Master, University of Bordeaux 1
Formal Design (Code: `J1IN7123`)
Grégoire Sutre

Date:	14 December 2011
Duration:	1 hour 30 (B part)
With documents:	Yes
Subject length:	4 pages

Warning

This written examination shall be realized without the help of Atelier B. Still, your answers must conform to the following requirements:

- Write syntactically correct B code. You may use, at your convenience, the B mathematical or the ASCII notation, but avoid mixing these two notations.
- Machines, refinements and implementations must be correct, i.e., the associated correctness proof obligations must hold.

1 Specification and implementation of a bit-width operation

Let us define the *bit-width* of a natural number $x \in \mathbb{N}$ as the least $n \in \mathbb{N}$ such that $x < 2^n$. This section is devoted to the completion and implementation of the abstract machine `BitWidth` given below. The `width` operation shall return the bit-width of its argument.

```
MACHINE
    BitWidth
OPERATIONS
    res <-- width (xx) =
        PRE
            xx : NAT
        THEN
            >>>  To Be Completed (Exercise 1)  <<<
        END
END
```

Exercise 1 *Fill the part marked* `To be completed` *in the abstract machine* `BitWidth`.

Exercise 2 *Write a B implementation for the machine* `BitWidth`.

2 Specification and refinement of an elevator

This section focuses on the specification and refinement of an elevator. The set of floors served by the elevator is modeled by a constant interval of integers. There is a single cab (also known as cage) in this specification. The cab is equipped with a door, that may be either closed or open. For simplicity, the specification does not account for floor doors.

2.1 Specification of the elevator

The elevator is specified in the B abstract machine `Elevator` provided below. The meaning of the *cab* and *door* concrete variables should be self-explanatory. The *req* variable contains the pending requests as a subset of $floors$. Requests may originate either from inside the cab or from a floor: the machine does not care (i.e., it processes these two kinds of requests

identically). There is initially no pending request. The machine contains three operations: an operation to `move` the cab to another floor, an operation to `toggle` the state of the doors, and an operation to add a new `request`. A request is considered served (and, hence, removed from *req*) when closing the cab door at the corresponding floor.

```
MACHINE
    Elevator
SETS
    /* Possible states of the cab door. */
    DSTATE = {closed, open}
CONSTANTS
    /* Floors range from flmin to flmax. */
    flmin,
    flmax,
    floors
PROPERTIES
    flmin : INT &
    flmax : INT &
    flmin < flmax &
    floors = (flmin..flmax)
CONCRETE_VARIABLES
    /* Cab's current floor. */
    cab,
    /* State of the cab door. */
    door
ABSTRACT_VARIABLES
    /* Pending requests. */
    req
INVARIANT
    cab : floors &
    req <: floors &
    door : DSTATE
    &
    >>>  To Be Completed (Exercise 3)  <<<
INITIALISATION
    cab := flmin ||
    req := {} ||
    door := closed
OPERATIONS
    request (fl) =
        PRE
            >>>  To Be Completed (Exercise 4)  <<<
        END
    ;

    move =
        PRE
            >>>  To Be Completed (Exercise 5)  <<<
        END
    ;

    toggle =
        PRE
            >>>  To Be Completed (Exercise 6)  <<<
        END
END
```

Exercise 3 *Specify in the* `INVARIANT` *clause of the machine* `Elevator` *the following property: if the door is open then the current floor is among the pending requests.*

The operations of the machine `Elevator` are, informally, specified as follows. The use of the *must* modality indicates a property that shall be specified as a precondition of the operation.

- `request`(fl) adds the floor fl to the set (not a multiset) of pending requests. It is allowed for the given floor fl to already be a pending request.
- `move` moves the cab to an arbitrary floor that is among the pending requests. The current floor must not be a pending request, and the door must be closed.
- `toggle` switches the state of the cab door: opens it if it is closed, and closes it otherwise. The cab door must be toggled only if the cab's current floor is a pending request. If the operation closes the door, the request is removed from the set of pending requests.

Exercise 4 *Complete the specification of the* `request` *operation.*

Exercise 5 *Complete the specification of the* `move` *operation.*

Exercise 6 *Complete the specification of the* `toggle` *operation.*

2.2 Refinement of the elevator

Requests in the abstract machine `Elevator` may be served in any order. Thus, the machine permits behaviors where a pending request is never served. To avoid such unfair behaviors, the following refinement assigns a time stamp to each request. The set req of pending requests is therefore replaced by a partial injection $treq$ from $floors$ to $\mathbb{N}$.

```
REFINEMENT
    Elevator_r
REFINES
    Elevator
ABSTRACT_VARIABLES
    /* Pending requests, with timestamps. */
    treq
INVARIANT
    treq : floors >+> NATURAL
    &
    >>>  To Be Completed (Exercise 7)  <<<
INITIALISATION
    cab := flmin ||
    treq := {} ||
    door := closed
OPERATIONS
    >>>  To Be Completed (Exercises 8, 9, 10)  <<<
END
```

Exercise 7 *Specify in the* `INVARIANT` *clause of the refinement* `Elevator_r` *the gluing invariant relating* req *and* $treq$.

Exercise 8 *Write the refinement of the* `toggle` *operation.*

In this refinement, to ensure fairness, pending requests shall be served with a first-in-first-out (FIFO) strategy:

- The `move` operation moves the cab to the floor that has the least time stamp.
- In the `request`(fl) operation, if the given floor fl is already a pending request, then nothing is done ; otherwise, fl is added to *treq* with an arbitrary time stamp that is strictly larger than the time stamp of all other pending requests.

Exercise 9 *Write the refinement of the* `move` *operation.*

Exercise 10 *Write the refinement of the* `request` *operation.*

Serving requests with a FIFO strategy is sub-optimal in some cases. For instance, with this strategy, the elevator may move directly from the floor $flmin$ to the floor $flmax$ even though there is a pending request for another floor $flmin < fl < flmax$. To reduce movements of the cab, we consider the following *weak* FIFO strategy:

- The `move` operation moves the cab to an arbitrary floor fl such that
  1. fl is a pending request, and
  2. fl is located between the current floor and the floor that has the least time stamp.

Exercise 11 *Modify your refinement of the* `move` *operation to conform with the weak FIFO strategy presented above.*