A pair of elements, a and b, is a function that takes a function f of type
\(\text{'}a \Rightarrow \text{'}b \Rightarrow \text{'}c\) and applies f to a and b, giving a result of type \(\text{'}c\).

That is, generally, a pair is of type \(\text{'}a \Rightarrow \text{'}b \Rightarrow \text{'}c\).

Let us abbreviate this by \((\text{'}a, \text{'}b, \text{'}c)\) pair.

\textbf{make_pair} takes elements a and b and gives us a pair. Note that the type \(\text{'}c\) in the result type is unconstrained and can be anything.

\textbf{definition} \text{make_pair} :: \(\text{'}a \Rightarrow \text{'}b \Rightarrow (\text{'}a, \text{'}b, \text{'}c)\) pair

where
"\text{make_pair} \equiv \lambda a b. \lambda f. f a b"

The function \text{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 \(\text{'}a \Rightarrow \text{'}b \Rightarrow \text{'}a\). Therefore, p must be of type \((\text{'}a, \text{'}b, \text{'}a)\) pair.

\textbf{definition} \text{fst} :: \(\text{'}a \Rightarrow \text{'}b \Rightarrow ('a,'b,'a) pair \Rightarrow 'a\)

where
"\text{fst} p \equiv p (\lambda a b. a)"

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

\textbf{definition} \text{snd} :: \(\text{'}a \Rightarrow \text{'}b \Rightarrow ('a,'b,'b) pair \Rightarrow 'b\)

where
"\text{snd} p \equiv p (\lambda 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