Práctico 4: Introducción a la Programación Imperativa

Algoritmos y Estructuras de Datos I
2$^{\text{do}}$ cuatrimestre 2021

Esta guía tiene como objetivo introducir los conceptos elementales de la programación imperativa a través del análisis y la comprensión de la *traza* de ejecución de programas.

La semántica (el significado) de un programa imperativo es modificar el estado en que comienza la computación y obtener, eventualmente, un estado final. Si el programa es simple (**skip** y asignación ($x, y := E, E'$)), entonces podemos calcular el estado final fácilmente. En cambio, para las otras sentencias (comandos) necesitamos ir calculando estados intermedios. Una forma de escribir esos estados intermedios es anotar cada comando (y también cada guarda) con una etiqueta (como si fuera el número de línea) y listar los estados que obtenemos al ejecutar cada comando simple. Por ejemplo, dado el programa de la izquierda al anotar sus líneas obtenemos el programa de la derecha.

```
Const N : Int;
Var r, i : Int;
r, i := 0, 0;
do i < N →
   if
   p.i →
         r := r + 1
   ¬p.i →
         skip
   fi;
   i := i + 1
od
```

```
Const N : Int;
Var r, i : Int;
ℓ1  r, i := 0, 0;
ℓ2  do i < N →
ℓ3     if  p.i →
ℓ4                r := r + 1
ℓ5  □¬p.i →
ℓ6                skip
ℓ7     fi;
ℓ8     i := i + 1
ℓ9  od
```

Así podemos construir fácilmente la secuencia de estados intermedios y el valor de cada guarda en cada punto de la ejecución del programa. Por ejemplo, si $p.x \doteq x$ es primo y comenzamos con un estado $\sigma_0 : r \mapsto 7, i \mapsto 54$ y $N = 3$ tendremos la siguiente (traza de) ejecución:

línea(s)	nombre del estado	estado/guardas	aclaración
	σ_0	$r \mapsto 7$, $i \mapsto 54$, $N = 3$	estado inicial
ℓ_1	σ_1	$r \mapsto 0$, $i \mapsto 0$	después de ejecutar la asignación
ℓ_2		$i < N \equiv True$	evaluamos la guarda del ciclo
ℓ_3, ℓ_5		$p.i \equiv False\,, \neg p.i \equiv True$	evaluamos las guardas del condicional
ℓ_6	σ_2	$r \mapsto 0$, $i \mapsto 0$	ejecutamos el **skip**
ℓ_8	σ_3	$r \mapsto 0$, $i \mapsto 1$	ejecutamos la asignación
ℓ_2		$i < N \equiv True$	evaluamos la guarda del ciclo
ℓ_3, ℓ_5		$p.i \equiv False\,, \neg p.i \equiv True$	evaluamos las guardas del condicional
ℓ_6	σ_4	$r \mapsto 0$, $i \mapsto 1$	ejecutamos el **skip**
ℓ_8	σ_5	$r \mapsto 0$, $i \mapsto 2$	ejecutamos la asignación
ℓ_2		$i < N \equiv True$	evaluamos la guarda del ciclo
ℓ_3, ℓ_5		$p.i \equiv True\,, \neg p.i \equiv False$	evaluamos las guardas del condicional
ℓ_4	σ_6	$r \mapsto 1$, $i \mapsto 2$	ejecutamos la asignación
ℓ_8	σ_7	$r \mapsto 1$, $i \mapsto 3$	ejecutamos la asignación
ℓ_2		$i < N \equiv False$	evaluamos la guarda del ciclo
	σ_7	$r \mapsto 1$, $i \mapsto 3$	estado final

Notar que en el estado sólo están las variables que declaramos y que la(s) constante(s) solo aparece(n) en el estado inicial. Podemos observar que tenemos el patrón $\ell_2, \ldots, \ell_8, \ell_2$ repetido tantas veces como se ejecuta el ciclo. Otro punto es que las guardas de los condicionales se evalúan simultáneamente.

1. Sobre la ejecución del programa mostrado arriba, cuántas filas de la tabla debemos ver para decidir la línea siguiente (es decir el estado luego de ejecutar la línea correspondiente)?

2. Armá la traza de ejecución de los siguientes programas. Indicamos el estados inicial como σ_0. Para el punto 2k no hagas más de 20 líneas (mejor si descubrís antes por qué).

a) Var $x, y : Int$;
$\sigma_0 : x \mapsto 1, y \mapsto 10$
ℓ_1 $x := 7$

b) Var $x, y : Int$;
$\sigma_0 : x \mapsto 2, y \mapsto 5$
ℓ_1 $x := x + y$;
ℓ_2 $y := y + y$

c) Var $x, y : Int$;
$\sigma_0 : x \mapsto 2, y \mapsto 5$
ℓ_1 $y := y + y$;
ℓ_2 $x := x + y$

d) Var $x, y : Int$;
$\sigma_0 : x \mapsto 2, y \mapsto 5$
ℓ_1 $y, x := y + y, x + y$

e) Var $x, y : Int$;
$\sigma_0 : x \mapsto 3, y \mapsto 1$
ℓ_1 **if** $x \geq y \rightarrow$
ℓ_2 $x := 0$
ℓ_3 [] $x \leq y \rightarrow$
ℓ_4 $x := 2$
ℓ_5 **fi**

f) Var $x, y : Int$;
$\sigma_0 : (x \mapsto -100, y \mapsto 1)$
ℓ_1 **if** $x \geq y \rightarrow$
ℓ_2 $x := 0$
ℓ_3 [] $x \leq y \rightarrow$
ℓ_4 $x := 2$
ℓ_5 **fi**

g) Var $x, y : Int$;
$\sigma_0 : (x \mapsto 1, y \mapsto 1)$
ℓ_1 **if** $x \geq y \rightarrow$
ℓ_2 $x := 0$
ℓ_3 [] $x \leq y \rightarrow$
ℓ_4 $x := 2$
ℓ_5 **fi**

h) Var $i : Int$;
$\sigma_0 : (i \mapsto 4)$
ℓ_1 **do** $i \neq 0 \rightarrow$
ℓ_2 $i := i - 1$
ℓ_3 **od**

i) Var $i : Int$;
$\sigma_0 : (i \mapsto 400)$
ℓ_1 **do** $i \neq 0 \rightarrow$
ℓ_2 $i := 0$
ℓ_3 **od**

j) Var $i : Int$;
$\sigma_0 : (i \mapsto 4)$
ℓ_1 **do** $i < 0 \rightarrow$
ℓ_2 $i := i - 1$
ℓ_3 **od**

k) Var $i : Int$;
$\sigma_0 : (i \mapsto 0)$
ℓ_1 **do** $i \leq 0 \rightarrow$
ℓ_2 $i := i - 1$
ℓ_3 **od**

l) Var $r : Int$;
$\sigma_0 : (r \mapsto 3)$
ℓ_1 **do** $r \neq 0 \rightarrow$
ℓ_2 **if** $r < 0 \rightarrow$
ℓ_3 $r := r + 1$
ℓ_4 [] $r > 0 \rightarrow$
ℓ_5 $r := r - 1$
ℓ_6 **fi**
ℓ_7 **od**

3. Analizando las trazas que obtuviste:

   a) Considerando los ítems 2a, 2b, 2c y 2d: ¿obtuviste el mismo estado final en todos? ¿Hay estados intermedios diferentes?

   b) ¿Cuáles de esos programas dirías que son *equivalentes*?

   c) ¿Se puede escribir un programa equivalente al del ítem 2c con sólo una asignación múltiple? Si se puede, escriba el programa. Si no, explique por qué. A todo esto, ¿cuándo dos programas son equivalentes?

   d) Con respecto al programa del ítem 2k: ¿Cuántas veces se repite el patrón ℓ_1, ℓ_2, ℓ_1 en la primer columna de la traza? ¿Dejará de ser cierta la guarda en algún momento?

   e) ¿Cuál será el valor final de r si ejecutamos el programa de 2l en el estado $\sigma_0 : r \mapsto -20$? ¿Es ese programa equivalente al del punto 2j? ¿En qué estados son diferentes?

4. Escribí un programa que calcule el máximo entre x e y; claramente la gracia del ejercicio es asumir que no tenemos el operador max disponible. Escribí la traza para $\sigma_0 : x \mapsto 100, y \mapsto 2$ y la traza para $\sigma_0 : x \mapsto -3, y \mapsto 2$.