Реализация сравнения различных представлений лямбда-термов

предварительные результаты

студент: Жаворонков Эдгар

научный руководитель: В.И. Исаев

СПб АУ НОЦНТ РАН

11/05/2017

Введение. λ-термы и связанные определения

1. Термы λ-исчисления строятся индуктивно, путем применения и создания анонимных функций.
2. Пусть $V = \{x, y, \ldots\}$ – счетное множество переменных. Тогда множество λ-термов в абстрактном синтаксисе определяется как $\Lambda ::= V|\Lambda\ \Lambda|\lambda V.\Lambda$
3. Примеры термов – $\lambda fgx.f\,x\,(g\,x)$ или $(\lambda x.f\,(x\,x))(\lambda x.f\,(x\,x))$.

Введение. λ-термы и связанные определения(2)

1. Переменная, которая стоит после λ называется *связанной*. Соответственно, до абстракции она была *свободной*.
2. В первом примере f — связана, а во втором — свободна.
3. Имена связанных переменных не важны. $\lambda fgx.f\ x\ (g\ x)$ при применении к аргументам ведет себя так же, как и $\lambda xyz.x\ z\ (y\ z)$.

Введение. λ-термы и связанные определения(3)

1. На Λ можно задать отношение α-эквивалентности, которое и говорит о том, что имена формальных параметров лямбд не важны.
2. Для λ-термов определена операция подстановки $M[x \mapsto N]$. Она описывает, что происходит когда абстракция применяется к аргументу.
3. Мы записывали термы, используя именованные переменные. Но это не единственный способ записи. Помимо него есть еще неименованное представление и монадическое. Но о них позже.

Мотивация

1. Соответствие Карри-Говарда говорит нам о том, что между логикой и теорией типов есть прямая связь.
2. Основная мысль состоит в том, что для доказательства утверждений мы можем писать λ-термы(читай – программы).
3. Чем больше логических связок у нас есть, тем более мощные теории типов приходится использовать.

Мотивация(2)

1. Теории с зависимыми типами позволяют доказывать утверждения в которых есть кванторы($\forall x P(x)$ или $\exists x P(x)$).
2. Известные примеры таких теорий – исчисление конструкций или интуиционистская теория типов Мартин-Лёфа.
3. Их расширения лежат в основе систем автоматического доказательства теорем Coq и Agda.

Мотивация(3)

1. Для доказательств иногда полезна функциональная экстенсиональность.
2. В Agda она постулируется, притом сами разработчики языка не рекомендуют использовать постулаты по очевидным причинам.
3. Хочется доказывать свойства термов, не используя "плохие" конструкции. Есть ли такая теория типов(такой язык)?

Мотивация(3)

1. Для доказательств иногда полезна функциональная экстенсиональность.
2. В Agda она постулируется, притом сами разработчики языка не рекомендуют использовать постулаты по очевидным причинам.
3. Хочется доказывать свойства термов, не используя "плохие" конструкции. Есть ли такая теория типов(такой язык)?
4. Да, есть. Он называется Vclang[1] и основан на гомотопической теории типов с типом интервала.

[1] https://github.com/valis/vclang

Цель и задачи

Цель работы – формализовать различные представления λ-термов и их свойства.

Задачи:

1. Описать типы данных для термов.
2. Описать свойства различных представлений(α-эквивалентность, операция подстановки etc.).
3. Показать, что различные представления эквивалентны между собой.
4. Для каждого представления доказать соответствующие свойства.

Существующие решения

1. Задача формализации λ-исчисления довольно популярна.
2. Одно из решений – A Formalization of Typed and Untyped λ-Calculi in SSReflect-Coq and Agda2. [2]
3. В нем, как и в аналогичных описывается только одно представление.
4. Кроме того, нам не очень нравится Agda как язык реализации. В нем нет конструкций, которыми нам хотелось бы воспользоваться(фактор-типы).

[2] https://github.com/pi8027/lambda-calculus

Решение. Именованное представление

Тип данных для термов:

```
data Term =
    Var Name
  | App Term Term
  | Lam Name Term
```

1. Нужно разрешимое равенство на *Name*.
2. α-эквивалентность определяется индукцией по структуре терма плюс некоторый трюк в случае абстракции.
3. Подстановка определяется аналогично индукцией, но для абстракций снова необходим трюк!

Решение. Именованное представление(2)

1. Все утверждения о подстановке рассматривают термы с точностью до α-эквивалентности.
2. Свойства подстановки формулируются следующим образом:

$$
\begin{aligned}
&x[x \mapsto t] =_\alpha t \\
&t[x \mapsto x] =_\alpha t \\
&t[x \mapsto N][y \mapsto M] =_\alpha t[y \mapsto M][x \mapsto N[y \mapsto M]] (x \notin FV(M))
\end{aligned}
$$

3. Доказываются очень нудной индукцией по терму t.

Решение. Неименованное представление

Термы:

```
data Term (n : Nat) =
    Var (i : Fin n)
    | App (Term n) (Term n)
    | Lam (Term (suc n))
```

1. Подстановка в таком представлении – полная(во все переменные)
2. Так как нет имен переменных, то и определяется она намного проще.

Решение. Монадическое представление

Термы:

```
data Term (V : Set) =
    Var V
    | App (Term V) (Term V)
    | Lam (Term (Maybe V))
```

1. Нетрудно заметить, что это функтор.
2. Чуть менее очевидно, но это монада!
3. Монадные законы в точности описывают свойства подстановки.
4. Для их доказательств удобно использовать функциональную экстенсиональность, которая в нашей теории конструируется, а не постулируется.

Результаты

1. Написаны преобразования:
   1. Из именованных термов в неименованные.
   2. Из неименованных в монадические.
   3. И обратно.
2. Доказано, что преобразования – биекции.
3. Для именованных термов описана α-эквивалентность.
4. Для каждого представления определена операция подстановки.
5. Для каждого представления доказаны свойства термов:
   1. Преобразование именованного терма в неименованный уважает α-эквивалентность.
   2. Свойства операции подстановки.

Спасибо за внимание!
Вопросы?

Github repo:

https://github.com/edgarzhavoronkov/vclang-lib/tree/lambda_calculus/test/LC