1 Basic inductive definitions

1.1 Booleans

We define

\[
\text{Inductive bool : Type := true : bool | false : bool.}
\]

1- Define boolean negation \texttt{negb} and boolean conjunction \texttt{andb} in CCI.

2- Detail the normalisation steps of expressions \( \lambda x : \text{bool}. \texttt{negb (andb false x)} \) et \( \lambda x : \text{bool}. \texttt{negb (andb x false)} \) (in Coq, one can use the command \texttt{Eval compute in}). What is remarkable ?

1.2 Logical connectives

Observe how the logical connectives \texttt{and}, \texttt{or}, \texttt{ex} and their induction schemes are defined in the standard library of Coq, using the command \texttt{Print ident}.

2 Recursive types

A- Propose in Coq an inductive definition with parameter corresponding to the ML type of polymorphic lists:

\[
\text{type 'a list = nil | cons of 'a * 'a list}
\]

B- Coq library defines the binary product, the unit type and the type of natural numbers:

\[
\begin{align*}
\text{Inductive prod (A B : Type) : Type := pair : A \rightarrow B \rightarrow prod A B.} \\
\text{Inductive unit : Type := tt : unit.} \\
\text{Inductive nat : Type := O : nat | S : nat \rightarrow nat.}
\end{align*}
\]

Construct an expression \texttt{prodn} in CCI of type \( \text{Type} \rightarrow \text{nat} \rightarrow \text{Type} \) which builds the n-ary product of a given type \( A \): (i.e. \( \texttt{prodn A n} \) is \( A \times \ldots \times A \) (\( n \) times)). The definition will be by recursion on \( n \).

Give an expression \texttt{length} of type \( \forall A. \text{list A} \rightarrow \text{nat} \) which computes the length of a list.

Give an expression \texttt{embed} of type \( \forall A. \forall l : \text{list A}. \texttt{prodn A (length l)} \) which translates a list into a \( n \)-uple.