texttt{fresh (q)}\ ([\![P]\!]\ \texttt{q}) \rightsquigarrow^* [\texttt{q}{=}v]$$

Апробация: реализация преобразования

- Входной язык: подмножество `OCaml`
- Выходной язык: `OCanren`
- Реализован на `OCaml`

Апробация: вывод типов для системы Хиндли-Милнера

- Выражения
  - λ-исчисление
  - `let`-связывания

- Функция вывода типов

```
val type_inference : term → typ
```

Апробация: вывод типов для системы Хиндли-Милнера

$$[\![\texttt{type\_inference}]\!]\ \text{`}\lambda x \to x\text{`}\ \square \rightsquigarrow^*$$
$$[\square = \text{`}a \to a\text{`}]$$

$$[\![\texttt{type\_inference}]\!]$$
$$\text{`}\texttt{let f} = \square\ \texttt{in f}\ (\lambda\ \texttt{x}\ \to \texttt{f x})\text{`}\ \ \text{`}a \to a\text{`}\ \rightsquigarrow^*$$
$$[\square = \text{`}\lambda\ \boxed{0}\ \to \boxed{0}\text{`};\ \ldots]$$

Апробация: вывод типов для системы Хиндли-Милнера

$[\![\texttt{type\_inference}]\!]\ \square\ \texttt{'}a\texttt{'} \rightsquigarrow^*$
  [] (Не остановится, но поток будет пустой)

$[\![\texttt{type\_inference}]\!]\ \square\ \texttt{'}a \to a\texttt{'} \rightsquigarrow^*$
  [
    $\square = \texttt{'}\lambda\ \boxed{0} \to \boxed{0}\texttt{'};$
    $\square = \texttt{'}\lambda\ \boxed{0} \to (\lambda\ \boxed{1} \to \boxed{1})\ \boxed{0}\texttt{'};$
    $\square = \texttt{'}\lambda\ \boxed{0} \to \texttt{let}\ \boxed{1} = \boxed{2}\ \texttt{in}\ \boxed{0}\texttt{'}\ (\boxed{0} \not\equiv \boxed{1});$
    $\square = \texttt{'}(\lambda\ \boxed{0} \to \boxed{0})\ (\lambda\ \boxed{1} \to \boxed{1})\texttt{'};$
    ...
  ]

Конференции

- Конвертация функций высшего порядка в реляционную форму
  - PLC-2017
  - Петр Лозов
- Relational conversion for `OCaml`
  - ML Workshop (ICFP-2017)
  - Петр Лозов, Дмитрий Булычев
- Typed Relational Conversion
  - TFP-2017
  - Петр Лозов, Андрей Вяткин, Дмитрий Булычев
  - Статья опубликована в журнале Lecture Notices in Computer Science

Результаты

- Разработаны формальные модели функционального и реляционного языков
- Разработано преобразование типизированных функций в реляционную форму
- Доказана статическая и частичная динамическая корректность преобразования
- Проведена апробация преобразования