CS412 Exercise sheet 6: solutions

Structuring

1. (a) As the Door machine only SEES the Light machine it is not allowed to include invariant requirements which make reference to Light's variable, switch. Door does not have control over the machine which it SEES and there is nothing to prevent the light being switched off by some other machine which includes Light. If this happened, Door's invariant could become false.

   (b) This would produce a consistent machine, but it wouldn't be the machine required because the light could be switched off but the door open.

   (c) Could use USES which would propogate a proof obligation upwards to be considered later in the development. Or, if nothing else needed to control the light, it could be INCLUDED (or EXTENDED) here.

   (d) It doesn't matter which structuring mechanism is used because the invariant condition will be the same. Neither machine has constraints or properties so the proof obligation is:
   $[Init_{Light};\ InitDoor]Inv_{Door}$
   $\equiv [switch := off]([door := closed](door \in DSTATE \wedge$
   $door = open \Rightarrow switch = on))$
   $\equiv [switch := off](closed \in DSTATE \wedge (closed = open \Rightarrow switch = on))$
   $\equiv closed \in DSTATE \wedge (closed = open \Rightarrow off = on)$
   $\equiv True \wedge (False \Rightarrow False)$
   $\equiv True$

2.
```
MACHINE           Fifo(ELEM,cap)
CONSTRAINTS       cap : NAT1
VARIABLES         contents
INVARIANT         contents : seq(ELEM) & size(contents) <= cap
INITIALISATION    contents := <>
OPERATIONS
  input(ee) =
     PRE  ee : ELEM & size(contents) < cap
     THEN contents := contents <- ee
     END;
  ee <-- output =
     PRE  size(contents) > 0
     THEN ee := first(contents) || contents := tail(contents)
     END
END

MACHINE           Router
INCLUDES          Fifo(MSG, qmax)
SETS              MSG; DEST; STATUS = {yes,no}
```

```
CONSTANTS         qmax
PROPERTIES        qmax : NAT1
VARIABLES         pending, is_pending, nexthop
INVARIANT         nexthop : MSG --> DEST & pending : MSG & is_pending : STATUS
INITIALISATION    nexthop :: MSG --> DEST || pending :: MSG || is_pending := no
OPERATIONS
  receive(mm) =
    PRE  mm:MSG
    THEN IF   size(contents) < qmax
         THEN input(mm)
    END
    END;
  retreive =
    IF   size(contents) > 0 & is_pending = no
    THEN pending <-- output || is_pending := yes
    END;
  ndest,msg <-- forward =
    IF   is_pending = yes
    THEN msg := pending || ndest := nexthop(pending) || is_pending := no
    END
END
```

The rule for operation PRE P THEN S END (with Router as machine 2 and Fifo as machine 1, C for constraints, B for properties, I for invariant) is:

$$C_1 \wedge C_2 \wedge B_1 \wedge B_2 \wedge I_1 \wedge I_2 \wedge P \Rightarrow [S]I_2$$

In addition, as Fifo has its parameters supplied by the calling machine, we know that $ELEM = MSG$ and $cap = qmax$.

(a) For the receive operation hypotheses (everything to the left of the implication) are:

```
cap : NAT1 & qmax : NAT1 & contents : seq(MSG) & size(contents) <= cap &
nexthop : MSG --> DEST & pending : MSG & is_pending : STATUS & mm:MSG
```

The goal is:

$[IF\ size(contents) < qmax\ THEN\ input(mm)\ END]I_2$
$\equiv size(contents) < qmax \Rightarrow [input(mm)]I_2$

If we assume the LHS we can then try to prove the RHS.

$[input(mm)]I_2$
$\equiv$ $[PRE\ mm : ELEM \wedge size(contents) < cap$
$THEN\ contents := contents \leftarrow mm$
$END]I_2$
$\equiv$ $[PRE\ mm : MSG \wedge size(contents) < qmax$
$THEN\ contents := contents \leftarrow mm$
$END]I_2$ Using known instantiation of parameters.
$\equiv$ $mm : MSG \wedge size(contents) < qmax \wedge [contents := contents \leftarrow mm]I_2$ *

The first part of this comes directly from the hypothesis list. The second part is the assumption we added when dealing with the implication. Thus we are left with:

$[contents := contents \leftarrow mm]I_2$
$\equiv [contents := contents \leftarrow mm](nexthop : MSG \rightarrow DEST \wedge pending : MSG \wedge$
$is\_pending : STATUS)$

$\equiv nexthop : MSG \rightarrow DEST \wedge pending : MSG \wedge is\_pending : STATUS)$

This is simply I_2 and all conjuncts appear in the hypotheses.

(b) It would be calling the operation from the included machine outside its precondition. Looking at the answer for the previous part, if you haven't got the IF statement, then you wouldn't be adding the extra hypothesis of $size(contents) < qmax$ to the hypothesis list so you wouldn't later be able to discharge the 2nd conjunct from line *.

(c) For the retreive operation the hypothesis list is exactly the same except you don't have the final conjunct $mm : MSG$ because the operation has no precondition. Our goal is:

$$[IF\ size(contents) > 0 \wedge is\_pending = no$$
$$THEN\ pending \leftarrow output \mid\mid is\_pending := yes$$
$$END]I_2$$
$$\equiv (size(contents) > 0 \wedge is\_pending = no) \Rightarrow [pending \leftarrow output \mid\mid is\_pending := yes]I_2$$
$$\wedge$$
$$\neg\ (size(contents) > 0 \wedge is\_pending = no) \Rightarrow I_2$$

For the 2nd conjunct, I_2 comes from the hypotheses and so this conjunct is true.

For the 1st conjunct, if we again assume the LHS we're left with the goal:

$$[pending \leftarrow output \mid\mid is\_pending := yes]I_2$$
$$\equiv [(PRE\ size(contents) > 0$$
$$THEN\ pending := first(contents) \mid\mid contents := tail(contents)$$
$$END)\ \mid\mid is\_pending := yes]I_2$$
$$\equiv [(PRE\ size(contents) > 0$$
$$THEN\ pending := first(contents) \mid\mid contents := tail(contents) \mid\mid is\_pending := yes$$
$$END)]I_2$$
$$\equiv size(contents) > 0 \wedge I_2[first(contents)/pending, tail(contents)/contents, yes/is\_pending$$

The 1st conjunct comes from the hypothesis added when dealing with the implication. That just leaves:

$$I_2[first(contents)/pending, tail(contents)/contents, yes/is\_pending]$$
$$\equiv (nexthop : MSG \rightarrow DEST \wedge pending : MSG \wedge is\_pending : STATUS)[first(contents)/pending,$$
$$tail(contents)/contents,$$
$$yes/is\_pending]$$
$$\equiv nexthop : MSG \rightarrow DEST \wedge first(contents) : MSG \wedge yes : STATUS$$

The first conjunct is a hypothesis. The second conjunct follows since $contents : seq(MSG)$ is a hypothesis and the fact that it's nonempty (and therefore $first(contents)$ is meaningful) is an assumption made when dealing with the implication. The 3rd conjunct is trivially true since yes is one of the defined elements of $STATUS$.

3. Here is just one possible solution. The main feature from the structuring point of view is the inclusion of 2 separate copies of Fifo and the way that these (and their operations and variables) are prefixed. However, there are a few other points worth noting here. Firstly, the requirements are vague about how the decision is made concerning whether to discard a job or pass it on to the next queue. This spec represents it by a nondeterministic choice using the CHOICE operator. Secondly, in the operation to move a job from one queue to the next we have to be careful about how we obtain the relevant job from the first queue. This is because the syntax for calling an operation will only allow us to write the call in a place where an operation is expected NOT where an expression is expected. The spec gets round this by referring directly to $''first(q1.contents)''$ in an ANY clause and then using that as the element to add to q2. Note that in the THEN

section the use of *xx* is merely as a syntactic recepticle for the output from *q*1.*output*. Things get a lot easier in this respect when we move on to refinement and can use sequencing.

```
MACHINE           JobProcessing

INCLUDES          q1.Fifo(JOBID, qmax1), q2.Fifo(JOBID, qmax2)

SETS              JOBID

CONSTANTS         qmax1, qmax2

PROPERTIES        qmax1 : NAT1 & qmax2 : NAT1

OPERATIONS

/* Operation to receive a new job request onto te hfirst queue
*/
  getreq(jj) =
    PRE  jj:JOBID
    THEN IF   size(q1.contents) < qmax1
         THEN q1.input(jj)
    END
    END;

/* Operation to retrieve a queued job id and decide whether to discard it
   or move it to the next queue ready for procesing.
*/
  step1 =
    IF   size(q1.contents) > 0 &  size(q2.contents) < qmax2
    THEN ANY xx WHERE xx : JOBID & xx = first(q1.contents)
         THEN xx <-- q1.output ||
              CHOICE q2.input(xx) OR skip END
         END
    END;

/* Operation to remove job from 2nd quqe for processing.
*/
  jj <-- step2 =
    IF   size(q2.contents)  > 0
    THEN  jj <-- q2.output
    END

END
```