Pairs

Toby Murray
toby.murray@nicta.com.au

July 28, 2010

Contents

```
theory Pair
imports Main
begin
```

A pair of elements, `a` and `b`, is a function that takes a function `f` of type `'a ⇒ 'b ⇒ 'c` and applies `f` to `a` and `b`, giving a result of type `'c`.

That is, generally, a pair is of type `('a ⇒ 'b ⇒ 'c) ⇒ 'c`.

```
types ('a,'b,'c) pair = "('a ⇒ 'b ⇒ 'c) ⇒ 'c"
```

Let us abbreviate this by `('a, 'b, 'c) pair`.

`make_pair` takes elements `a` and `b` and gives us a pair. Note that the type `'c` in the result type is unconstrained and can be anything.

```
definition make_pair :: "'a ⇒ 'b ⇒ ('a,'b,'c) pair"
where
"make_pair ≡ λ a b. λ f. f a b"
```

The function `fst` takes a pair `p` and gives its first element. Note that it calls `p` with an argument function (that given the pair's two elements, returns the first one) of type `'a ⇒ 'b ⇒ 'a`. Therefore, `p` must be of type `('a, 'b, 'a) pair`.

```
definition fst :: "('a,'b,'a) pair ⇒ 'a"
where
"fst p ≡ p (λ a b. a)"
```

The function `snd` is naturally similar to `Pair.fst` but since the argument that it applies the pair to is of type `'a ⇒ 'b ⇒ 'b`, the pair it is given must be of type `('a, 'b, 'b) pair`.

```
definition snd :: "('a,'b,'b) pair ⇒ 'b"
where
"snd p ≡ p (λ a b. b)"
```

```
lemma "fst (make_pair a b) = a"
  apply(unfold make_pair_def fst_def)
  apply(rule refl)
  done

lemma "snd (make_pair a b) = b"
  apply(unfold make_pair_def snd_def)
  apply(rule refl)
  done

end
```