Лекция 11

Ilya Yaroshevskiy

23 ноября

Содержание

1 Язык

Примечание. Мы находимся где-то в λC

```
\func id (x : \Type) : \Type = x
```

К какой части λ-куба относится `id: * -> *`

```
\func idid => id id --- нельзя
```

Но такое мы можем написать:

```
\func id2 (x : \Type) : (x -> x) => \lam a => a
\func idid2 => id2 (\Type -> \Type) (id2 \Type)
```

Получается что `\Type -> \Type : \Type`

```
\func Church => (x : \Type)  -> (x -> x) -> (x -> x)
\func add (m n : Church) => m Church inc n
```

Здесь ожидаем, что `inc : Church -> Church`

Примечание. Как это фиксить: `\Type : \Type` невозможно (парадокс Жирара)

1.1 Предикативность

Определение.

- `\Type n`
- `\Type 0` — базовые типы
- `\Type 1` — все, включая `\Type 0`
- `\Type (k + 1)` — все, включая `\Type k`

Пример. Черч на типах

```
\func Church => \Pi (x : \Type) -> (x -> x) -> (x -> x)
\func inc (n : Church) => \lam t f x => n t f (f x)
\func add (m : Church \levels (\suc\lp)\lh)
          (n : Church \levels \lp \lh) : Church \levels \lp \lh
          => m Church inc m
```

В `add` нужно применить `Church` к другим `Church`. `\lp` — внутренняя перменная, которая хранит текущий уровень предикативности. Мы говорим что первый аргумент имеет уровень предикативности на один больше, второй аргуент и результат имеют один уровень предикативности. Тогда `m Church` будет иметь тип `(Church -> Church) -> (C -> C)`, `Church` который внутри имеет уровень предикативности на один меньше

Примечание. `\Prop` — вселенная пропозиций “чистых утверждений“

Определение. `X : \Type` — `\Prop`, если все элементы x равны

Пример. Доказательство `Nat` — prop

Пример. `a : \Prop`, `b : \Prop`, то `(a, b) : \Prop`

Пример. `Either a b` — не prop, `inLeft a` $\neq$ `inRight b`

Пример. $\exists x.\varphi(x)$ — не prop

Определение. `\Set` — тип, в котором равентсво — `\Prop`

Примечание. Гомотопический уровень типа — $+1$ от уровня равенств на нем

Определение. Импредикативность — нет различий по пропозициональным уровням

Примечание. Все `\Prop` импредикативны

Определение. Пропозициональное обрезание $\|x\|$ `: \Prop`

- $\|$ `Either a b` $\|$ `=> a || b`
- $\|$ `\Sigma (x : N) (T(x))` $\|$ — $\exists x^N\ T(x)$

В аренде это `TruncP : \Type -> \Prop`, `inP`

Можем объявлять равенство между некоторыми вещами

```
\data Int'
  | pos' Nat
  | neg' Nat
```

Проблема: есть два 0. Можно сделать так

```
\data Int
  | pos Nat
  | neg (n : Nat) \elim n {
    | 0 => pos 0
  }
```

Примечание. Можем писать, говоря что все элементы дататайпа равны между собой

```
\truncated \data Quotient
  (A : \Type) (R : A -> A -> \Type) : \Set
  | inR A
  | eq (a a' : A) (r : R a a') (i : I)
    \elim i {
      | left => inR a
      | right => inR a'
    }
```

Положили `A` в коробочку и рядом положили равенство

Примечание.

- `Set` деклалирует единственность равенства
- Prop деклалирует единственность элементов