

Práctica 7

Verificación de Programas

1. El lenguaje While y su semántica

Ejercicio 1

En la teórica se aclaró el lenguaje de predicados que se usa para escribir las pre y postcondiciones. Indicar cuál es la estructura de primer orden \mathcal{M} asociada a ese lenguaje indicando el dominio y la interpretación de las constantes, símbolos de función y símbolos de predicado.

Ejercicio 2

Supongamos que el estado σ verifica $\sigma(x) = -2$, $\sigma(y) = 5$ y $\sigma(z) = -1$. ¿Valen las siguientes afirmaciones?

1. $\sigma \models \neg(x + y < z)$
2. $\sigma \models y - x * z < z$
3. $\sigma \models \forall u(y < y \rightarrow y * z < u * z)$

Ejercicio 3

Dado el estado σ del ejercicio anterior, determinar el estado τ tal que $\sigma, P \Downarrow \tau$ y además exhibir la derivación del juicio, para cada uno de los siguientes programas:

1. $x := x + 1$
2. $y := 1; \text{while } (x! = 0) \{x := x + 1; y := y * 2\}$
3. $y := 1; \text{if } (x > 0) \text{ then } \{y := y + 1\} \text{ else } \{y := y - 1\}$

Ejercicio 4

Dar un ejemplo de un programa P tal que no existe ningún τ tal que $\sigma, P \Downarrow \tau$.

Ejercicio 5

Un comando común es la repetición indexada. Por ejemplo:

```
s:=1;
for (i:=0; i<9; i:=i+1)
  { s:=s*2 }
```

1. Mostrar cómo se podría simular un comando $\text{for}(C_1; B; C_2)C_3$ con los comandos que ya tenemos.
2. Proponer una regla de semántica operacional (i.e. de \Downarrow) para el for .

Ejercicio 6

Repetir la consigna del ejercicio anterior pero con el comando $\text{repeat } \{C\} \text{ until } (B)$.

2. Corrección parcial

Ejercicio 7

Probar la corrección parcial de las siguientes triplas de Hoare. Es decir, mostrar que:

1. $\vdash_{par} \llbracket y > 4 \wedge z > 1 \rrbracket y := y + z \llbracket y > 3 \rrbracket$
2. $\vdash_{par} \llbracket x = x_0 \rrbracket x := x + y \llbracket x = x_0 + y \rrbracket$
3. $\vdash_{par} \llbracket y + z > 4 \rrbracket y := y + z - 1; x := y + 2 \llbracket x > 5 \rrbracket$
4. $\vdash_{par} \llbracket x = x_0 \wedge y = y_0 \rrbracket z := x; x := y; y := z \llbracket x = y_0 \wedge y = x_0 \rrbracket$
5. $\vdash_{par} \llbracket x = x_0 \wedge y = y_0 \rrbracket x := x + y; y := x - y; x := x - y \llbracket x = y_0 \wedge y = x_0 \rrbracket$
6. $\vdash_{par} \llbracket \mathbf{T} \rrbracket \text{if } y > x \text{ then } \{m := y\} \text{ else } \{m := x\} \llbracket (x > y \wedge m = x) \vee (x \leq y \wedge m = y) \rrbracket$
7. $\vdash_{par} \llbracket y > 4 \rrbracket \text{if } (z > 1) \text{ then } \{y := y + z\} \text{ else } \{y := y - 1\} \llbracket y > 3 \rrbracket$

Ejercicio 8

Escribir un programa P tal que:

1. $\vdash_{par} \llbracket \mathbf{T} \rrbracket P \llbracket y = x + 2 \rrbracket$
2. $\vdash_{par} \llbracket \mathbf{T} \rrbracket P \llbracket z > x + y + 4 \rrbracket$

Ejercicio 9

Dados A y P tal que P no modifica ninguna variable que está en A . ¿Vale $\vdash_{par} \llbracket A \rrbracket P \llbracket A \rrbracket$?

Ejercicio 10

Probar que la siguiente variante de la regla para el condicional se puede derivar a partir de las demás:

$$\frac{\llbracket A_1 \rrbracket C_1 \llbracket C \rrbracket \quad \llbracket A_2 \rrbracket C_2 \llbracket C \rrbracket}{\llbracket (B \rightarrow A_1) \wedge (\neg B \rightarrow A_2) \rrbracket \text{if } B \text{ then } \{C_1\} \text{ else } \{C_2\} \llbracket C \rrbracket} \text{ (Condicional-Variante)}$$

Ejercicio 11

Identificar el invariante de las siguientes repeticiones condicionales:

1. `while (k! = n) {k := k + 1; j := j + 2}`
2. `while (z! = 0) {x := x + 1; z := z - 1}`
3. `while (k! = n) {k := k + 1; j := 2 * j}`

Ejercicio 12

Probar la corrección parcial de las siguientes triplas de Hoare:

1. $\vdash_{par} \llbracket n > 0 \wedge j = 1 \wedge k = k_0 \rrbracket \text{while } (k! = n) \{k := k + 1; j := j + 2\} \llbracket j = 1 + 2 * (n - k_0) \rrbracket$
2. $\vdash_{par} \llbracket x \geq 0 \rrbracket \text{Copy1} \llbracket x = y \rrbracket$ donde *Copy1* es el programa:

```

a := x;
y := 0;
while (a != 0) {
  y := y + 1;
  a := a - 1
}

```

3. $\vdash_{par} \llbracket y \geq 0 \rrbracket \text{Multi1} \llbracket z = x * y \rrbracket$ donde *Multi1* es el programa:

```

a := 0;
z := 0;
while (a != y) {
  z := z + x;
  a := a + 1
}

```

4. $\vdash_{par} \llbracket y = y_0 \wedge y \geq 0 \rrbracket \text{Multi2} \llbracket z = x * y_0 \rrbracket$ donde *Multi2* es el programa:

```

z := 0;
while (y != 0) {
  z := z + x;
  y := y - 1
}

```

5. $\vdash_{par} \llbracket x \geq 0 \rrbracket \text{Copy2} \llbracket x = y \rrbracket$ donde *Copy2* es el programa:

```

y := 0;
while (y != x) {
  y := y + 1
}

```

6. $\vdash_{par} \llbracket \neg(y = 0) \rrbracket \text{Div} \llbracket (x = d * y + r) \wedge (r < y) \rrbracket$ donde *Div* es el programa:

```
r := x;  
d := 0;  
while (r >= y) {  
  r := r - y;  
  d := d + 1  
}
```

Ejercicio 13

¿Vale $\vdash_{par} \llbracket \mathbf{T} \rrbracket Copy1 \llbracket x = y \rrbracket$?

3. Corrección total

Ejercicio 14

Probar la validez de las siguientes afirmaciones:

1. $\vdash_{tot} \llbracket x \geq 0 \rrbracket Copy1 \llbracket x = y \rrbracket$.
2. $\vdash_{tot} \llbracket y \geq 0 \rrbracket Multi1 \llbracket z = x * y \rrbracket$.
3. $\vdash_{tot} \llbracket y = y_0 \wedge y \geq 0 \rrbracket Multi2 \llbracket z = x * y_0 \rrbracket$.
4. $\vdash_{tot} \llbracket x \geq 0 \rrbracket Copy2 \llbracket x = y \rrbracket$.