Practica 5

Agustina Sol Rojas y Antonio Felix Glorioso Ceretti

Ejercicio 1.

Se define la postcondición más fuerte de la siguiente manera:

$post(p, S) = \{\sigma' \mid \exists\sigma : \sigma \models p \land val(\pi(S, \sigma)) = \sigma' \neq \bot\}$

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:

$pre(S, q) = \{\sigma \mid \forall\sigma': val(\pi(S, \sigma)) = \sigma' \neq \bot \longrightarrow \sigma' \models q\}$

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) $\models \{p\}\ S\ \{q\} \leftrightarrow post(p, S) \subseteq \{\sigma \mid \sigma \models q\}$

   Para probar $\models \{p\}\ S\ \{q\} \leftrightarrow post(p, S) \subseteq \{\sigma \mid \sigma \models q\}$ se tiene que probar:

   (1) $\models \{p\}\ S\ \{q\} \longrightarrow post(p, S) \subseteq \{\sigma \mid \sigma \models q\}$

   Dado por $\models \{p\}\ S\ \{q\}$ sabemos que al aplicar S a cualquier estado que satisfaga p, el estado final de la ejecución de S va a cumplir q. Esto implica que el estado final que se encuentre en post(p, S) estará contenido en el conjunto de estados que satisfacen q ya que satisfizo a p como precondición.

   (2) $post(p, S) \subseteq \{\sigma \mid \sigma \models q\} \longrightarrow \models \{p\}\ S\ \{q\}$

Dado por $post(p, S) \subseteq \{\sigma \mid \sigma \models q\}$ sabemos que el estado final de una computación finita de S que arranca desde un estado inicial que satisface p estará contenido en los estados que satisfacen q. Por lo tanto cuando se ejecuta un S finito luego de un estado inicial que satisface p se va a llegar a un estado final que satisface q.

b) $\models \{p\}\ S\ \{q\} \leftrightarrow \{\sigma \mid \sigma \models p\} \subseteq pre(S, q)$

Para probar $\models \{p\}\ S\ \{q\} \leftrightarrow \{\sigma \mid \sigma \models p\} \subseteq pre(S, q)$ se tiene que probar:

(1) $\models \{p\}\ S\ \{q\} \longrightarrow \{\sigma \mid \sigma \models p\} \subseteq pre(S, q)$

Dado por $\models \{p\}\ S\ \{q\}$ sabemos que al aplicar S a cualquier estado que satisfaga p, el estado final de la ejecución de S va a cumplir q. Esto implica que cualquier estado que satisface p va a estar contenido en pre(S, q) ya que a partir del mismo cuando se ejecuta S, si termina, se acaba en q.

(2) $\{\sigma \mid \sigma \models p\} \subseteq pre(S, q) \longrightarrow\ \models \{p\}\ S\ \{q\}$

Dado por $\{\sigma \mid \sigma \models p\} \subseteq pre(S, q)$ sabemos que cualquier estado inicial que satisface p, va a estar contenido dentro de los estados iniciales que al ejecutar un S finito este termina en un estado que satisface q. Por lo tanto cuando se ejecuta un S finito luego de un estado inicial que satisface p se va a llegar a un estado final que satisface q.

Ejercicio 2.

En la clase práctica anterior se probó usando el método H:

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

siendo S_{div} un programa 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:

$\{x > 0 \land y = 0\}\ S_{div}\ \{false\}$

es decir que el programa S_{div} no termina a partir de la precondición $(x > 0 \land y = 0)$.

$\{x > 0 \land y = 0\}\ S_{div}\ \{false\}$

$p = (x = y.q + r \wedge r > 0 \wedge y = 0)$

Probar:

a) $\{x > 0 \ni y = 0\}\ q := 0;\ r := x\ \{p\}$
b) $\{p\}$ while $r\ \varepsilon\ y$ do $r := r - y;\ q := q + 1$ od $\{p \ni \neg(r \geq y)\}$
c) Aplicando SEC a a) y b) y luego CONS se llega a: $\{x > 0 \ni y = 0\}\ S_{div}\ \{false\}$

Prueba (a)

1. $\{x = y.q + x \wedge x > 0 \wedge y = 0\}\ r := x\ \{x = y.q + r \wedge r > 0 \wedge y = 0\}$ (ASI)

2. $\{x = y.0 + x \wedge x > 0 \wedge y = 0\}\ q := 0\ \{x = y.q + x \wedge x > 0 \wedge y = 0\}$ (ASI)

3. $\{x = y.0 + x \wedge x > 0 \wedge y = 0\}\ q := 0;\ r := x\ \{x = y.q + r \wedge r > 0 \wedge y = 0\}$ (1, 2, SEC)

4. $(x > 0 \ni y = 0) \rightarrow (x = y.0 + x \wedge x > 0 \wedge y = 0)$ (MAT)

5. $\{x > 0 \ni y = 0\}\ q := 0;\ r := x\ \{x = y.q + r \wedge r > 0 \wedge y = 0\}$ (3, 4, CONS)

Prueba (b)

6. $\{x = y.(q+1) + r \wedge r > 0 \wedge y = 0\}\ q := q + 1\ \{x = y.q + r \wedge r > 0 \wedge y = 0\}$ (ASI)

7. $\{x = y.(q+1) + (r - y) \wedge (r - y) > 0 \wedge y = 0\}\ r := r - y;\ \{x = y.(q+1) + r \wedge r > 0 \wedge y = 0\}$ (ASI)

8. $\{x = y.(q+1) + (r - y) \wedge (r - y) > 0 \wedge y = 0\}\ r := r - y;\ q := q + 1\ \{x = y.q + r \wedge r > 0 \wedge y = 0\}$
(6, 7, SEC)

9. $(x = y.q + r \wedge r > 0 \wedge y = 0 \wedge r \geq y) \rightarrow (x = y.(q+1) + (r - y) \wedge (r - y) > 0 \wedge y = 0)$ (MAT)

10. $\{x = y.q + r \wedge r > 0 \wedge y = 0 \wedge r \geq y\}\ r := r - y;\ q := q + 1\ \{x = y.q + r \wedge r > 0 \wedge y = 0\}$

(8, 9, CONS)

11. $\{x = y.q + r \wedge r > 0 \wedge y = 0\}$ while $r \geq y$ do $r := r - y$; $q := q + 1$ od $\{x = y.q + r \wedge r > 0 \wedge y = 0 \wedge \neg(r \geq y)\}$ (10, REP)

Prueba de (c)

12. $\{x > 0 \ni y = 0\}$ $q := 0$; $r := x$; while $r \geq y$ do $r := r - y$; $q := q + 1$ od $\{x = y.q + r \wedge r > 0 \wedge y = 0 \wedge \neg(r \geq y)\}$ (5, 11, SEC)

13. $(x = y.q + r \wedge r > 0 \wedge y = 0 \wedge \neg(r \geq y)) \rightarrow (\text{false})$ (MAT)

14. $\{x > 0 \ni y = 0\}$ Sdiv $\{\text{false}\}$ (12, 13, CONS)

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

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.

Invariante $p = (x.y = x.k + prod \wedge k \geq 0)$

Variante $t = k$

Inicializacion:

1. $\langle x \geq 0 \ni y \geq 0\rangle$ prod := 0; k := y $\langle x.y = x.k + prod \wedge k \geq 0\rangle$ (ASI*, SEC*, CONS*)

<u>Premisa 1: $\langle p \wedge B\rangle$ S $\langle p\rangle$</u>

2. $\langle x.y = x.k + prod \wedge k \geq 0 \wedge k > 0\rangle$ prod := prod + x ; k := k – 1 $\langle x.y = x.k + prod \wedge k \geq 0\rangle$ (ASI*, SEC*, CONS*)

<u>Premisa 2: $\langle p \wedge B \wedge t = Z\rangle$ S $\langle t < Z\rangle$</u>

3. $\langle x.y = x.k + prod \wedge k \geq 0 \wedge k > 0 \wedge k = Z \rangle$ prod := prod + x ; k := k – 1 $\langle k < Z \rangle$

(ASI*, SEC*, CONS*)

<u>Premisa 3: p → t ≥ 0</u>

4. $(x.y = x.k + prod \wedge k \geq 0) \longrightarrow (k \geq 0)$ (MAT)

<u>Conclusión: ⟨p⟩ while B do S od ⟨p ∧ ¬B⟩</u>

5. $\langle x.y = x.k + prod \wedge k \geq 0 \rangle$ while k > 0 do prod := prod + x ; k := k – 1 od $\langle x.y = x.k + prod \wedge k \geq 0 \wedge \neg(k > 0) \rangle$ (2, 3, 4 , REP*)

<u>Programa completo:</u>

6. $\langle x \geq 0 \ni y \geq 0 \rangle \; S_{prod} \langle true \rangle$ (1,5, SEC*, CONS*)

Ejercicio 4.

Probar la sensatez de la regla de invariancia vista en clase:

$$\frac{\{p\}\ S\ \{q\}}{\{r \wedge p\}\ S\ \{r \wedge q\}}$$

cuando las variables libres de r son disjuntas con las variables modificables por S.

Ayuda: Utilizar inducción sobre la longitud de las pruebas, como hicimos en clase.

Hay que probar $|\!-_H \{r \wedge p\}\ S\ \{r \wedge q\} \longrightarrow\ |\!= \{r \wedge p\}\ S\ \{r \wedge q\}$

- $|\!- \{r \wedge p\}\ S\ \{r \wedge q\}$ proviene de $|\!- \{p\}\ S\ \{q\}$
- Por hipótesis inductiva $|\!= \{p\}\ S\ \{q\}$
- Se cumple $|\!=\{r \wedge p\}\ S\ \{r \wedge q\}$

- Sea σ |= {r ^ p}, y asumamos que S termina desde σ en un estado σ' |= {r ^ q}
- Si σ |= p, entonces por hipótesis inductiva vale σ' |= q

- Entonces |– H {r ^ p} S {r ^ q} ⟶ |= {r ^ p} S {r ^ q}

Ejercicio 5.

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

Tr |– {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}

<u>Base de la inducción:</u>

S :: skip

Se cumple |– {false} skip {q} por el axioma SKIP. Por la semántica de PLW, (skip, σ) ⟶ (E, σ).

Como |= {false} skip {false}, si σ |= false entonces σ' |= q, y por lo tanto false ⟶ q. Prueba de {false} skip {q}:

Por SKIP: {false} skip {false}

Por MAT: false ⟶ q

Por CONS 1,2: {false} skip {q}

Por lo tanto: |– {false} skip {q}

S :: x := e

Se cumple |– {false} x := e {q} por el axioma ASI.

Por la semántica de PLW, (x := e, σ) → (E, σ[x | e]). Como |= {false} x := e {q}, si σ |= false entonces σ[x | e] |= q, y también σ |= q[x | e] por el Lema de Sustitución, por lo que false ⟶ q[x | e]. Prueba de {false} x := e {q}:

Por ASI: {q[x | e]} x := e {q}

Por MAT: false ⟶ q[x | e].

Por CONS 1,2 : {false} x := e {q}.

Por lo tanto: |– {false} x := e {q}.

Paso inductivo:

S :: S1 ; S2

Por hipótesis inductiva: |– {false} S1 {false} y |– {false} S2 {false}.

Por MAT: false → q.

Por SEC sobre lo anterior: |– {false} S1 ; S2 {false}.

Por CONS sobre lo anterior: |– {false} S1 ; S2 {q}.

S :: if B then S1 else S2 fi

Por hipótesis inductiva: |– {false} S1 {false} y |– {false} S2 {false}.

Por MAT: false → q.

Por MAT: false ^ B → false y false ^ ¬B → false.

Por CONS sobre lo anterior: |– {false ^ B} S1 {false} y |– {false ^ ¬B} S2 {false}.

Por COND sobre lo anterior: |– {false} if B then S1 else S2 fi {false}.

Por CONS sobre lo anterior: |– {false} if B then S1 else S2 fi {q}.

S :: while B do S1 od

Por hipótesis inductiva: {false} S1 {false}.

Por MAT: false ^ B → false.

Por CONS sobre lo anterior: {false ^ B} S1 {false}.

Por REP sobre lo anterior: {false} while B do S1 od {false ^ ¬B}.

Por MAT: false ^ ¬B → q.

Finalmente por CONS sobre lo anterior: |– {false} while B do S1 od {q}