Proof assistants

TD 5- Functional Programming

1 Dependently typed programming

1. Develop an insertion sort program (functions `insert` and `sort`) using types including his specification. Download the file `http://www.lix.polytechnique.fr/~barras/mpri/insertion.v` which contains definitions and lemmas about orders and sorted lists.

2 Indexed types

The goal is to define the type of sorted lists directly, i.e. without introducing the type of unsorted lists first.
The specification of `cons` takes as argument a sorted list l and an element a. It should also check that a is smaller than all the elements of l. To avoid circularity, we first define a type of sorted lists indexed by an element x; intuitively the elements of this type are sorted lists which least element is x. Since the empty list has no element, the list is indexed by an inhabitant of `option` A. The empty list is indexed by `None` and list $a :: l$ by `Some` a.

1. Define inductively the type of sorted lists indexed by their least element.
2. Define the insertion function on those sorted lists.
3. Then, the type `set` A is defined as the pair formed of an index x and a sorted list of index x.
   (a) Define the type `set` as above.
   (b) Define `in_set` the membership predicate.
   (c) Define the object corresponding to the empty set and the function adding an element to a set.
4. We can see that the index appears in the terms extracted from these functions. One solution to this issue is to index the type of lists by a predicate $P : A \to \mathbf{Prop}$ such that $P\,x$ iff x is the least element of the list.
   (a) Define this new type.
   (b) Define the functions of the previous section and compare the extracted terms.