CS412 Solutions to exercise sheet 8

Verification of refinements and understanding loops

1. **The first version of** $AbSetR$ uses $num \in NAT$ as the concrete state. Consider the initialisation condition (noting that neither machine has constraints or properties).

   $$\begin{aligned}&[InitR]\neg\,[Init]\neg\,(num \in myset)\\ &= [num :\in \mathbb{N}]\neg\,[myset := \{\}]\neg\,(num \in myset)\\ &= [num :\in \mathbb{N}]\neg\,\neg\,(num \in \{\})\\ &= [num :\in \mathbb{N}](num \in \{\})\\ &= [num :\in \mathbb{N}]false\\ &= false\end{aligned}$$

   This isn't surprising since our linking invariant doesn't link any concrete state to the abstract state with $myset = \{\}$.

   There's nothing wrong with the general approach here - we just need to think about the linking invariant. If it's replaced with something which takes into acount the empty set then it will be ok: $myset = \{\} \vee num \in myset$

   The second version of $AbSetR$ is the one using $myseq \in \mathrm{iseq}(\mathbb{N})$ as the concrete state. Consider the applicability condition for set_add (again, no constraints or properties).

   $$\begin{aligned}&Inv \wedge InvR \wedge Pre \Rightarrow PreR\\ &= (myset \in \mathbb{P}(\mathbb{N}) \wedge myseq \in \mathrm{iseq}(\mathbb{N}) \wedge myset = \mathrm{ran}(myseq)) \Rightarrow nn \notin ran(myseq)\end{aligned}$$

   Can't prove this - there's nothing on the LHS that can help us. We've added a precondition in the refinement which decreases applicability. That isn't allowed.

   Could fix this in the refinement by using an IF instead of PRE since this guarantees to do nothing outside the guard condition.

2. For initialisation refinement between $Colours$ and $ColoursR$ we need to show that: $[InitR]\neg\,[InitA]\neg\,J$ where J is the linking invariant. In this case that's:

   $[colour :\in COLOUR - \{blue\}]\neg\,[cols :: \mathbb{P}(COLOUR - \{blue\})]\neg\,colour \in cols$

   Let's tackle the "inner" part of this first which, rewriting slightly for the known value of the type, is:

   $$\begin{aligned}&\neg\,[cols :\in \mathbb{P}(\{red, green\})]\neg\,colour \in cols\\ &= \neg\,(\forall xx \bullet (xx \in \mathbb{P}(\{red, green\}) \Rightarrow \neg\,(colour \in xx))\end{aligned}$$

$= \exists\, xx \bullet (xx \in \mathbb{P}(\{red, green\}) \wedge colour \in xx)$
$= colour \neq blue$

Then, using this result with the "outer" layer:
$[colour :\in COLOUR - \{blue\}]colour \neq blue$
is obviously true

The abstract initialisation and the linking invariant remain the same for the first 2 changes suggested in the question.

(a) $[InitR]\neg\, [Init]\neg\, (colour \in cols)$
$= [colour := red]colour \neq blue$
$= red \neq blue = true$

So this is ok.

(b) $[InitR]\neg\, [Init]\neg\, (colour \in cols)$
$= [colour :\in COLOUR]colour \neq blue$
$= \forall\, colour \bullet (colour \in COLOUR \Rightarrow colour \neq blue)$
$= false$

So this is no good.

(c) Here, abstract init changes, so must recalculate. $[InitR]\neg\, [Init]\neg\, (colour \in cols)$
$= [InitR]\neg\, [cols := \{\}]\neg\, (colour \in cols)$
$= [InitR]\neg\, (\neg\, colour \in \{\})$
$= [InitR]\neg\, (\neg\, false)$
$= [InitR]false$
$= [InitR]false$

So again, this is not a refinement.

(d) $[InitR]\neg\, [Init]\neg\, (colour \in cols)$
$= [InitR]\neg\, [cols := \{red\}]\neg\, (colour \in cols)$
$= [InitR]\neg\, \neg\, (colour \in \{red\})$
$= [InitR]colour = red$
$= [colour :\in COLOUR - \{blue\}]colour = red$
$= \forall\, colour \bullet (colour \in COLOUR - \{blue\} \Rightarrow colour = red)$
$= false$

So again, this is no good.

3. (a) **Initialisation** (no constraints or properties). There is no initialisation stated (no new variables) so can treat it as skip. Note that this is operation refinement without data refinement.
$[InitR]\neg\, [Init]\neg\, red \in cols$
$= [skip]\neg\, [cols :\in \mathbb{P}(COLOUR - \{blue\})]\neg\, red \in cols$
$= \neg\, [cols :\in \mathbb{P}(COLOUR - \{blue\})]\neg\, red \in cols$
$= \neg\, \forall\, cols \bullet (cols :\in \mathbb{P}(COLOUR - \{blue\}) \Rightarrow \neg\, red \in cols)$

$= \exists\, cols \bullet (cols :\in \mathbb{P}(COLOUR - \{blue\}) \wedge red \in cols)$
$= true$

Static requirement none of the sections are present in these machines, so nothing to prove.

Operations Neither of the refinement operations has a precondition, so there is nothing to prove for applicability. The correctness condition is:

$$Inv \wedge InvR \wedge Pre \Rightarrow [OpR[oo'/oo]]\neg\, [OpA]\neg\, (InvR \wedge oo' = oo)$$

add(cc)
$cols \subseteq COLOUR \wedge red \in cols \wedge cc \in COLOUR$
$\quad \Rightarrow [skip]\neg\, [cols := cols \cup \{cc\}]\neg\, (red \in cols)$
RHS $= \neg\,\neg\, red \in (cols \cup \{cc\})$
And this is implied by the LHS.

cc <– query
$cols \subseteq COLOUR \wedge red \in cols \wedge cols \neq \{\}$
$\quad \Rightarrow [cc := red[cc'/cc]]\neg\, [cc :\in cols]\neg\, (red \in cols \wedge cc' = cc)$
RHS $= [cc' := red]\neg\, \forall\, cc \bullet (cc \in cols \Rightarrow \neg\, (red \in cols \wedge cc' = cc))$
$= [cc' := red]\, \exists\, cc \bullet (cc \in cols \wedge red \in cols \wedge cc' = cc))$
$= \exists\, cc \bullet (cc \in cols \wedge red \in cols \wedge red = cc))$
From the LHS, know that $red \in cols$ and this implies the above.

change
$cols \subseteq COLOUR \wedge red \in cols \Rightarrow$
$\quad [skip]\neg\, [cols :\in (\mathbb{P}(COLOUR) - \{cols\})]\neg\, (red \in cols)$
RHS $= \neg\, \forall\, cols \bullet (cols :\in (\mathbb{P}(COLOUR) - \{cols\}) \Rightarrow \neg\, red \in cols)$
$= \exists\, cols \bullet (cols :\in (\mathbb{P}(COLOUR) - \{cols\}) \wedge red \in cols)$
$= true$

All the proof obligations are met, so this is a refinement.

(b) In this case, consider the query operation. The correctness requirement becomes:
$cols \subseteq COLOUR \wedge true \wedge cols \neq \{\}$
$\quad \Rightarrow [cc' := red]\neg\, [cc :\in cols]\neg\, (cc' = cc)$
RHS $= \exists\, cc \bullet cc \in cols \wedge red = cc$
$= red \in cols$
Not enough information on the LHS to be able to prove this. Therefore, this cannot be proved as a refineemnt. The problem is simply that the invariant doesn't give enough information for the correspondence between the two levels to be made.

4. (a) **Condition 1:** $I \wedge E \Rightarrow [S]I$
$0 \leq i \leq 10 \wedge s = \Sigma k \bullet (i+1 \leq k \leq 10 \mid b(k)) \wedge i \neq 0$
$\quad \Rightarrow [i, s := i-1, s + b(i)]I$
RHS $= 0 \leq i-1 \leq 10 \wedge s + b(i) = \Sigma k \bullet (i \leq k \leq 10 \mid b(k))$
$= 0 \leq i-1 \leq 10 \wedge s = \Sigma k \bullet (i+1 \leq k \leq 10 \mid b(k))$
First conjunct follows from 1st and 3rd conjumcts of LHS.

Second conjunct is 2nd conjunct of LHS.

Condition 2: $I \wedge \neg E \Rightarrow P$

$$0 \leq i \leq 10 \wedge s = \Sigma k \bullet (i+1 \leq k \leq 10 \mid b(k)) \wedge i = 0$$
$$\Rightarrow\ s = \Sigma k \bullet (1 \leq k \leq 10 \mid b(k))$$

RHS follows from LHS with $i = 0$.

Condition 3: $I \wedge E \Rightarrow v \in \mathbb{N}$

RHS is $i \in \mathbb{N}$ which follows from the first conjunct of I.

Condition 4: $I \wedge E \wedge i = i_0 \ \Rightarrow\ [S]i < i_0$

RHS $= i < i_0[i-1/i] = i - 1 < i_0$ which follows from $i = i_0$ on LHS.

Condition 5: $[Init]I$

$$[i, s := 10, 0]I \ = \ 0 \leq 10 \leq 10 \wedge 0 = \Sigma k \bullet (11 \leq k \leq 10 \mid b(k)) \ = \ \mathit{true}$$

(b) **Condition 1:** $I \wedge E \Rightarrow [S]I$

$$1 \leq i \leq n+1 \wedge x \notin b[1\,..\,i-1] \wedge i \leq n \wedge x \neq b(i)$$
$$\Rightarrow [i := i+1]I$$

RHS $= 1 \leq i+1 \leq n+1 \wedge x \notin b[1\,..\,i]$

$= 1 \leq i+1 \wedge i \leq n \wedge x \notin b[1\,..\,i-1] \wedge x \neq b(i)$

Each conjunct is implied by LHS.

Condition 2: $I \wedge \neg E \Rightarrow P$

LHS $= 1 \leq i \leq n+1 \wedge x \notin b[1\,..\,i-1] \wedge (i > n \vee x = b(i))$

Can expand this using $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$. However, if you do this you find the $A \wedge C$ part needs to know that $i \leq n$. (Try it!) So, at this point it's useful to make that explicit using the identity: $A \vee B = A \vee (\neg A \wedge B)$. This allows us to rewrite the second conjunct as: $i > n \vee (i \leq n \wedge x = b(i))$.

Now the LHS expands to:

$$(1 \leq i \leq n+1 \wedge x \notin b[1\,..\,i-1] \wedge i > n) \vee$$
$$(1 \leq i \leq n+1 \wedge x \notin b[1\,..\,i-1] \wedge i \leq n \wedge x = b(i))$$

This implies $(i = n+1 \wedge x \notin b[1\,..\,n]) \vee (1 \leq i \leq n \wedge x = b(i))$ as required.

Condition 3: $I \wedge E \Rightarrow v \in \mathbb{N}$

RHS is $n - i \in \mathbb{N}$ or equivalently, $n \geq i$. This is given in E.

Condition 4: $I \wedge E \wedge v = v_0 \ \Rightarrow\ [S]v < v_0$

RHS $= [i := i+1]n - i, v_0 \ = \ n - (i+1) < v_0$

From LHS, $v_0 = v$, giving: $n - (i+1) < n - i$ which is true.

Condition 5: $[Init]I$

This is: $1 \leq 1 \leq n+1 \wedge x \notin b[1\,..\,0]$

which simplifies to $1 \leq n+1$ and this is implied by the precondition.

5. Suitable **precondition**: $n \geq 1$

Suitable **postcondition**: $d = card(b \rhd \{x\})$
Since the domain of b is $1\,..\,n$ the postcondition is equivalent to:
$d = card((\{1\,..\,n\} \lhd b) \rhd \{x\})$

Plan is to start at the first position and increase towards n. Introduce variable k for this. This suggests:

Invariant: $d = card((\{1\,..\,k\} \lhd b) \rhd \{x\}) \wedge 0 \leq k \leq n$

Guard: $k \neq n$

By construction, this satisfies loop condition 2 ($I \wedge \neg\, E \Rightarrow P$).

A sensible initialisation would be to set both d and k to 0. For loop condition 5:
$[Init]I$
$= 0 = card((\{1\,..\,0\} \lhd b) \rhd \{x\}) \wedge 0 \leq 0 \leq n$
$= 0 \leq n$
which is given by the precondition.,

The program is:

```
    $d, k := 0, 0$
    WHILE  $k \neq n$
    DO  $k; = k + 1;$
        IF  $b(k) = x$  THEN  $d := d + 1$  END
    END
```

Loop condition 1 requires: $I \wedge E \Rightarrow [S]I$
LHS $= d = card((\{1\,..\,k\} \lhd b) \rhd \{x\}) \wedge 0 \leq k \leq n \wedge k \neq n$
RHS $= [k := k+1;\ IF\ b(k) = x\ THEN\ d := d+1\ END](d = card((\{1\,..\,k\} \lhd b) \rhd \{x\})$
$\wedge\, 0 \leq k \leq n)$
$= [k := k+1]((b(k) = x \Rightarrow (d+1 = card((\{1\,..\,k\} \lhd b) \rhd \{x\}) \wedge 0 \leq k \leq n))$
$\wedge$
$(b(k) \neq x \Rightarrow (d = card((\{1\,..\,k\} \lhd b) \rhd \{x\}) \wedge 0 \leq k \leq)))$
$= (b(k+1) = x \Rightarrow (d+1 = card((\{1\,..\,k+1\} \lhd b) \rhd \{x\}) \wedge 0 \leq k+1 \leq n))$
$\wedge$
$(b(k+1) \neq x \Rightarrow (d = card((\{1\,..\,k+1\} \lhd b) \rhd \{x\}) \wedge 0 \leq k+1 \leq)))$
LHS includes $0 \leq k \leq n \wedge k \neq n$ so this implies $0 \leq k+1 \leq n$ on both branches of RHS.
Also, from LHS, $d = card((\{1\,..\,k\} \lhd b) \rhd \{x\})$.
So, if $b(k+1) = x$ then $card((\{1\,..\,k+1\} \lhd b) \rhd \{x\}) = d+1$
and if $b(k+1) \neq x$ then $card((\{1\,..\,k+1\} \lhd b) \rhd \{x\}) = d$
as required.

For termination, as suitable variant is $n - k$.

Loop condition 3: $I \wedge E \Rightarrow v \in \mathbb{N}$
LHS $= n - k \in \mathbb{N}$ which follows from I since $n \geq k$.

Loop condition 4: As in previous examples, k is strictly increased by each iteration of the loop, so $n - k$ strictly decreases.