lam.1df.pai: sec

**Definition ldf.1.** The pair of $M$ and $N$ (written $⟨M, N⟩$) is defined as follows:

$$⟨M, N⟩ ≡ \lambda f. fMN.$$

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:

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

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:

$$\text{Fst} ≡ \lambda p. p (\lambda mn.m)$$
$$\text{Snd} ≡ \lambda p. p (\lambda mn.n)$$

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

Now with pairs we can \textit{\lambda-define} the predecessor function:

$$\text{Pred} ≡ \lambda n. \text{Fst}(n(\lambda p. (\text{Snd} p, \text{Succ}(\text{Snd} p))))(⟨0,0⟩)$$

Remember that $\pi fx$ 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 $⟨0,0⟩$. Thus, the result is $⟨0,0⟩$ for $n = 0$, and $⟨n−1, n⟩$ otherwise. Pred then returns the first component of the result.

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

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

**Photo Credits**

**Bibliography**