软件科学基础

第一次习题课

2024/3/28

Church Encoding

0: O

1: $S\ O$

2: $S\ (S\ O)$

3: $S\ (S\ (S\ O))$

…

Church Encoding

$$0: \lambda f.\, O$$

$$1: \lambda f.\, f\ O$$

$$2: \lambda f.\, f\ (f\ O)$$

$$3: \lambda f.\, f\ (f\ (f\ O))$$

$$\ldots$$

Church Encoding

$0: \lambda f.\ \lambda x.\ x$

$1: \lambda f.\ \lambda x.\ f\ x$

$2: \lambda f.\ \lambda x.\ f\ (f\ x)$

$3: \lambda f.\ \lambda x.\ f\ (f\ (f\ x))$

$\ldots$

Church Encoding

$$(0\ f\ x) = f^0(x)$$
$$(1\ f\ x) = f^1(x)$$
$$(2\ f\ x) = f^2(x)$$
$$(3\ f\ x) = f^3(x)$$

...

Church Encoding

```
Definition cnat := forall X : Type, (X -> X) -> X -> X.

Definition zero : cnat :=
  fun (X : Type) (f : X -> X) (x : X) => x.
Definition one : cnat :=
  fun (X : Type) (f : X -> X) (x : X) => f x.
Definition two : cnat :=
  fun (X : Type) (f : X -> X) (x : X) => f (f x).

...
```

Church Encoding

后继

$$f^{n+1}(x) = f(f^n(x)) = f(\mathfrak{n}\ f\ x)$$

```
Definition succ (n : cnat) : cnat :=
  fun (X : Type) (f : X -> X) (x : X) => f (n X f x).
```

Church Encoding

加法 $n + m$

$$f^{n+m}(x) = f^m f^n(x) = f^m(\mathfrak{n}\ f\ x) = \mathfrak{m}\ f\ (\mathfrak{n}\ f\ x)$$

```
Definition plus (n m : cnat) : cnat :=
  fun (X : Type) (f : X -> X) (x : X) => m X f (n X f x).
```

Church Encoding

乘法 $n \times m$

$$f^{n\times m}(x) = (f^n)^m(x) = (\mathfrak{n}\ f)^m(x) = \mathfrak{m}\ (\mathfrak{n}\ f)\ x$$

```
Definition mult (n m : cnat) : cnat :=
  fun (X : Type) (f : X -> X) (x : X) => m X (n X f) x.
```

Church Encoding

乘方 n^m

$$(\mathfrak{m}\ f\ \_) = f^m(\_) \longrightarrow (\mathfrak{m}\ \mathfrak{n}\ \_) = n^m(\_)$$

```
Definition exp (n m : cnat) : cnat :=
  := fun (X : Type) => m (X -> X) (n X).
```

Church Encoding

布尔值

True: $\lambda x.\ \lambda y.\ x$

False: $\lambda x.\ \lambda y.\ y$

```
Definition true  (X : Type) := fun (x y : X) => x.
Definition false (X : Type) := fun (x y : X) => y.
```

Church Encoding

```
true then-expr else-expr = then-expr
false then-expr else-expr = else-expr
```

$$\begin{aligned}
\text{and} &= \lambda p.\, \lambda q.\, p\; q\; p \\
\text{or} &= \lambda p.\, \lambda q.\, p\; p\; q \\
\text{not}_1 &= \lambda p.\, \lambda a.\, \lambda b.\, p\; b\; a \\
\text{not}_2 &= \lambda p.\, p\; (\lambda a.\, \lambda b.\, b)\; (\lambda a.\, \lambda b.\, a) = \lambda p.\, p\, \text{false}\, \text{true} \\
\text{xor} &= \lambda a.\, \lambda b.\, a\; (\text{not}\;\; b)\; b \\
\text{if} &= \lambda p.\, \lambda a.\, \lambda b.\, p\; a\; b
\end{aligned}$$

Church Encoding

- 减法
- 序对 pair: $\lambda x.\ \lambda y.\ \lambda z.\ z\ x\ y$
- 分数 $q = \frac{k}{1+a}$
- 实数 $|x-q| < 2^{-k}, k \in \mathbb{N}$

$$
\begin{aligned}
\text{pair} &\equiv \lambda x.\, \lambda y.\, \lambda z.\, z\ x\ y \\
\text{first} &\equiv \lambda p.\, p\ (\lambda x.\, \lambda y.\, x) \\
\text{second} &\equiv \lambda p.\, p\ (\lambda x.\, \lambda y.\, y)
\end{aligned}
$$

$$
\begin{aligned}
&\text{first}\ \ (\text{pair}\ \ a\ b) \\
=&(\lambda p.\, p\ (\lambda x.\, \lambda y.\, x))\ ((\lambda x.\, \lambda y.\, \lambda z.\, z\ x\ y)\ a\ b) \\
=&(\lambda p.\, p\ (\lambda x.\, \lambda y.\, x))\ (\lambda z.\, z\ a\ b) \\
=&(\lambda z.\, z\ a\ b)\ (\lambda x.\, \lambda y.\, x) \\
=&(\lambda x.\, \lambda y.\, x)\ a\ b = a
\end{aligned}
$$

rev_injective

```
Theorem rev_injective : forall (l1 l2 : natlist),
    rev l1 = rev l2 -> l1 = l2.
```

```
Proof.
  intros l1 l2 H.
  replace (l1) with (rev (rev l1)).
  replace (l2) with (rev (rev l2)).
  rewrite H. reflexivity.
  - rewrite rev_involutive. reflexivity.
  - rewrite rev_involutive. reflexivity.
Qed.
```

其它问题?