Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Санкт-Петербургский национальный исследовательский
Академический университет Российской академии наук»
Центр высшего образования

Кафедра математических и информационных технологий

Машкин Эдельвейс Захарович

Пустое подмножество как замкнутое множество

Магистерская диссертация

Допущена к защите.
Зав. кафедрой:
д. ф.-м. н., профессор Омельченко А. В.

Научный руководитель:
д. ф.-м. н., профессор Выбегалло А. А.

Рецензент:
ст. преп. Привалов А. И.

Санкт-Петербург
2017

SAINT-PETERSBURG ACADEMIC UNIVERSITY
Higher education centre

Department of Mathematics and Information Technology

Edelweis Mashkin

Empty subset as closed set

Graduation Thesis

Admitted for defence.
Head of the chair:
professor Alexander Omelchenko

Scientific supervisor:
professor Amvrosy Vibegallo

Reviewer:
assistant Alexander Privalov

Saint-Petersburg
2017

Оглавление

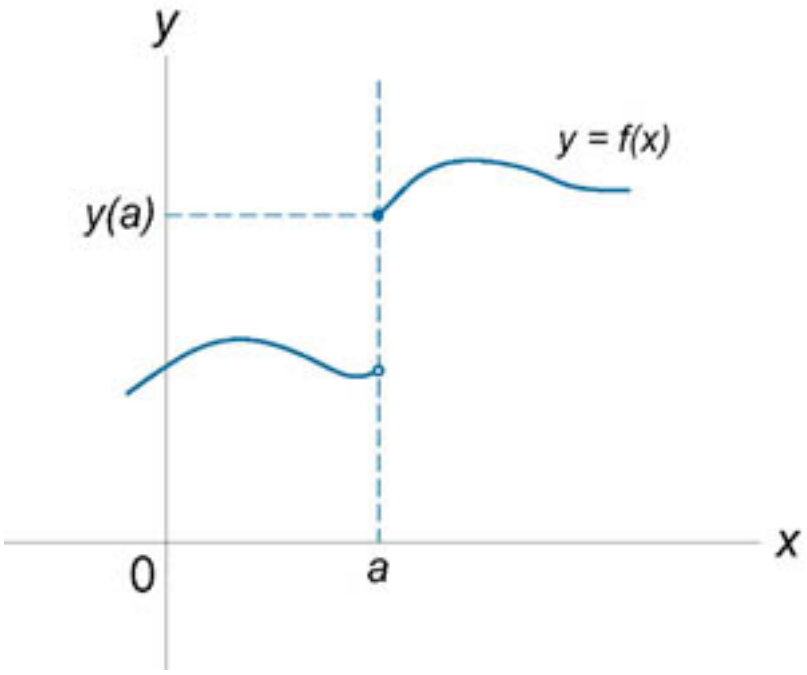

Рис. 1: Разрыв функции

Введение

$$\lim_{x \to a} \frac{f(x)}{g(x)} = \lim_{x \to a} \frac{f'(x)}{g'(x)} \tag{1}$$

1. Зависимые языки - что это и проч и экспозиция

1.1. Проверка типов в зависимых языках

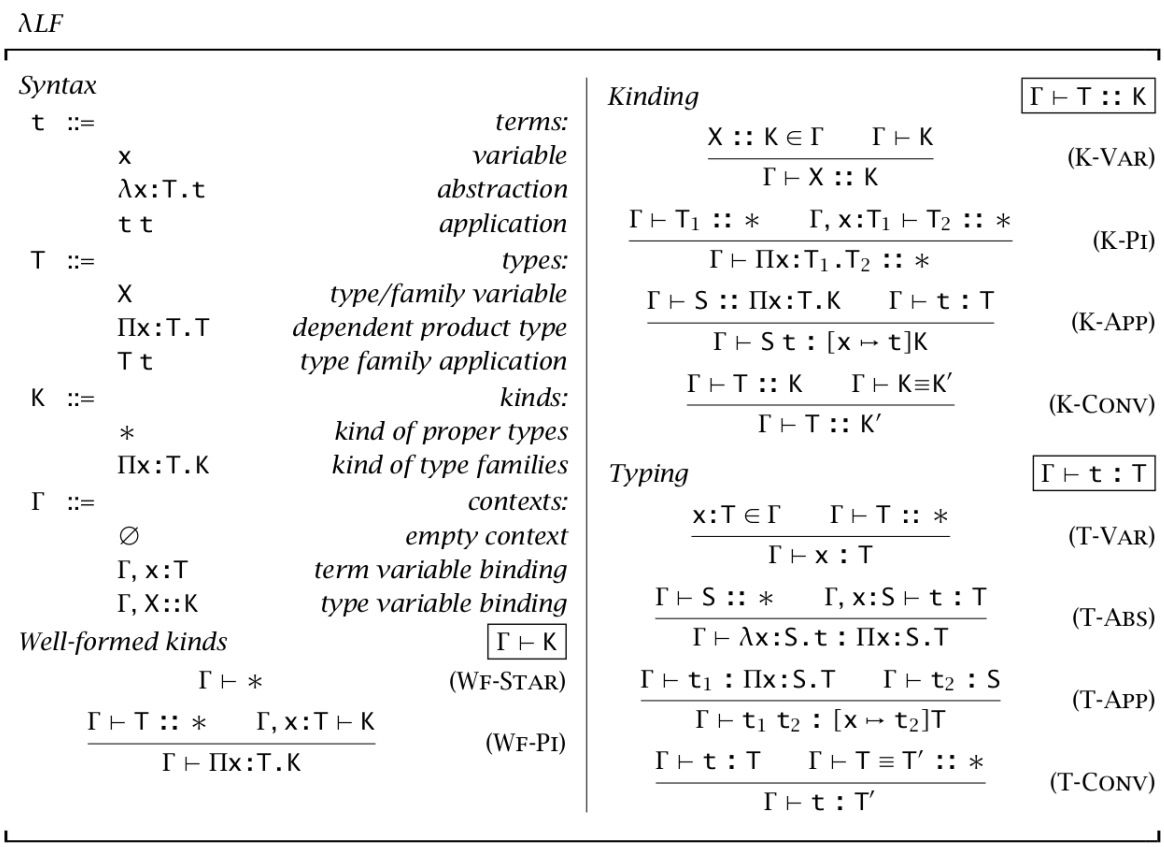

Рис. 2: Язык с лямбдой и П-типами

2. Определение языка

Вдохновением данной работы послужила статьи [2] и [1]. Поэтому сам язык спецификации выглядит как язык описания алгебраических теорий.

Начнем с примера языка с зависимыми типами (рис.2) [3, Глава 2.1]

Так этот язык будет выглядеть в нашем языке[1]

```
DependentSorts:
  tm, ty
FunctionalSymbols:
  lam: (ty, 0)*(tm, 1) -> tm
  app: (tm, 0)*(tm, 0)*(ty, 1) -> tm
  pi : (ty, 0)*(ty, 1) -> ty
Axioms:
  K-Pi =
    forall T1 : ty, x.T2 : ty
      x : T1 |- T2 def |--- |- pi(T1, x.T2) def
```

[1]правила связанные с кайндами опущены для простоты

```
  TAbs =
    forall S : ty, x.T : ty, x.t : tm
      x : S |- t : T |--- |- lam(S, x.t) : pi(S, x.T)
  TApp =
    forall t1 : tm, t2 : tm, S : ty, x.T : ty
           |- t1 : pi(S, x.T),
           |- t2 : S,
      x : S |- T def
      |-------------------------
      |- app(t1, t2, x.T) : T[x:=t2]
Reductions:
  Beta =
    forall x.b : tm, A : ty, a : tm, z.T : ty
       |--- |- app(lam(A, x.b), a, z.T) => b[x:=a] : T[z:=a]
```

2.1. Ограничения на спецификации, налагаемые языком

1. Все используемые метапеременные должны иметь аннотацию (сорт), то есть присутствовать в секции forall аксиомы/редукции.

2. Запрещено равенство в заключении аксиом, для определенности каждого шага в проверке типов определяемого языка (если видим равенство не ясно в какую сторону идти при редуцировании)

3. Все аргументы в функциональный символ в заключении аксиомы должны быть метапеременными. Ещё и с теми же аргуементами что и в forall (не больше).

4. Если в заключении аксиомы написан функциональный символ возвращающий сорт, он обязан также иметь тип (нельзя просто написать $\vdash f(\ldots)def$).

5. Определения функциональных символов всегда одно, иначе появляется недетерминированность в проверке типов. Не играет особой роли, тк в данном случае можно сделать недетерминированность в проверке.

6. Подстановки разрешены только в метапеременные - в принципе это слабое ограничение, которое облегчает жизнь при реализации, не ограничивая пользователя.

7. В заключении контекст не должен быть расширен - это ограничение связано с тем, что иначе смысл аксиомы становится странным. А именно: функциональный символ применим только при введении перепенных в контекст.

8. Все метапеременные используемые в предпосылках должны либо присутствовать в метапеременных заключения или же должны быть типами какой-либо предпосылки.

9. Если в функциональном символе встречаются метапеременные с контекстами $x_1 \dots x_k.T$, должна существовать предпосылка вида $x_1 : S_1 \dots x_k : S_k \vdash T$. Это сделано для того чтобы не передавать типы контекстов метапеременных функционального символа явно.

10. Если метапеременная является типом предпосылки и не встречается в аргументах функционального символа, то она может использоваться только справа от двоеточия. Таким образом избегаются ситуации связанные с порядком проверки предпосылок языка. А именно: если у нас есть $x : S \vdash t : T, x : T \vdash r : S$. То нужно строить граф зависимостей для предпосылок и использовать порядок полученный в результате его топологической сортировки в генерации кода. (Аналогично с 3.5).

11. Все переменные контекстов метапеременных могут использовать только метапеременные левее внутри функционального символа в заключении - это связано с тем, что иначе могут возникнуть циклы в определениях метапеременных: S тип с аргументом типа R, R тип с аргументом типа S, S тип с аргументом типа R...

12. Из-за ослабления условия на метапеременные в пункте 8, порядок метапеременных неочевиден. Решение данной проблемы описано в секции 3.5.

13. Редукции не учитывают предпосылок при приведении в нормальную форму - предполагается что они не конфликтуют с аксиомами и проверки в аксиомах достаточно.

14. В редукциях все метапеременные справа от '=>' должны встречаться и слева от него.

15. Подстановка запрещена слева от '=>'.

2.2. Проверки корректности спецификации языка

Тривиальными проверками, осуществляемыми после парсинга языка, являются: проверка того, что сорты используемых выражений совпадают с аргументами функциональных символов.

3. Реализация

3.1. Парсер генераторы

3.2. Индексы де Брейна и их проблемы(задачки с индексами)

3.3. Проверка типов

3.4. сама генерация кода - просто описать exts + структуру

3.5.

Заключение

Список литературы

[1] Isaev Valery. Algebraic Presentations of Dependent Type Theories. — arxiv : math.LO, cs.LO, math.CT/http://arxiv.org/abs/1602.08504v3.

[2] Palmgren E., Vickers S.J. Partial Horn logic and cartesian categories // Annals of Pure and Applied Logic. — 2007. — Vol. 145, no. 3. — P. 314 – 353. — Access mode: http://www.sciencedirect.com/science/article/pii/S0168007206001229.

[3] Pierce Benjamin C. Advanced Topics in Types and Programming Languages. — The MIT Press, 2004. — ISBN: 0262162288.