

ldf.1 Pairs and Predecessor

lam:ldf:pai:
sec

Definition ldf.1. The pair of M and N (written $\langle M, N \rangle$) is defined as follows:

$$\langle M, N \rangle \equiv \lambda f. fMN.$$

Intuitively it is a function that accepts a function, and applies that function to the two elements of the pair. Following this idea we have this constructor, which takes two terms and returns the pair containing them:

$$\text{Pair} \equiv \lambda mn. \lambda f. fmn$$

Given a pair, we also want to recover its elements. For this we need two access functions, which accept a pair as argument and return the first or second elements in it:

$$\text{Fst} \equiv \lambda p. p(\lambda mn. m)$$

$$\text{Snd} \equiv \lambda p. p(\lambda mn. n)$$

Problem ldf.1. Explain why the access functions `Fst` and `Snd` work.

Now with pairs we can λ -define the predecessor function:

$$\text{Pred} \equiv \lambda n. \text{Fst}(n(\lambda p. \langle \text{Snd } p, \text{Succ}(\text{Snd } p) \rangle)(\bar{0}, \bar{0}))$$

Remember that $\bar{n} f x$ reduces to $f^n(x)$; in this case f is a function that accepts a pair p and returns a new pair containing the second component of p and the successor of the second component; x is the pair $\langle 0, 0 \rangle$. Thus, the result is $\langle 0, 0 \rangle$ for $n = 0$, and $\langle \bar{n} - 1, \bar{n} \rangle$ otherwise. `Pred` then returns the first component of the result.

Subtraction can be defined as `Pred` applied to a, b times:

$$\text{Sub} \equiv \lambda ab. b\text{Pred } a.$$

Photo Credits

Bibliography