Universidad Nacional Autónoma de México

Facultad de Ciencias

Tarea Examen IV

Lenguajes de Programación

Dafne Bonilla Reyes
José Camilo García Ponce
Rodrigo Aldair Ortega Venegas

Profesor: Javier Enríquez Mendoza

Ayudante: Andrea Regina García Correa
Ayudante: Karla Denia Salas Jiménez
Ayudante Laboratorio: Ramón Arenas Ayala
Ayudante Laboratorio: Oscar Fernando Millán Pimentel

Noviembre 2023

Lenguajes de Programación

Tarea Examen 4

1. Considera el siguiente programa en el lenguaje While:

```
new z = 0;
while (y < x + 1) do
  (z := z + 1;
   x := x - y)
end
```

   a) Ejecuta el programa en el estado en el que $\sigma(x) = 17$ y $\sigma(y) = 5$ ¿Cuál es el estado resultante de la evaluación?

   Veamos que llegar al estado resultante nos tomará un total de 4 iteraciones, las cuales serán las siguientes:

   - Iteración 1:

	$\Diamond \succ \langle c_1; c_2, \sigma[x \to 17, y \to 5]\rangle$
	$\Diamond \succ \langle$ new $z = 0; c_2, \sigma[x \to 17, y \to 5]\rangle$
sec$(\Box, c_2)$;	$\Diamond \succ \langle$ new $z = 0; \sigma[x \to 17, y \to 5]\rangle$
sec$(\Box, c_2)$;	$\Diamond \prec \sigma[x \to 17, y \to 5, z \to 0]$
	$\Diamond \succ \langle c_2, \sigma[x \to 17, y \to 5, z \to 0]\rangle$
	$\Diamond \succ \langle$ while $(y < x + 1)$ do $(c_3; c_4)$ end, $\sigma[x \to 17, y \to 5, z \to 0]\rangle$
	$\Diamond \succ \langle$ if $(y < x + 1)$ then $((c_3; c_4);$ (while $(y < x + 1)$ do $(c_3; c_4)$ end)), else skip, $\sigma[x \to 17, y \to 5, z \to 0]\rangle$
	$\Diamond \succ \langle$ if $(5 < x + 1)$ then $((c_3; c_4);$ (while $(y < x + 1)$ do $(c_3; c_4)$ end)), else skip, $\sigma[x \to 17, y \to 5, z \to 0]\rangle$
	$\Diamond \succ \langle$ if $(5 < 17 + 1)$ then $((c_3; c_4);$ (while $(y < x + 1)$ do $(c_3; c_4)$ end)), else skip, $\sigma[x \to 17, y \to 5, z \to 0]\rangle$
	$\Diamond \succ \langle$ if $(5 < 18)$ then $((c_3; c_4);$ (while $(y < x + 1)$ do $(c_3; c_4)$ end)), else skip, $\sigma[x \to 17, y \to 5, z \to 0]\rangle$
	$\Diamond \succ \langle$ if $(\top)$ then $((c_3; c_4);$ (while $(y < x + 1)$ do $(c_3; c_4)$ end)), else skip, $\sigma[x \to 17, y \to 5, z \to 0]\rangle$
	$\Diamond \succ \langle (c_3; c_4);$ (while $(y < x + 1)$ do $(c_3; c_4)$ end), $\sigma[x \to 17, y \to 5, z \to 0]\rangle$
sec$(\Box,$ (while$(y < x + 1)$ do $(c_3; c_4)$ end))	$\Diamond \succ \langle (c_3, c_4), \sigma[x \to 17, y \to 5, z \to 0]\rangle$
sec$(\Box, c_4)$; sec$(\Box,$ (while$(y < x +$ 1) do $(c_3; c_4)$ end))	$\Diamond \succ \langle c_3, \sigma[x \to 17, y \to 5, z \to 0]\rangle$
sec$(\Box, c_4)$; sec$(\Box,$ (while$(y < x +$ 1) do $(c_; c_4)$ end))	$\Diamond \succ \langle z := z + 1, \sigma[x \to 17, y \to 5, z \to 0]\rangle$
sec$(\Box, c_4)$; sec$(\Box,$ (while$(y < x +$ 1) do $(c_3; c_4)$ end))	$\Diamond \succ \langle \sigma[x \to 17, y \to 5, z \to 1]\rangle$
sec$(\Box,$ (while$(y < x + 1)$ do $(c_3; c_4)$ end))	$\Diamond \succ \langle c_4, \sigma[x \to 17, y \to 5, z \to 1]\rangle$
sec$(\Box,$ (while$(y < x + 1)$ do $(c_3; c_4)$ end))	$\Diamond \succ \langle x := x - y, \sigma[x \to 17, y \to 5, z \to 1]\rangle$
sec$(\Box,$ (while$(y < x + 1)$ do $(c_3; c_4)$ end))	$\Diamond \prec \langle \sigma[x \to 12, y \to 5, z \to 1]\rangle$

- Iteración 2:

	$\Diamond \succ \langle$ while $(y < x+1)$ do $(c_3; c_4)$ end, $\sigma[x \to 12, y \to 5, z \to 1]\rangle$
	$\Diamond \succ \langle$ if $(y < x+1)$ then $((c_3; c_4);$ (while $(y < x+1)$ do $(c_3; c_4)$ end)), $\sigma[x \to 12, y \to 5, z \to 1]\rangle$
	$\Diamond \succ \langle$ if $(5 < x+1)$ then $((c_3; c_4);$ (while $(y < x+1)$ do $(c_3; c_4)$ end)), $\sigma[x \to 12, y \to 5, z \to 1]\rangle$
	$\Diamond \succ \langle$ if $(5 < 12+1)$ then $((c_3; c_4);$ (while $(y < x+1)$ do $(c_3; c_4)$ end)), $\sigma[x \to 12, y \to 5, z \to 1]\rangle$
	$\Diamond \succ \langle$ if $(5 < 13)$ then $((c_3; c_4);$ (while $(y < x+1)$ do $(c_3; c_4)$ end)), $\sigma[x \to 12, y \to 5, z \to 1]\rangle$
	$\Diamond \succ \langle$ if $(\top)$ then $((c_3; c_4);$ (while $(y < x+1)$ do $(c_3; c_4)$ end)), $\sigma[x \to 12, y \to 5, z \to 1]\rangle$
	$\Diamond \succ \langle (c_3; c_4);$ (while $(y < x+1)$ do $(c_3; c_4)$ end), $\sigma[x \to 12, y \to 5, z \to 1]\rangle$
sec($\square$, (while($y < x+1$) do $(c_3; c_4)$ end))	$\Diamond \succ \langle (c_3, c_4), \sigma[x \to 12, y \to 5, z \to 1]\rangle$
sec($\square, c_4$); sec($\square$, (while($y < x+1$) do $(c_3; c_4)$ end))	$\Diamond \succ \langle c_3, \sigma[x \to 12, y \to 5, z \to 1]\rangle$
sec($\square, c_4$); sec($\square$, (while($y < x+1$) do $(c_3; c_4)$ end))	$\Diamond \succ \langle z := z+1, \sigma[x \to 12, y \to 5, z \to 1]\rangle$
sec($\square, c_4$); sec($\square$, (while($y < x+1$) do $(c_3; c_4)$ end))	$\Diamond \succ \langle \sigma[x \to 12, y \to 5, z \to 2]\rangle$
sec($\square$, (while($y < x+1$) do $(c_3; c_4)$ end))	$\Diamond \succ \langle c_4, \sigma[x \to 12, y \to 5, z \to 2]\rangle$
sec($\square$, (while($y < x+1$) do $(c_3; c_4)$ end))	$\Diamond \succ \langle x := x-y, \sigma[x \to 12, y \to 5, z \to 2]\rangle$
sec($\square$, (while($y < x+1$) do $(c_3; c_4)$ end))	$\Diamond \prec \langle \sigma[x \to 7, y \to 5, z \to 2]\rangle$

- Iteración 3:

	$\Diamond \succ \langle$ while $(y < x+1)$ do $(c_3; c_4)$ end, $\sigma[x \to 7, y \to 5, z \to 2]\rangle$
	$\Diamond \succ \langle$ if $(y < x+1)$ then $((c_3; c_4);$ (while $(y < x+1)$ do $(c_3; c_4)$ end)), $\sigma[x \to 7, y \to 5, z \to 2]\rangle$
	$\Diamond \succ \langle$ if $(5 < x+1)$ then $((c_3; c_4);$ (while $(y < x+1)$ do $(c_3; c_4)$ end)), $\sigma[x \to 7, y \to 5, z \to 2]\rangle$
	$\Diamond \succ \langle$ if $(5 < 7+1)$ then $((c_3; c_4);$ (while $(y < x+1)$ do $(c_3; c_4)$ end)), $\sigma[x \to 7, y \to 5, z \to 2]\rangle$
	$\Diamond \succ \langle$ if $(5 < 8)$ then $((c_3; c_4);$ (while $(y < x+1)$ do $(c_3; c_4)$ end)), $\sigma[x \to 7, y \to 5, z \to 2]\rangle$
	$\Diamond \succ \langle$ if $(\top)$ then $((c_3; c_4);$ (while $(y < x+1)$ do $(c_3; c_4)$ end)), $\sigma[x \to 7, y \to 5, z \to 2]\rangle$
	$\Diamond \succ \langle (c_3; c_4);$ (while $(y < x+1)$ do $(c_3; c_4)$ end), $\sigma[x \to 7, y \to 5, z \to 2]\rangle$
sec($\square$, (while($y < x+1$) do $(c_3; c_4)$ end))	$\Diamond \succ \langle (c_3, c_4), \sigma[x \to 7, y \to 5, z \to 2]\rangle$
sec($\square, c_4$); sec($\square$, (while($y < x+1$) do $(c_3; c_4)$ end))	$\Diamond \succ \langle c_3, \sigma[x \to 7, y \to 5, z \to 2]\rangle$
sec($\square, c_4$); sec($\square$, (while($y < x+1$) do $(c_3; c_4)$ end))	$\Diamond \succ \langle z := z+1, \sigma[x \to 7, y \to 5, z \to 2]\rangle$
sec($\square, c_4$); sec($\square$, (while($y < x+1$) do $(c_3; c_4)$ end))	$\Diamond \succ \langle \sigma[x \to 7, y \to 5, z \to 3]\rangle$

sec($\square$, (while($y < x + 1$) do $(c_3; c_4)$ end)) $\lozenge \succ \langle c_4, \sigma[x \to 7, y \to 5, z \to 3] \rangle$
sec($\square$, (while($y < x + 1$) do $(c_3; c_4)$ end)) $\lozenge \succ \langle x := x - y, \sigma[x \to 7, y \to 5, z \to 3] \rangle$
sec($\square$, (while($y < x + 1$) do $(c_3; c_4)$ end)) $\lozenge \prec \langle \sigma[x \to 2, y \to 5, z \to 3] \rangle$

- Iteración 4:

$\lozenge \succ \langle$ while $(y < x + 1)$ do $(c_3; c_4)$ end, $\sigma[x \to 2, y \to 5, z \to 3] \rangle$
$\lozenge \succ \langle$ if $(y < x + 1)$ then $((c_3; c_4);$ (while $(y < x + 1)$ do $(c_3; c_4)$ end)), $\sigma[x \to 2, y \to 5, z \to 3] \rangle$
$\lozenge \succ \langle$ if $(5 < x + 1)$ then $((c_3; c_4);$ (while $(y < x + 1)$ do $(c_3; c_4)$ end)), $\sigma[x \to 2, y \to 5, z \to 3] \rangle$
$\lozenge \succ \langle$ if $(5 < 2 + 1)$ then $((c_3; c_4);$ (while $(y < x + 1)$ do $(c_3; c_4)$ end)), $\sigma[x \to 2, y \to 5, z \to 3] \rangle$
$\lozenge \succ \langle$ if $(5 < 3)$ then $((c_3; c_4);$ (while $(y < x + 1)$ do $(c_3; c_4)$ end)), $\sigma[x \to 2, y \to 5, z \to 3] \rangle$
$\lozenge \succ \langle$ if $(\bot)$ then skip, $\sigma[x \to 2, y \to 5, z \to 3] \rangle$
$\lozenge \succ \langle$ skip, $\sigma[x \to 2, y \to 5, z \to 3] \rangle$
$\lozenge \prec \sigma[x \to 2, y \to 5, z \to 3] \rangle$

$\therefore$ El estado resultante de la evaluación es en el que $\sigma(x) = 2$ y $\sigma(y) = 5$

b) Da un estado σ tal que si se evalúa el programa anterior con dicho estado la evaluación se ciclaría infinitamente.

Consideremos un ejemplo en el que y es inicialmente negativo. Supongamos el siguiente estado inicial:
$\sigma(x) = 8$ $\sigma(y) = -2$
Ahora bien, notemos que al evaluar la condición del bucle $y < x + 1$ en este estado, obtenemos $-2 < 8 + 1$, que es $\top$. Por lo tanto, el bucle se ejecutará, y las instrucciones dentro del bucle modificarán z y x, pero no cambiarán y (de hecho sin impotar que, y nunca cambia).

Después de cada iteración del bucle, la condición seguirá siendo verdadera, ya que y nunca cambia. La instrucción $x := x - y$ disminuirá el valor de x en cada iteración, pero como y es negativo, la resta en realidad aumentará el valor absoluto de x.

La ejecución del programa continuará en un bucle infinito, ya que la condición siempre será verdadera y el valor de x seguirá aumentando en términos absolutos debido a la resta con un valor negativo, que finalmente termina siendo una suma.

2. Extiende el lenguaje While con el operador:

$$\texttt{for}\ x := a_1\ \texttt{to}\ a_2\ \texttt{do}\ c$$

a) Modifica la estructura de la máquina $\mathcal{W}$ (agregando marcos, estados o transiciones) para evaluar la expresión `for`.

$$\frac{}{P \succ \langle \text{ for } x := a_1 \text{ to } a_2 \text{ do } c, \sigma \rangle \to_w P \succ \langle\ x := a_1;\ \text{while } x < a_2 + 1 \text{ do } c;\ x := x + 1 \text{ end },\sigma \rangle}$$

b) Da las reglas de semántica estática para verificación de tipos para el nuevo operador `for`.

$$\frac{\Gamma \vdash a_1 : \text{Int} \quad \Gamma \vdash a_2 : \text{Int} \quad \Gamma \vdash c : \text{Void}}{\Gamma \vdash \text{for x} := a_1 \text{ to } a_2 \text{ do c : Void}}$$

c) ¿Es posible definir el operador `for` como **azúcar sintáctica** dentro del lenguaje While? Justifica tu respuesta.
El `for` es azúcar sintáctica ya que podríamos verlo en el lenguaje While como un ciclo while con una variable contador que se vaya reduciendo con cada iteración.

3. Decimos que dos programas en el lenguaje While son equivalentes ($c_1 \equiv_w c_2$) ejecución de ambos programas resulta en el mismo estado, es decir, si para todo estado de las variable σ, $\Diamond \succ \langle c1, \sigma\rangle \to^*_{\mathcal{W}} \Diamond \succ \sigma'$ y $\Diamond \succ \langle c2, \sigma\rangle \to^*_{\mathcal{W}} \Diamond \succ \sigma'$ entonces $c_1 \equiv_w c_2$.
Con la definición de equivalencia anterior, demuestra o da un contraejemplo de lo siguiente:
Supongamos que While es determinista

a) $\equiv_W$ realmente es una relación de equivalencia. Esto es, demuestra que la relación $\equiv_W$ es transitiva, reflexiva y simétrica.

Suponiendo $c1$, $c2$ y $c3$ cualquiera, ademas σ cualquiera

- Reflexiva
Trivialmente se cumple ya que si $\Diamond \succ \langle c1, \sigma\rangle \to^*_W \Diamond \prec \sigma'$, entonces tenemos que $\Diamond \succ \langle c1, \sigma\rangle \to^*_W \Diamond \prec \sigma'$ y $\Diamond \succ \langle c1, \sigma\rangle \to^*_W \Diamond \prec \sigma'$, entonces se cumple $c1 \equiv_W c1$ por def de $\equiv_W$

- Simetrica
Supongamos que $c1 \equiv_W c2$, por def de $\equiv_W$ tenemos que $\Diamond \succ \langle c1, \sigma\rangle \to^*_W \Diamond \prec \sigma'$ y $\Diamond \succ \langle c2, \sigma\rangle \to^*_W \Diamond \prec \sigma'$, pero en particular $\Diamond \succ \langle c2, \sigma\rangle \to^*_W \Diamond \prec \sigma'$ y $\Diamond \succ \langle c1, \sigma\rangle \to^*_W \Diamond \prec \sigma'$, entonces se cumple $c2 \equiv_W c1$ por def de $\equiv_W$

- Transitiva
Supongamos que $c1 \equiv_W c2$ y $c2 \equiv_W c3$, por def de $\equiv_W$ tenemos que $\Diamond \succ \langle c1, \sigma\rangle \to^*_W \Diamond \prec \sigma'$ y $\Diamond \succ \langle c2, \sigma\rangle \to^*_W \Diamond \prec \sigma'$ y tambien que $\Diamond \succ \langle c2, \sigma\rangle \to^*_W \Diamond \prec \sigma'$ y $\Diamond \succ \langle c3, \sigma\rangle \to^*_W \Diamond \prec \sigma'$, pero en particular $\Diamond \succ \langle c1, \sigma\rangle \to^*_W \Diamond \prec \sigma'$ y $\Diamond \succ \langle c3, \sigma\rangle \to^*_W \Diamond \prec \sigma'$, entonces se cumple $c1 \equiv_W c3$ por def de $\equiv_W$

Por lo tanto $\equiv_W$ si es de equivalencia $\Box$

b) $c; skip \equiv_W c$

Suponiendo c cualquiera y σ cualquiera
Supongamos que $\Diamond \succ \langle c, \sigma\rangle \to^*_W \Diamond \prec \sigma'$
Entonces notemos que $\Diamond \succ \langle skip, \sigma'\rangle \to_W \Diamond \prec \sigma'$, por lo tanto
$\Diamond \succ \langle c; skip, \sigma\rangle \to_W secu(\Box, skip); \Diamond \succ \langle c, \sigma\rangle$
$secu(\Box, skip); \Diamond \succ \langle c, \sigma\rangle \to^*_W secu(\Box, skip); \Diamond \prec \sigma'$
$secu(\Box, skip); \Diamond \prec \sigma' \to_W \Diamond \succ \langle skip, \sigma'\rangle$
$\Diamond \succ \langle skip, \sigma'\rangle \to_W \Diamond \prec \sigma'$
Entonces $\Diamond \succ \langle c; skip, \sigma\rangle \to^*_W \Diamond \prec \sigma'$
Entonces se cumple $c; skip \equiv_W c$ por def de $\equiv_W$ $\Box$

c) $c_1; c_2 \equiv_W c_2; c_1$

Contraejemplo
Digamos que $c1$ es $x := 5$ y $c2$ es $x := 6$, además σ cualquiera
Entonces $\Diamond \succ \langle c1; c2, \sigma\rangle \to_W secu(\Box, c2); \Diamond \succ \langle c1, \sigma\rangle$
$secu(\Box, c2); \Diamond \succ \langle c1, \sigma\rangle \to_W secu(\Box, c2); \Diamond \prec \sigma'$, con $\sigma' = [x \mapsto 5]$
$secu(\Box, c2); \Diamond \prec \sigma' \to_W \Diamond \succ \langle c2, \sigma'\rangle$
$\Diamond \succ \langle c2, \sigma'\rangle \to_W \Diamond \prec \sigma''$, con $\sigma'' = [x \mapsto 6]$
Por lo tanto $\Diamond \succ \langle c1; c2, \sigma\rangle \to^*_W \Diamond \prec \sigma''$
Pero $\Diamond \succ \langle c2; c1, \sigma\rangle \to_W secu(\Box, c1); \Diamond \succ \langle c2, \sigma\rangle$

$secu(\Box, c1); \lozenge \succ \langle c2, \sigma\rangle \rightarrow_W secu(\Box, c1); \lozenge \prec \sigma''$
$secu(\Box, c1); \lozenge \prec \sigma'' \rightarrow_W \lozenge \succ \langle c1, \sigma''\rangle$
$\lozenge \succ \langle c1, \sigma' 0\rangle \rightarrow_W \lozenge \prec \sigma'$
Por lo tanto $\lozenge \succ \langle c2; c1, \sigma\rangle \rightarrow^*_W \lozenge \prec \sigma'$
Y como $\sigma' \neq \sigma''$, concluimos que $c1; c2 \equiv_W c2; c2$ no se cumple

d) $c_1; (c_2; c_3) \equiv_W (c_1; c_2); c_3$

Suponiendo $c1$, $c2$ y $c3$, y tambien σ cualquiera
Supongamos que $\lozenge \succ \langle c1, \sigma\rangle \rightarrow^*_W \lozenge \prec \sigma'$, $\lozenge \succ \langle c2, \sigma'\rangle \rightarrow^*_W \lozenge \prec \sigma''$ y $\lozenge \succ \langle c3, \sigma''\rangle \rightarrow^*_W \lozenge \prec \sigma'''$
Entonces $\lozenge \succ \langle c1; (c2; c3), \sigma\rangle \rightarrow_W secu(\Box, c2; c3); \lozenge \succ \langle c1, \sigma\rangle$
$secu(\Box, c2; c3); \lozenge \succ \langle c1, \sigma\rangle \rightarrow^*_W secu(\Box, c2; c3); \lozenge \prec \sigma'$
$secu(\Box, c2; c3); \lozenge \prec \sigma' \rightarrow_W \lozenge \succ \langle c2; c3, \sigma'\rangle$
$\lozenge \succ \langle c2; c3, \sigma'\rangle \rightarrow_W secu(\Box, c3); \lozenge \succ \langle c2, \sigma'\rangle$
$secu(\Box, c3); \lozenge \succ \langle c2, \sigma'\rangle \rightarrow^*_W secu(\Box, c3); \lozenge \prec \sigma''$
$secu(\Box, c3); \lozenge \prec \sigma'' \rightarrow_W \lozenge \succ \langle c3, \sigma''\rangle$
$\lozenge \succ \langle c3, \sigma''\rangle \rightarrow^*_W \lozenge \prec \sigma'''$
Por lo tanto $\lozenge \succ \langle c1; (c2; c3), \sigma\rangle \rightarrow^*_W \lozenge \prec \sigma'''$
Pero $\lozenge \succ \langle (c1; c2); c3, \sigma\rangle \rightarrow_W secu(\Box, c3); \lozenge \succ \langle c1; c2, \sigma\rangle$
$secu(\Box, c3); \lozenge \succ \langle c1; c2, \sigma\rangle \rightarrow_W secu(\Box, c2); secu(\Box, c3); \lozenge \succ \langle c1, \sigma\rangle$
$secu(\Box, c2); secu(\Box, c3); \lozenge \succ \langle c1, \sigma\rangle \rightarrow^*_W secu(\Box, c2); secu(\Box, c3); \lozenge \prec \sigma'$
$secu(\Box, c2); secu(\Box, c3); \lozenge \prec \sigma' \rightarrow_W secu(\Box, c3); \lozenge \succ \langle c2, \sigma'\rangle$
$secu(\Box, c3); \lozenge \succ \langle c2, \sigma'\rangle \rightarrow^*_W secu(\Box, c3); \lozenge \prec \sigma''$
$secu(\Box, c3); \lozenge \prec \sigma'' \rightarrow_W \lozenge \succ \langle c3, \sigma''\rangle$
$\lozenge \succ \langle c3, \sigma''\rangle \rightarrow^*_W \lozenge \prec \sigma'''$
Por lo tanto $\lozenge \succ \langle (c1; c2); c3, \sigma\rangle \rightarrow^*_W \lozenge \prec \sigma'''$
Entonces se cumple $c1; (c2; c3) \equiv_W (c1; c2); c3$ por def de $\equiv_W$ $\Box$