`http://www.labri.fr/~sutre/Teaching/B`

This project is devoted to the specification and implementation of a utility module for collections, similar to what can be found in standard libraries (stacks and queues).

1 Project Requirements

The development shall be realized, and proved, within Atelier B. All components must pass type-checking, as well as B0 checking. It is not required to prove all proof obligations. However, the remaining unproved obligations must be analyzed and checked for validity. Recall that you may add assertions in your components to simplify the proof process. It is also recommended to animate your components with ProB in order to spot obvious bugs before attempting proofs.

Projects must be sent to `gregoire.sutre@labri.fr` **by Friday, 9 December 2011**. Send an archive of your Atelier B project (including the `bdp` sub-directory), along with a small report presenting your B development. The report should, in particular, provide an informal proof sketch for each unproved obligation.

2 Specification of a Collection

The specification models the collection as a *multiset* of elements. Those elements are taken in a deferred set `UNIVERSE`, which is given as parameter of the specification. The multiset is specified by a total function $bag \in \texttt{UNIVERSE} \to \mathbb{N}$, that maps each element $e \in \texttt{UNIVERSE}$ to the number of copies of e in the bag.

A template of the `Collection` abstract machine is provided in the `Collection.mch` file. In addition to the *bag* variable, this template also specifies a *bagsize* variable that counts the number of elements in the multiset *bag*.

Exercise 1 *Fill the parts marked "To be completed" in the abstract machine* `Collection`*. This abstract machine should be entirely proved by the automatic prover in force 0.*

3 Intermediate Refinement

The first refinement of the `Collection` abstract machine is an intermediate refinement to factorize and simplify the development of the `Stack` and `Queue` refinements. A template for this first refinement is provided in the `Collection_r.ref` file. Here, the multiset is replaced by a partial function $list \in \mathbb{N} \nrightarrow \texttt{UNIVERSE}$, that, intuitively, distinguishes copies by numbering each element in the multiset.

Example. Assuming that the set `UNIVERSE` contains $\{a, b\}$, the multiset $\{a \mapsto 2, b \mapsto 3\}$ may be represented, for instance, by the partial function $\{0 \mapsto a, 5 \mapsto b, 7 \mapsto b, 16 \mapsto a, 25 \mapsto b\}$.

Exercise 2 *Fill the parts marked "To be completed" in the refinement* `Collection_r`*. Prove as many proof obligations as possible.*

4 Stack Refinement

The intermediate refinement `Collection_r` is now refined into a stack: the operation `pop` becomes deterministic, and returns the element in the collection that was `push`ed last. A template of the `Stack` refinement is provided in the `Stack.ref` file. Here, the stack is specified as a total function $sequence \in 1..length \rightarrow \texttt{UNIVERSE}$, where $length$ is the size of the stack.

Exercise 3 *Fill the parts marked "To be completed" in the refinement* `Stack`*. Prove all proof obligations.*

5 Queue Refinement

The intermediate refinement `Collection_r` is now refined into a queue: the operation `pop` becomes deterministic, and returns the element in the collection that was `push`ed first. A template of the `Queue` refinement is provided in the `Queue.ref` file. Here, the queue is specified as a total function $window \in start..(end-1) \rightarrow \texttt{UNIVERSE}$, that models the queue as a sliding window.

Exercise 4 *Fill the parts marked "To be completed" in the refinement* `Queue`*. Prove all proof obligations.*