TEORÍA DE LA COMPUTACIÓN Y VERIFICACIÓN DE PROGRAMAS

TP 6 - Verificación de programas(clases 12 y 13)

Ejercicio 1. Se define la postcondición más fuerte de la siguiente manera:

$$post(p,\ S)\ =\ \{\sigma'\ |\ \exists\sigma\ :\ \sigma\ |=\ p \wedge val(\pi(S,\ \sigma))\ =\ \sigma'\ \neq \bot\}$$

es decir que un estado está en post(p, S) si es el estado final de una computación finita de S que arranca desde un estado inicial que satisface p.
Y se define la precondición liberal más débil de la siguiente manera:

$$pre(S,\ q)\ =\ \{\sigma\ |\ \forall\sigma'\ :\ val(\pi(S,\ \sigma))\ =\ \sigma'\ \neq \bot \rightarrow \sigma'| = q\}$$

es decir que un estado está en pre(S, q) si es el estado inicial a partir del cual se obtiene, por la ejecución de S, si termina, un estado final que satisface q. Probar:

(a) $| = \{p\}S\{q\} \leftrightarrow post(p,S) \subseteq \{\sigma|\sigma| = q\}$

(b) $| = \{p\}S\{q\} \leftrightarrow \{\sigma|\sigma| = p\} \subseteq pre(S,q)$

Se probará (a):
$| = \{p\}S\{q\} \leftrightarrow post(p,S) \subseteq \{\sigma|\sigma| = q\}$

$| = \{p\}S\{q\} \rightarrow post(p,S) \subseteq \{\sigma|\sigma| = q\}$
Si $| = \{p\}S\{q\}$ entonces $(\sigma| = p \wedge val(\pi(S,\ \sigma)) \neq \bot) \rightarrow val(\pi(S,\ \sigma)) = \sigma'| = q$
Asumiendo $val(\pi(S,\ \sigma)) \neq \bot$ y $\sigma| = p$ para cualquier estado σ, entonces $post(p,S)$ estaría conformado, según la definición de correctitud parcial por aquellos valores de $\sigma'| = q$ ($val(\pi(S,\ \sigma)) = \sigma'| = q$) para cualquier input que valide p sobre S**.**
Entonces queda demostrado que $post(p,S) \subseteq \{\sigma|\sigma| = q\}$ **.**

$| = \{p\}S\{q\} \leftarrow post(p,S) \subseteq \{\sigma|\sigma| = q\}$
Considerando $post(p,S) \subseteq \{\sigma|\sigma| = q\}$, podríamos redefinir $post(p,S)$ de la siguiente forma:
$post(p,\ S)\ =\ \{\sigma'\ |\ \exists\sigma\ :\ \sigma\ |=\ p \wedge val(\pi(S,\ \sigma))\ =\ \sigma'\ \neq \bot \wedge \sigma'| = q\}$
Con esta nueva definición podemos probar las 2 situaciones de $\{p\}S\{q\}$ **:**

1. si $val(\pi(S,\ \sigma)) = \bot \vee \sigma\ | \neq p$ entonces $| = \{p\}S\{q\}$
2. si $post(p,\ S) \neq \varnothing$ entonces $| = \{p\}S\{q\}$ por la nueva definición de $post(p,S)$ ya que si $\sigma\ | =\ p \wedge val(\pi(S,\ \sigma))\ =\ \sigma'\ \neq \bot \wedge \sigma'| = q$ es válido, entonces $(\sigma| = p \wedge val(\pi(S,\ \sigma)) \neq \bot) \rightarrow val(\pi(S,\ \sigma)) = \sigma'| = q$ es válido.

Se probará (b):
$| = \{p\}S\{q\} \leftrightarrow \{\sigma|\sigma| = p\} \subseteq pre(S,q)$

$| = \{p\}S\{q\} \rightarrow \{\sigma|\sigma| = p\} \subseteq pre(S,q)$

$pre(S,\ q) = \{\sigma \mid \forall\sigma' : val(\pi(S,\ \sigma)) = \sigma' \neq \perp \rightarrow \sigma' | = q\}$

Si $| = \{p\}S\{q\}$ entonces $(\sigma| = p \wedge val(\pi(S,\ \sigma)) \neq \perp) \rightarrow val(\pi(S,\ \sigma)) = \sigma'| = q$

1. si $val(\pi(S,\ \sigma)) = \perp$ entonces se cumple $| = \{p\}S\{q\}$, por lo que $\sigma \in pre(S,\ q)$ pero sólo pertenecería a $\{\sigma|\sigma| = p\}$ si $\sigma| = p$.
2. si $\sigma| \neq p$ entonces se cumple $| = \{p\}S\{q\}$, por lo que $\sigma \in pre(S,\ q)$.
3. si $\sigma| = p \wedge val(\pi(S,\ \sigma)) \neq \perp$ entonces $val(\pi(S,\ \sigma)) = \sigma'| = q$, por lo que $\sigma \in pre(S,\ q)$ y $\sigma \in \{\sigma|\sigma| = p\}$.

Por lo que queda demostrado $\{\sigma|\sigma| = p\} \subseteq pre(S, q)$

$| = \{p\}S\{q\} \leftarrow \{\sigma|\sigma| = p\} \subseteq pre(S, q)$

Sabiendo que $\{\sigma|\sigma| = p\} \subseteq pre(S, q)$ entonces $\{\sigma|\sigma| = p\}$ se podría definir como:

$\{\sigma|\ \sigma| = p \wedge val(\pi(S,\ \sigma)) = \sigma' \neq \perp \rightarrow \sigma'| = q\}$

Con esta nueva definición podemos probar las 2 situaciones de $\{p\}S\{q\}$:

1. si $val(\pi(S,\ \sigma)) = \perp \vee \sigma | \neq p$ entonces $| = \{p\}S\{q\}$
2. si $\{\sigma|\ \sigma| = p \wedge val(\pi(S,\ \sigma)) = \sigma' \neq \perp \rightarrow \sigma'| = q\} \neq \emptyset$ entonces $| = \{p\}S\{q\}$ ya que si $\sigma | = p \wedge val(\pi(S,\ \sigma)) = \sigma' \neq \perp \rightarrow \sigma'| = q$ es válido, entonces $(\sigma| = p \wedge val(\pi(S,\ \sigma)) \neq \perp) \rightarrow val(\pi(S,\ \sigma)) = \sigma'| = q$ es válido.

Ejercicio 2. En el trabajo práctico anterior se pidió probar usando el método H:

$\{x \geq 0 \wedge y > 0\}S_{idiv} ::\ q := 0;\ r := x;\ while\ r \geq y\ do\ r := r - y;\ q := q + 1\ od\ \{x = q.y + r \wedge 0 \leq r < y\}$

Sidiv un programa PLW que calcula por restas sucesivas la división entera de x sobre y en q, dejando el resto en r.

Se pide ahora probar en H:

$\{x > 0 \wedge y = 0\}S_{idiv}\{false\}$

es decir que el programa Sidiv no termina a partir de la precondición ($x > 0 \wedge y = 0$**).**

Se quiere probar $\{x > 0 \wedge y = 0\}S_{idiv}\{false\}$

Se propone como invariante $p = (x = q.y + r \wedge 0 < r \wedge y = 0)$

La prueba se estructurará de la siguiente manera.

a. $\{x > 0 \wedge y = 0\}q := 0;\ r := x\{p\}$
b. $\{p\}while\ r \geq y\ do\ r := r - y;\ q := q + 1\ od\{p \wedge \neg(r \geq y)\}$
c. $p \wedge \neg(r \geq y) \rightarrow false$

a)

1. $\{x > 0 \wedge y = 0 \wedge x = q.y + x\}\ r := x\{p\}$ (ASI)
2. $\{x > 0 \wedge y = 0 \wedge x = 0.y + x\}q := 0\{x > 0 \wedge y = 0 \wedge x = 0.y + x\}$ (ASI)
3. $(x > 0 \wedge y = 0 \wedge x = 0.y + x) \rightarrow (x > 0 \wedge y = 0)$ (MAT)
4. $\{x > 0 \wedge y = 0\}q := 0;\ r := x\{p\}$ (1,2,SEC,3,CONS)

b)

1. $\{x = (q + 1).y + r \wedge 0 < r \wedge y = 0\}q := q + 1\{p\}$ (ASI)
2. $\{x = (q + 1).y + (r - y) \wedge 0 < r \wedge y = 0\}r := r - y\{x = (q + 1).y + r \wedge 0 < r \wedge y = 0\}$ (ASI)

3. $(x = (q+1).y + (r-y) \wedge 0 < r \wedge y = 0) \rightarrow (x = q.y + r \wedge 0 < r \wedge r \geq y \wedge y = 0)$ (MAT)
4. $\{x = q.y + r \wedge 0 < r \wedge r \geq y \wedge y = 0\} r := r{-}y;\ q := q+1 \{p\}$ (1,2,SEC,3,CONS)
5. $(x = q.y + r \wedge 0 < r \wedge r \geq y \wedge y = 0) \rightarrow (x = q.y + r \wedge 0 < r \wedge y = 0)$ (MAT)
6. $\{p\} while\ r \geq y\ do\ r := r{-}y;\ q := q+1\ od \{p \wedge \neg(r \geq y)\}$ (4,REP,5,CONS)

c)

1. $x = q.y + r \wedge 0 < r \wedge y = 0 \wedge \neg(r \geq y) \rightarrow false$ (MAT)

c.1) es false porque $\neg(r \geq y) = r < y$ es absurdo ya que $0 < r \wedge y = 0$

Ejercicio 3. Probar:

$\langle x{\geq}0 \wedge y{\geq}0 \rangle\ S_{prod} :: prod := 0;\ k := y;\ while\ k > 0\ do\ prod := prod + x;\ k := k-1\ od\ \langle true \rangle$

Ayuda: S_{prod} ***calcula en la variable prod el producto entre x e y. Notar que k se decrementa en cada iteración y que se mantiene siempre mayor o igual que cero.***

Probaremos

$\langle x{\geq}0 \wedge y{\geq}0 \rangle\ S_{prod}\ \langle true \rangle$

Se propone como invariante $p = (k \geq 0)$
y la cota $t = k$
para facilitar la resolución dividiremos el problema en 3 partes:

inicialización:

$\langle x{\geq}0 \wedge y{\geq}0 \rangle prod := 0;\ k := y \langle p \rangle$

repetición:

a. $\langle p \wedge k > 0 \rangle prod := prod + x;\ k := k-1 \langle p \rangle$
b. $\langle p \wedge k > 0 \wedge k = Z \rangle prod := prod + x;\ k := k-1 \langle p \wedge k < Z \rangle$
c. $(k \geq 0) \rightarrow (k \geq 0)$

Por lo que:

d. $\langle p \rangle while\ k > 0\ do\ prod := prod + x;\ k := k-1\ od \langle p \wedge \neg(k > 0) \rangle$

final:

$\langle x{\geq}0 \wedge y{\geq}0 \rangle S_{prod} \langle true \rangle$

Ejercicio 4. Probar sin recurrir a la completitud relativa de H (es decir que la prueba debe ser sintáctica) que para todo programa S de PLW y toda aserción q de Assn se cumple:

Tr |─ H {false} S {q}

Ayuda: Utilizar inducción estructural sobre la forma de los programas S, similar a lo visto en clase para probar sintácticamente la fórmula {true} S {true}.

Utilizaremos la inducción estructural sobre los programas de PLW:

Base de la inducción:

1. $S :: skip$
Por SKIP {false}skip{false}
Por CONS {false}skip{q}
2. $S :: x := e$
Por ASI {false}x:=e{false}
Por CONS {false}x:=e{q}

Paso inductivo:

3. $S :: S_1; S_2$
Por hipótesis inductiva: $\{false\}S_1\{false\}$ y $\{false\}S_2\{false\}$
Por SEC: $\{false\}S_1; S_2\{false\}$
Por CONS: $\{false\}S_1; S_2\{q\}$
4. $S :: if\ B\ then\ S_1\ else\ S_2\ fi$
Por hipótesis inductiva: $\{false\}S_1\{false\}$ y $\{false\}S_2\{false\}$
Por MAT: $false \wedge B \rightarrow false$ y $false \wedge \neg B \rightarrow false$
Por CONS: $\{false \wedge B\}S_1\{false\}$ y $\{false \wedge \neg B\}S_2\{false\}$
Por COND: $\{false\}\ if\ B\ then\ S_1\ else\ S_2\ fi\ \{false\}$
Por CONS: $\{false\}\ if\ B\ then\ S_1\ else\ S_2\ fi\ \{q\}$
5. $S :: while\ B\ do\ S\ od$
Por hipótesis inductiva: $\{false\}S\{false\}$
Por MAT: $(false \wedge B) \rightarrow false$
Por CONS: $\{false \wedge B\}S\{false\}$
Por REP: $\{false\}while\ B\ do\ S\ od\{false \wedge \neg B\}$
Por MAT: $(false \wedge \neg B) \rightarrow q$
Por CONS: $\{false\}while\ B\ do\ S\ od\{q\}$

Ejercicio 5. Probar la redundancia en H de la siguiente regla ya vista en clase, la regla OR:

$$\frac{\{p\}\ S\ \{q\}\ ,\ \{r\}\ S\ \{q\}}{\{p \vee r\}\ S\ \{q\}}$$

Es decir, probar que si Tr |– H {p} S {q} y Tr |– H {r} S {q} entonces Tr |– H { p v r} S {q}, usando sólo los axiomas SKIP y ASI y las reglas SEC, COND, REP y CONS del método H.
***Ayuda**: Utilizar inducción estructural sobre la forma de los programas S, similar a lo visto en clase para probar la redundancia de la regla AND.*

Utilizaremos la inducción estructural sobre los programas de PLW:

Base de la inducción:

1. $S :: skip$, y se tiene |– {p} skip {q} y |– {r} skip {q}

Debe ser $p \to q$ o $r \to q$ y por lo tanto $(p \vee r) \to q$

Por SKIP: $\{p \vee r\}skip\{p \vee r\}$

Por CONS: $\{p \vee r\}skip\{q\}$

2. $S :: x := e$, y se tiene |– {p} x:=e {q} y |– {r} x:=e {q}

Debe ser $p \to q$ o $r \to q$ y por lo tanto $(p \vee r) \to q$

Por ASI: $\{(p \vee r)[x|e]\}x := e\{p \vee r\}$

Por CONS: $\{(p \vee r)[x|e]\}x := e\{q\}$

<u>Paso inductivo:</u>

3. $S :: S_1; S_2$, y se tiene |– {p} $S_1; S_2$ {q} y |– {r} $S_1; S_2$ {q}

Debe ser $|-\{p\}S_1\{t_1\}$ y $|-\{t_1\}S_2\{q\}$,y $|-\{r\}S_1\{t_2\}$ y $|-\{t_2\}S_2\{q\}$

Por CONS: $|-\{p\}S_1\{t_1 \vee t_2\}$ y $|-\{r\}S_1\{t_1 \vee t_2\}$

Por hipótesis inductiva considerando lo anterior: $|-\{p \vee r\}S_1\{t_1 \vee t_2\}$

Por hipótesis inductiva: $\{t_1 \vee t_2\}S_2\{q\}$

Por SEC: $|-\{p \vee r\}S_1; S_2\{q\}$

4. $S :: if\ B\ then\ S_1\ else\ S_2\ fi$ se tiene $\{p\}if\ B\ then\ S_1\ else\ S_2\ fi\{q\}$ o $\{r\}if\ B\ then\ S_1\ else\ S_2\ fi\{q\}$

Debe ser $|-\{p \wedge B\}S_1\{q\}$ y $|-\{p \wedge \neg B\}S_2\{q\}$, o $|-\{r \wedge B\}S_1\{q\}$ y $|-\{r \wedge \neg B\}S_2\{q\}$

Por hipótesis inductiva: $|-\{p \vee r \wedge B\}S_1\{q\}$ y $|-\{p \vee r \wedge \neg B\}S_2\{q\}$

Por COND: $|-\{p \vee r\}if\ B\ then\ S_1\ else\ S_2\ fi\{q\}$

5. $S :: while\ B\ do\ S\ od$ y se tiene $\{p\}while\ B\ do\ S\ od\{q\}$ o $\{r\}while\ B\ do\ S\ od\{q\}$

Debe ser $|-\{q \wedge B\}S\{q\}$, con $p \to q$ o $r \to q$ y por lo tanto $(p \vee r) \to q$

Por REP: $\{q\}while\ B\ do\ S\ od\{q \wedge \neg B\}$

Por CONS: $\{p \vee r\}while\ B\ do\ S\ od\{q \wedge \neg B\}$

$(q \wedge \neg B) \to q$

Por CONS: $\{p \vee r\}while\ B\ do\ S\ od\{q\}$