CS 142 Problem set #3 Timur Kuzhagaliyev

① a) $\text{transient}(p) \wedge [p' \Rightarrow p]$

$\equiv$ {def. of transient}

$(\exists a :: \{p\}\, a \,\{\neg p\}) \wedge [p' \Rightarrow p]$

$\equiv$ {def. of $\exists$, distribution of $\wedge$}

$(\exists a :: (\{p\}\, a\, \{\neg p\}) \wedge [p' \Rightarrow p])$

$\equiv$ {Hoare triples: strengthening LHS and weakening RHS}

$(\exists a :: \{p'\}\, a \,\{\neg p'\})$

$\equiv$

$\text{transient}(p')$

b) False. Consider a program that starts with $p := 0$ and has only one action: $p = 0 \rightarrow p := 3$.

Clearly, $p \leq 2$ is transient and $[p \leq 2 \Rightarrow p \leq 3]$, but $p \leq 3$ is not transient as no action can take us to $p > 3$.

② a) $P \leadsto P$

$\Leftarrow$ {from slides}

P ensures P

$\equiv$ {Def of ensures}

$(P \wedge \neg P)\ \text{next}\ (P \vee P) \wedge \text{transient}(P \wedge \neg P)$

$\equiv$ {$P \wedge \neg P \equiv \text{false}$, $P \vee P \equiv P$}

$\text{false next } P \wedge \text{transient false}$

$\equiv$ {false next $P \equiv$ true (from H/W #2), def. of transient}

$\text{true} \wedge (\exists a : \{\text{false}\}\, a\, \{\text{true}\})$

$\equiv$ {true is identity of $\wedge$}

...

$\equiv$ ...

$(\exists a :: \{false\}\, a\, \{true\})$

$\equiv$ { ~~def~~. assignment axiom }

$(\exists a :: false \Rightarrow true)$

$\equiv$ { def. of $\Rightarrow$ }

$(\exists a :: true)$

$\equiv$ { def. of $\exists$, $\vee$ }

$true$

b) $P \leadsto (P \wedge Q)$

$\Leftarrow$ { antecedent strengthening, from slides }

P ensures $(P \wedge Q)$

$\equiv$ { Def. of ensures }

$(P \wedge \neg(P \wedge Q))$ next $(P \vee (P \wedge Q)) \wedge$ transient $(P \wedge \neg(P \wedge Q))$

$\equiv$ { $P \wedge \neg(P \wedge Q) \equiv P \wedge (\neg P \vee \neg Q) \equiv P \wedge \neg Q$, $P \vee (P \wedge Q) \equiv P$ }

$(P \wedge \neg Q)$ next $(P) \wedge$ transient $(P \wedge \neg Q)$

$\equiv$ { Def. of next, ~~def. of~~ assignment axiom }

$(\forall a :: P \wedge \neg Q \Rightarrow P_a) \wedge$ transient $(P \wedge \neg Q)$

$\Leftarrow$ { antecedent strengthening }

$(\forall a :: P \Rightarrow P_a) \wedge$ transient $(P \wedge \neg Q)$

$\equiv$ { assignment axiom }

$(\forall a :: \{P\}\, a\, \{P\}) \wedge$ transient $(P \wedge \neg Q)$

$\equiv$ { def of next, stable }

stable$(P) \wedge$ transient $(P \wedge \neg Q)$

(2) c) False: Consider a program that starts with $x := 0$, and has a single action $x := x+1$. Clearly $x=0 \leadsto x=2$ and $x=0 \leadsto x=3$, but $(x=0 \wedge x=0) \not\leadsto (x=2 \wedge x=3)$.

③ We have no guarded actions in our program, so to find the fixed point we need to replace all assignments with equalities and take their conjunction, what gives us:

$$FP \equiv (r = f(r)) \wedge (r = g(r)) \wedge (r = h(r)) \equiv r = f(r) = g(r) = h(r)$$

Indeed: $stable\,(FP)$

$\equiv$ {def. of stable}

FP next FP

$\equiv$ {def. of next}

$(\forall a :: \{FP\}\, a\, \{FP\})$

$\equiv$ {def. of program, FP}

$(\forall a :: true)$

$\equiv$ {def. of $\forall$, $\wedge$}

$true$

Invariant is $r \le M$. Indeed, initially we have $r = 0$ and $f(r), g(r), h(r) \geq 0$ as we can't have negative time, hence $r \le M$.

$stable\,(r \le M)$

$\equiv$ {def. of stable}

$(r \le M)$ next $(r \le M)$

$\equiv$ {def. of next}

$(\forall a :: \{r \le M\}\, a\, \{r \le M\})$

$\equiv$ {repeat step on the left for $f(r), g(r), h(r)$}

$(\forall a :: true)$

$\equiv$ {def. of $\forall$, $\wedge$}

$true$

$\{r \le M\}\ r := f(r)\ \{r \le M\}$

$\equiv$ {assignment axiom}

$r \le M \Rightarrow f(r) \le M$

$\equiv$ { $f(r)$ is the soonest time of availability, since $r \le M$ $f(r)$ can either be equal to M or less than M if professor is available sooner than other, so by definition of M }

$true$

③ cont.

r as a metric satisfies 3 key properties:

① It is guaranteed to never decrease:

$$(r = k) \text{ next } (r \geq k)$$

$\equiv$ { def. of next }

$$(\forall a :: \{r = k\}\ a\ \{r \geq k\})$$

$\equiv$ { def. of program }

$$\{r=k\}\ r := f(r)\ \{r \geq k\} \wedge \{r=k\}\ r := g(r)\ \{r \geq k\} \wedge \{r=k\}\ r := h(r)\ \{r \geq k\}$$

$\equiv$ { assignment axiom 3 times }

$$(r = k \Rightarrow f(r) \geq k) \wedge (r = k \Rightarrow g(r) \geq k) \wedge (r = k \Rightarrow h(r) \geq k)$$

$\equiv$ { $f(r), g(r), h(r) \geq r$ }

true $\wedge$ true $\wedge$ true

$\equiv$ { def. of $\wedge$ }

true

② r is clearly bound above by the invariant, $r \leq M$.

③ There is an action that increases r if it's below M:

$$\text{transient } (r = k \wedge r < M)$$

$\equiv$ { Def. of transient }

$$(\exists a :: \{r = k \wedge r < M\}\ a\ \{r \neq k \vee r \geq M\})$$

$\equiv$ { $(r=k)$ next $(r \geq k)$ }

$$(\exists a :: \{r = k \wedge r < M\}\ a\ \{r > k \vee r \geq M\})$$

$\equiv$ { at any point $r < M$, $f(r) \neq g(r) \vee g(r) \neq h(r) \vee f(r) \neq h(r)$. pick the ~~smallest~~ function such that $p(r) \neq r$ } ~~p(r)~~ biggest $p(r)$

$$\{r = k \wedge r < M\}\ r := p(r)\ \{r > k \vee r \geq M\}$$

$\equiv$

$\equiv$ {assignment axiom}

$(r=k \wedge r<M) \Rightarrow (p(r)>k \vee p(r) \geq M)$

$\equiv$ {definition of $p(r)$: $p(r) \geq r$, but $p(r) \neq r$ hence $p(r) > r$}

$(r=k \wedge r<M) \Rightarrow (true \vee p(r) \geq M)$

$\equiv$ {predicate calculus}

$k<M \Rightarrow true$

$\equiv$ {def. of $\Rightarrow$}

$true$

Now we can prove that the program terminates:

$true$

$\Rightarrow$ {proof above}

$transient\,(r=k \wedge r<M)$

$\Rightarrow$ {from slides}

$r=k \wedge r<M \rightsquigarrow r \neq k \vee r \geq M$

$\equiv$ {$(r=k)$ next $(r \geq k)$}

$r=k \wedge r<M \rightsquigarrow r>k \vee r \geq M$

$\equiv$ {$X \vee Y \equiv (\neg Y \wedge X) \vee Y$}

$r<M \wedge r=k \rightsquigarrow (r<M \wedge r>k) \vee r \geq M$

$\Rightarrow$ {induction}

$r<M \rightsquigarrow r \geq M$

$\equiv$ {invariant, def. of fixed point}

$r<M \rightsquigarrow FP$

$\equiv$ {initially $r<M$}

$true \rightsquigarrow FP$

④ a) F is a metric as it satisfies the following properties:

① F is guaranteed to not decrease:

Let F be the metric before applying the action, and F' after.

If guard is inactive, then no changes occur, clearly $F = F'$.

If guard is active, then some $D[i,k]$ gets a smaller value, increasing the index in $L[i,k]$ and decreasing F', by definition, hence $F \leq F'$.

Therefore $(F = k)$ next $(F \geq k)$.

② F is the sum of indices in some collection of finite lists, so it's clearly bound by the sum of last indices in these lists. Define this value as M.

③ There is an action increasing F when its below the bound:

transient $(F = k \wedge F < M)$

$\equiv$ { def. of transient }

$(\exists a :: \{F = k \wedge F < M\}\, a\, \{F \neq k \vee F \geq M\})$

$\equiv$ { pick an action s.t. the guard is activated, use assignment axiom }

$F = k \wedge F < M \Rightarrow F' \neq k \vee F' \geq M$

$\Leftarrow$ $\{X \Rightarrow Z \Rightarrow X \wedge Y \Rightarrow Z\}$

$F = k \Rightarrow F' \neq k \vee F' \geq M$

$\equiv$ { F is non-decreasing }

$F = k \Rightarrow F' > k \vee F' \geq M$

$\Leftarrow$ $\{X \Rightarrow Y \Rightarrow X \Rightarrow Y \vee Z\}$

$F = k \Rightarrow F' > k$

$\equiv$ { By def. of our action, guard was activated so some $D[i,k]$ got a smaller value, increasing F }

true

(4) b) Let $\pi(i,k)$ be the shortest possible path length between i and k.

Then $FP \equiv \forall i,j,k\ \ \pi(i,k) \le D[i,j] + D[j,k]$.

Note that by definition of $\pi(i,k)$ and F, FP is also $F \geq M$.

Invariant is $\forall j,k : D[j,k] \le W[j,k]$.

$$
\begin{aligned}
& \text{true} \\
\Rightarrow\ & \{\text{proof above}\} \\
& \text{transient}(F=k \wedge F<M) \\
\Rightarrow\ & \{\text{from slides}\} \\
& F=k \wedge F<M \rightsquigarrow F\neq k \vee F \geq M \\
\equiv\ & \{X \vee Y \equiv (X \wedge \neg Y) \vee Y\} \\
& F=k \wedge F<M \rightsquigarrow (F<M \wedge F\neq k) \vee F \geq M \\
\equiv\ & \{F \text{ is non-decreasing}\} \\
& F<M \wedge F=k \rightsquigarrow (F<M \wedge F>k) \vee F \geq M \\
\Rightarrow\ & \{\text{induction}\} \\
& F<M \rightsquigarrow F \geq M \\
\equiv\ & \{\text{def. of fixed point}\} \\
& F<M \rightsquigarrow FP \\
\equiv\ & \{\text{initially } F<M\} \\
& \text{true} \rightsquigarrow FP
\end{aligned}
$$

$\therefore$ program terminates.