λ-calculus

Other Data Structures

Booleans

$$\left.\begin{aligned} true &:= \lambda x.\lambda y.\, x \\ false &:= \lambda x.\lambda y.\, y \end{aligned}\right\}\text{same as projections}$$

The if-then-else operation can be implemented as

$$if := \lambda b.\lambda u.\lambda v.\, b\, u\, v$$

Verify that

$$if\ true\ t_1\ t_2 \leadsto^* t_1$$

$$if\ false\ t_1\ t_2 \leadsto^* t_2$$

Boolean operations can be defined in terms of if

For example

$$or := \lambda a.\lambda b.\, if\ a\ true\ b$$

Pairs

If t_1 and t_2 are λ-terms we can represent the pair of them by

$$\langle t_1, t_2\rangle := \lambda x.\, x\, t_1\, t_2$$

Projections are just applications to Booleans:

$$\text{first} := \lambda p.\, p\ \text{true}$$

$$\text{second} := \lambda p.\, p\ \text{false}$$

Check that they work:

$$\begin{aligned}
&\text{first}\ \langle t_1, t_2\rangle \\
&= (\lambda p.p\ \text{true})\langle t_1, t_2\rangle \\
&\rightsquigarrow \langle t_1, t_2\rangle\ \text{true} \\
&= (\lambda x.x\, t_1 t_2)(\lambda x.\lambda y.x) \\
&\rightsquigarrow (\lambda x.\lambda y.x)\, t_1\, t_2 \\
&\rightsquigarrow^* t_1
\end{aligned}$$

Similarly

$$\text{second}\ \langle t_1, t_2\rangle \rightsquigarrow^* t_2$$

Vectors can be repeated pairs:

$$\langle t_1, t_2, t_3\rangle := \langle t_1, \langle t_2, t_3\rangle\rangle$$

or directly defined similarly:

$$\langle t_1, t_2, t_3\rangle := \lambda x.x\, t_1 t_2 t_3$$

Lists

We may think of defining also lists as repeated pairs:

$$[t_1, t_2, \dots, t_n] := \langle t_1, \langle t_2, \dots\rangle\rangle$$

But how about the empty list?

We could give it a conventional value: $\text{nil} := \text{false}$

What about the test: isnil ?

A more convenient way in the spirit of Church numerals:

$$[t_1, t_2, t_3] := \lambda f.\lambda x. f\, t_1 (f\, t_2 (f\, t_3\, x))$$

In general:

$$\mathrm{nil} := \lambda f.\lambda x.x$$

$$\mathrm{cons} := \lambda h.\lambda t. \lambda f.\lambda x. f\, h\, (t\, f\, x)$$

Exercise:

Check what

$$\mathrm{cons}\ t_1\ (\mathrm{cons}\ t_2\ \mathrm{nil})$$

reduces to

The empty list test:

$$\mathrm{isnil} := \lambda e.\, e\, (\lambda y.\lambda z.\mathrm{false})\ \mathrm{true}$$

Verify that it is correct:

$$\mathrm{isnil}\ \mathrm{nil} \rightsquigarrow^* \mathrm{true}$$

$$\mathrm{isnil}\ (\mathrm{cons}\ h\ t) \rightsquigarrow^* \mathrm{false}$$

Exercise:

Write head and tail functions extracting the first element and the rest of a non-empty list.

(Doesn't matter what it does on nil.)

Exercise:

- Define a function **sum** that adds up all elements of a list of numerals.
- Define a function **mirror** that reverses a list.

Can we do infinite lists?

An infinite list (or **stream**) is a neverending sequence:

$a_0 \triangleleft a_1 \triangleleft a_2 \triangleleft \cdots$

Idea: represent it as

$$\lambda f. f a_0 (f a_1 (f a_2 \cdots))$$

But it can't actually be infinitely long.

The stream $0 \triangleleft 1 \triangleleft 2 \triangleleft \cdots$ should be a term t_0 such that:

$$\begin{aligned} t_0 &\rightsquigarrow^* \lambda f. f\, 0\, t_1 \\ &\rightsquigarrow^* \lambda f. f\, 0\, (f\, 1\, t_2) \\ &\rightsquigarrow^* \lambda f. f\, 0\, (f\, 1\, (f\, 2\, t_3)) \cdots \end{aligned}$$