

Hoja-2-Resuelta.pdf



DEYORS



Teoria de la Programacion



4º Grado en Matemáticas



Facultad de Ciencias Matemáticas
Universidad Complutense de Madrid

La vida es más bonita con dinero extra para caprichos, por eso te damos **5€ de bienvenida al abrir tu Cuenta NoCuenta de ING.**

Aquí los tienes



LA DE ESTUDIAR, AHORA
MISMO TE LA SABES.
LA DE TRABAJAR, VAMOS
A VERLO.



InfoJobs

O

T PRO

HOJA 2

21/03/21.

(2.3) Considerese: $z := 0$; while $y \leq x$ do ($z := z + 1$; $x := x - y$)

Construye un árbol de denivación ejecutado desde un estado

$$S[x \mapsto 17, y \mapsto 5] = S_{1752}$$

$$\textcircled{*} S_{1750} = S_{1752}[z \mapsto 0].$$

$$\frac{\frac{\frac{< z := 0; S_{1750} > \rightarrow S_{1750}, < \text{while } y \leq x \text{ do } (z := z + 1; x := x - y), S_{1750} > \rightarrow S_{253}}{< z := 0; \text{while } y \leq x \text{ do } (z := z + 1; x := x - y), S_{1752} > \rightarrow S_{253}}}{\text{dónde } T_1 : \frac{< z := z + 1, S_{1750} > \rightarrow S_{1751}, < x := x - y, S_{1751} > \rightarrow S_{1251}}{< z := z + 1; x := x - y, S_{1750} > \rightarrow S_{1251}}}$$

$$\text{dónde } T_2 : \frac{< z := z + 1, S_{1251} > \rightarrow S_{1252}, < x := x - y, S_{1252} > \rightarrow S_{752}}{< z := z + 1; x := x - y, S_{1251} > \rightarrow S_{752}}, T_3$$

$$\text{dónde } T_3 : \frac{< \text{while } y \leq x \text{ do } (z := z + 1; x := x - y), S_{1251} > \rightarrow S_{253}}{< z := z + 1, S_{752} > \rightarrow S_{753}, < x := x - y, S_{753} > \rightarrow S_{253}}, T_4$$

$$\text{dónde } T_4 : \frac{< z := z + 1; x := x - y, S_{752} > \rightarrow S_{253}}{< \text{while } y \leq x \text{ do } (z := z + 1; x := x - y), S_{752} > \rightarrow S_{253}}$$

$$\text{dónde } T_4 : < \text{while } y \leq x \text{ do } (z := z + 1; x := x - y), S_{253} > \rightarrow S_{253}$$

y donde:

$$S_{1752} = S[x \mapsto 17, y \mapsto 5].$$

$$S_{1750} = S[x \mapsto 17, y \mapsto 5, z \mapsto 0].$$

$$S_{1751} = S[x \mapsto 17, y \mapsto 5, z \mapsto 1].$$

$$S_{1251} = S[x \mapsto 12, y \mapsto 5, z \mapsto 1].$$

$$S_{1252} = S[x \mapsto 12, y \mapsto 5, z \mapsto 2].$$

$$S_{ijk} = S[x \mapsto i, y \mapsto j, z \mapsto k].$$

$$i, j, k \in \mathbb{N}.$$

Tenemos + 13.000 ofertas
de trabajo en Madrid para tí.
Un besito.



WUOLAH

2.4 Considerando los siguientes statements, determina para todos estos si terminan siempre no, o si se quedan en bucle siempre o no.

- while $\neg(x=1)$ do ($y := y * x$; $x := x - 1$)

Sea $s \in \text{State}$, desarrollemos el árbol de denivación de la expresión. Al ser while:

$\neg(x=1) \Leftrightarrow s_x = 1$
Si $\mathcal{B}[\neg(x=1), s] = \text{ff}$, la regla que habrá que aplicar será $[\text{while}_{\text{ns}}^{\text{ff}}]$ y al ser así acaba terminando siempre.

Si $\mathcal{B}[\neg(x=1), s] = \text{tt}$, aplicando $[\text{while}_{\text{ns}}^{\text{tt}}]$:

$$\frac{\begin{array}{c} \langle y := y * x, s \rangle \rightarrow S_{(sx)(sy*x)}, \langle x := x - 1, S_{(sx)(sy*x)} \rangle \rightarrow S_{(sx-1)(sy*x)} \\ \langle y := y * x; x := x - 1, s \rangle \rightarrow S_{(sx-1)(sy*x)} \end{array}}{\langle \text{while } \neg(x=1) \text{ do } (y := y * x; x := x - 1), s \rangle \rightarrow S}, T_1$$

donde T_1 tendrá como raíz: $\langle \text{while } \neg(x=1) \text{ do } (y := y * x; x := x - 1), S_{(sx-1)(sy*x)} \rangle \rightarrow s''$.

- Si $s_x > 1$, una rama del árbol de T_1 tendrá como raíz: $\langle \text{while } \neg(x=1) \text{ do } (y := y * x; x := x - 1), S_{(1)(sy_n)} \rangle \rightarrow s^*$, con $s^* \in \text{State}$, $y_n \in \text{Var}$. Lo que llevaría aplicar $[\text{while}_{\text{ns}}^{\text{ff}}]$ y terminando siempre.

- Si $s_x \leq 1$, todos los ramos de T_1 tendrán aplicadas la propiedad $[\text{while}_{\text{ns}}^{\text{tt}}]$, porque para todos los estados intermedios s'' se verificará que $\mathcal{B}[\neg(x=1), s''] = \text{tt}$, luego la ejecución nunca terminará.

- while $1 \leq x$ do ($y := y * x$; $x := x - 1$) Análogo al anterior, sólo que si $s_x > 1$ acaba siempre y si $s_x \leq 1$ no acaba nunca.

- while true do skip

Siempre se verificará $\mathcal{B}[\text{true}, s] = \text{tt}$, luego todos los ramos tendrán aplicadas las reglas $[\text{while}_{\text{ns}}^{\text{tt}}]$ y $[\text{skip}_{\text{ns}}]$, por tanto:

$$\frac{\langle \text{skip}, s \rangle \rightarrow S, T_2}{\langle \text{while true do skip}, s \rangle \rightarrow S'}$$

con T_2 árbol de denivación idéntico al árbol desde la base, es decir, con su misma longitud, y por tanto sería un bucle para todo $s \in \text{state}$ (infinito).

Hazte una **Cuenta NoCuenta**
de ING y te llevas **5€ por la
cara...**

para que no estés más tieso
que un turrón caducado.

Usa el código [WUOLAH5](#)



[Quiero 5€](#)



Teoría de la Programación



Comparte estos flyers en tu clase y consigue más dinero y recompensas

- 1 Imprime esta hoja
- 2 Recorta por la mitad
- 3 Coloca en un lugar visible para que tus compis puedan escanear y acceder a apuntes
- 4 Llévate dinero por cada descarga de los documentos descargados a través de tu QR



Banco de apuntes de la

WUOLAH



HOJA 2

(2.6) Prueba que $S_1; (S_2; S_3)$ y $(S_1; S_2); S_3$ son semánticamente equivalentes.

Sean $s, s' \in \text{State}$:

\Rightarrow Supongamos que $\overbrace{\langle S_1; (S_2; S_3), s \rangle \rightarrow s'}^{(*)}$, demostremos que $\langle (S_1; S_2); S_3, s \rangle \rightarrow s'$.

Construyendo un árbol para $(*)$:

$$\frac{\langle S_2, s_2 \rangle \rightarrow s_3, \langle S_3, s_3 \rangle \rightarrow s'}{\langle S_1, s \rangle \rightarrow s_2 \quad \langle (S_2; S_3), s_2 \rangle \rightarrow s'}$$

$$\frac{}{\langle S_1; (S_2; S_3), s \rangle \rightarrow s'}$$

Supongamos que $\overbrace{\langle (S_1; S_2); S_3, s \rangle \rightarrow s''}^{(**)}$. Demostremos que $s'' = s'$.

Construyendo un árbol para $(**)$:

$$\frac{\begin{array}{l} \langle S_1, s \rangle \rightarrow S_2^*, \langle S_2, S_2^* \rangle \rightarrow S_3^* \\ \langle (S_1; S_2), s \rangle \rightarrow S_3^*, \langle S_3, S_3^* \rangle \rightarrow s'' \end{array}}{\langle (S_1; S_2); S_3, s \rangle \rightarrow s''}$$

Como $\langle S_1, s \rangle \rightarrow S_2$ y $\langle S_1, s \rangle \rightarrow S_2^* \Rightarrow \underline{S_2 = S_2^*}$.

Como $\langle S_2, S_2 \rangle \rightarrow S_3$, $\langle S_2, S_2^* \rangle \rightarrow S_3^*$ y $S_2^* = S_2 \Rightarrow S_3 = S_3^*$

Como $\langle S_3, S_3 \rangle \rightarrow S'$, $\langle S_3, S_3^* \rangle \rightarrow S''$ y $S_3^* = S_3 \Rightarrow \boxed{S'' = S'}$ #

\Leftarrow Totalmente análogo.

(2.7) Extiende el lenguaje While con el statement
repeat S until b.

Luego prueba que repeat S' until b es semánticamente equivalente a
S; if b then skip else (repeat S until b)

$$[\text{repeat}_{ns}^{ff}] \quad \frac{\langle S, s \rangle \rightarrow S^*, \langle \text{repeat } S \text{ until } b, S^* \rangle \rightarrow S'}{\langle \text{repeat } S \text{ until } b, s \rangle \rightarrow S'} \quad \text{if } \beta[b] = ff$$

$$[\text{repeat}_{ns}^{tt}] \quad \frac{\langle S, s \rangle \rightarrow S'}{\langle \text{repeat } S \text{ until } b, s \rangle \rightarrow S'} \quad \text{if } \beta[b] = tt$$

Sea $s, s' \in \text{State}$:

$\langle \text{repeat } S \text{ until } b, s \rangle \rightarrow S'$ si y solo si $\langle S; \text{if } b \text{ then skip else (repeat } S \text{ until } b), s \rangle \rightarrow S'$?

\Rightarrow Supongamos que $\langle \text{repeat } S \text{ until } b, s \rangle \rightarrow S'$,
y supongamos que $\langle S; \text{if } b \text{ then skip else (repeat } S \text{ until } b), s \rangle \rightarrow S''$

Demostremos que $S' = S''$.

Construyendo el árbol de (*):

T_L.

$$[\text{comps}_s] \quad \frac{\langle S, s \rangle \rightarrow S_1, \langle \text{if } b \text{ then skip else (repeat } S \text{ until } b), S_1 \rangle \rightarrow S''}{\langle S; \text{if } b \text{ then skip else (repeat } S \text{ until } b), s \rangle \rightarrow S''}$$

• Si $\beta[b] = tt$; entonces:

$$T_L = [\text{if}_{ns}^{tt}] : \langle \text{skip}, S_1 \rangle \rightarrow S_1 \quad [\text{skip}_{ns}], \text{ luego } S_1 = S''$$

Construyendo el árbol de (**):

$$[\text{repeat}_{ns}^{tt}] \quad \frac{\langle S, s \rangle \rightarrow S'}{\langle \text{repeat } S \text{ until } b, s \rangle \rightarrow S'}$$

Como $\langle S, s \rangle \rightarrow S'$; $\langle S, s \rangle \rightarrow S_1$

$$\text{y } S_1 = S'' \rightarrow \boxed{S' = S''}$$

**“SOY CREW DE McDONALD'S,
Y POR SUPUESTO QUE
MI TRABAJO Y MI PASIÓN
SON COMPATIBLES”**



My CREW
Mi trabajo. Mi pasión. Mi gente.

07/03/21 (2)

PRO

- Si $\mathcal{B}[b] = \text{ff}$, entonces:

T_1 :

$\boxed{\text{if } b}$

$$\frac{T_2}{\langle \text{repeat } S \text{ until } b, s_1 \rangle \rightarrow s''}$$

Construyendo el árbol de (**):

$$\frac{\begin{array}{l} \text{[repeat } b \\ \text{until } s_1 \end{array}}{\frac{\langle S, S \rangle \rightarrow S_1^*, \langle \text{repeat } S \text{ until } b, S_1^* \rangle \rightarrow S'}{\langle \text{repeat } S \text{ until } b, s \rangle \rightarrow S'}}$$

$$\text{Como } \langle S, s \rangle \rightarrow S_1^* \text{ y } \langle S, s \rangle \rightarrow S_1 \Rightarrow S_1 = S_1^*.$$

Como $\langle \text{repeat } S \text{ until } b, S_1^* \rangle \rightarrow S'$, $\langle \text{repeat } S \text{ until } b, S_1 \rangle \rightarrow S''$ y $S_1^* = S_1$,

$$\text{entonces } \boxed{S' = S''} \quad \checkmark$$

\Leftrightarrow Totalmente análogo.

(2.8) Exprende el lenguaje While con el statement:

~~for $x := a_1$ to a_2 do S .~~

Luego evalúa el statement:

$y := 1$; for $z := 1$ to x do ($y := y * x$; $x := x - 1$) desde

un estado donde x tiene valor 5.

$\boxed{\text{for } tt}$

$$\frac{\langle x := a_1, s \rangle \rightarrow S_1, \langle S, S_1 \rangle \rightarrow S_2, \langle \text{for } x := x+1 \text{ to } a_2 \text{ do } S, S_2 \rangle \rightarrow S'}{\langle \text{for } x := a_1 \text{ to } a_2 \text{ do } S, s \rangle \rightarrow S'}$$

if $\mathcal{B}[x \leq a_2] = tt$

$\boxed{\text{for } ff}$

$$\frac{\langle x := a_1, s \rangle \rightarrow S'}{\langle \text{for } x := a_1 \text{ to } a_2 \text{ do } S, s \rangle \rightarrow S'} \quad \text{if } \mathcal{B}[x \leq a_2] = ff.$$

My CREW
Mi trabajo. Mi pasión. Mi gente.

¿TE VIENES?



WUOLAH

Construimos símbolos de $y := 1$; for $z := 1$ to x do ($y := y * x$; $x := x - 1$)

desde $S = [x \mapsto 1]$. (Para que no haya tantos pasos)

[comps]
+[assigns]

$$\frac{<y := 1, S> \rightarrow S[y \mapsto 1], \quad T_0}{<y := 1; \text{for } z := 1 \text{ to } x \text{ do } (y := y * x; x := x - 1), S> \rightarrow S_{002}}$$

$T_0:$
[for_{ns}^{tt}]

$$\frac{<z := 1, S[y \mapsto 1]> \rightarrow S[y \mapsto 1][z \mapsto 1], < T_1 \quad T_2, \quad T_2>}{<\text{for } z := 1 \text{ to } x \text{ do } (y := y * x; x := x - 1), S[y \mapsto 1]> \rightarrow S_{002}}$$

$T_1:$
[comps]
+[assigns]

$$\frac{<y := y * x, S_{111}> \rightarrow S_{111}, <x := x - 1, S_{111}> \rightarrow S_{011}}{<(y := y * x; x := x - 1), S_{111}> \rightarrow S_{011}}$$

$T_2:$
[for_{ns}^{tt}]

$$\frac{<z := z + 1, S_{011}> \rightarrow S_{012}}{<\text{for } z := z + 1 \text{ to } x \text{ do } (y := y * x; x := x - 1), S_{011}> \rightarrow S_{012}}$$

El estado final es $S_{012} = S[x \mapsto 0, y \mapsto 1, z \mapsto 2]$.

Suponiendo que $S_{n_1 n_2 n_3} = S[x \mapsto n_1, y \mapsto n_2, z \mapsto n_3]$.

(2.10) Prueba que repeat S until b es semánticamente equivalente a S ; while $\neg b$ do S . Razona que esto significa que la semántica extendida es determinista.

En el ej. 2.7 se define la regla para repeat S until b :

$$[\text{rep}_{ns}^{tt}]: \frac{\langle S, s \rangle \rightarrow S'}{\langle \text{repeat } S \text{ until } b, s \rangle \rightarrow S'} \quad \text{si } \mathcal{B}[b] = \text{tt}$$

$$[\text{rep}_{ns}^{ff}]: \frac{\langle S, s \rangle \rightarrow S', \langle \text{repeat } S \text{ until } b, s' \rangle \rightarrow S''}{\langle \text{repeat } S \text{ until } b, s \rangle \rightarrow S''} \quad \text{si } \mathcal{B}[b] = \text{ff}$$

Sea $s, s' \in \text{State}$, hay que demostrar que:

$$(\langle \text{repeat } S \text{ until } b, s \rangle \rightarrow S') \Leftrightarrow (\langle S; \text{while } \neg b \text{ do } S, s \rangle \rightarrow S')$$

Usaremos inducción sobre la longitud del árbol de derivación:

\Rightarrow I Suponemos que $\langle \text{repeat } S \text{ until } b, s \rangle \rightarrow S'$, luego para construir su árbol de derivación son posibles dos reglas:

- $[\text{rep}_{ns}^{tt}]$: entonces $\mathcal{B}[b] = \text{tt}$ y el árbol será

$$\frac{\langle S, s \rangle \rightarrow S'}{\langle \text{repeat } S \text{ until } b, s \rangle \rightarrow S'}$$

Por otro lado, el árbol de derivación de $S; \text{while } \neg b \text{ do } S$ es:
utilizando [compas]:

$$\frac{\langle S, s \rangle \rightarrow S''' , \langle \text{while } \neg b \text{ do } S, s''' \rangle \xrightarrow{(*)} S'''}{\langle S; \text{while } \neg b \text{ do } S, s \rangle \rightarrow S''}$$

(*) ya que $\mathcal{B}[\neg b] = \text{ff}$. Por tanto, $S''' = S'' = S' \Rightarrow \langle S; \text{while } \neg b \text{ do } S, s \rangle \rightarrow S'$

(**) ya que $\langle S, s \rangle \rightarrow S'''$
y $\langle S, s \rangle \rightarrow S'$

- [rep_{ns}^{ff}]. Entonces $\beta[b]s = ff$, y el árbol será:

$$\frac{\langle S, s \rangle \rightarrow S'', T_1}{\langle \text{repeat } S \text{ until } b, s \rangle \rightarrow S'}$$

donde T_1 es un árbol de derivación conocida:

$$\langle \text{repeat } S \text{ until } b, s'' \rangle \rightarrow S'.$$

Como T_1 tiene menor tamaño que el árbol de inicio, podemos formular la hipótesis de inducción:

(H1) Supongamos que $\langle S; \text{while } \neg b \text{ do } S, s'' \rangle \rightarrow S'$.

Desarrollando la H1 con la regla [camps] :

$$T_2 = \frac{\langle S, s'' \rangle \rightarrow S''' \quad \langle \text{while } \neg b \text{ do } S, s''' \rangle \rightarrow S' }{\langle S; \text{while } \neg b \text{ do } S, s'' \rangle \rightarrow S'}$$

Siembargo, como $\beta[b]s = tt$, las premisas T_2 y T_3 también son las ramas de: (aplicando [while_{ns}^{tt}])

$$T_4 = \frac{T_2 \quad T_3}{\langle \text{while } \neg b \text{ do } S, s'' \rangle \rightarrow S'} \quad \text{y como también se verifica } T_5 = \langle S, s \rangle \rightarrow S'', \text{ entonces } T_4 \text{ y } T_5 \text{ también son las ramas de: (aplicando [camps])}$$

$$\frac{T_5 \quad T_4}{\langle S; \text{while } \neg b \text{ do } S, s \rangle \rightarrow S'} \quad \text{verificándose } \langle S; \text{while } \neg b \text{ do } S, s \rangle \rightarrow S' \checkmark$$

**"SOY CREW DE McDONALD'S,
Y POR SUPUESTO QUE
MI TRABAJO Y MI PASIÓN
SON COMPATIBLES"**



My CREW
Mi trabajo. Mi pasión. Mi gente.

720

HOTA 2

08.03.21 (2)

(E) Suponemos que $\langle S; \text{while } b \text{ do } S, S \rangle \rightarrow S'$, y desarrollando su árbol de derivación usando [couples]:

$$\frac{\langle S, S \rangle \rightarrow S'' , (\langle \text{while } b \text{ do } S, S'' \rangle \rightarrow S') = T_1}{\langle S; \text{while } b \text{ do } S, S \rangle \rightarrow S'}$$

Para desarrollar T_1 se pueden aplicar las reglas según $\mathcal{B}[\neg b]S$:

1) Si $\mathcal{B}[\neg b]S = tt$, entonces aplicando [while_{ns}] a T_1 :

$$T_2 = \frac{\langle S, S'' \rangle \rightarrow S''' , (\langle \text{while } b \text{ do } S, S''' \rangle \rightarrow S') = T_3}{T_1}$$

donde los premisos T_2 y T_3 también pueden conseguirse utilizando [couples] en el árbol:

$$\frac{T_2 \quad T_3}{(\langle S; \text{while } b \text{ do } S, S'' \rangle \rightarrow S') = T_4}$$

Como T_4 es la raíz de un árbol de derivación de menor tamaño que el original, podemos formar la hipótesis de inducción:

(H1) Suponemos que $\langle \text{repeat } S \text{ until } b, S'' \rangle \rightarrow S'$

Como $\mathcal{B}[\neg b]S = ff$, entonces $\mathcal{B}[b]S = ff$, y como $(\langle S, S \rangle \rightarrow S'') = T_5$, las premisas T_5 y H1 pueden ser consecuencia de la regla [repeat_{ns}]:

$$\frac{T_5 \quad H1}{\langle \text{repeat } S \text{ until } b, S \rangle \rightarrow S'} , \text{ demostrando que } \langle \text{repeat } S \text{ until } b, S \rangle \rightarrow S' \checkmark$$

My CREW
Mi trabajo. Mi pasión. Mi gente.

¿TE VIENES?



WUOLAH

2) Si $\mathcal{B}[\neg b]s = \text{ff}$, entonces aplicando $[\text{while}_n^{\text{ff}}]$ a T_2 :

$\langle \text{while } \neg b \text{ do } S, s'' \rangle \rightarrow S'$, y por tanto $S' = s''$, luego
 $\langle S, s \rangle \rightarrow S'$.

Desarrollando repeat S until b , como $\mathcal{B}[b]s = \text{tt}$, usando $[\text{repeat}_n^{\text{tt}}]$:

$$\frac{\langle S, s \rangle \rightarrow S^*}{\langle \text{repeat } S \text{ until } b, s \rangle \rightarrow S^*} \quad \text{Como } \langle S, s \rangle \rightarrow S', \text{ entonces } S' = S^*,$$

Luego $\langle \text{repeat } S \text{ until } b, s \rangle \rightarrow S' \neq \text{#}$.

¿Es repeat S until b determinista?

Para ello, se debe cumplir que $\forall s \in \text{State}, \forall S \in \text{Stm}$, si existen s' y $s'' \in \text{State}$ tales que $\langle S, s \rangle \rightarrow S'$ y $\langle S, s \rangle \rightarrow S''$, entonces $S' = S''$.

Como por el th. 2.9 todas las semánticas naturales básicas son deterministas, veamos que también lo es repeat S until b .

Sea $s, s', s'' \in \text{State}$ y supongamos que:

$$\langle \text{repeat } S \text{ until } b, s \rangle \rightarrow S' \text{ y } \langle \text{repeat } S \text{ until } b, s \rangle \rightarrow S''.$$

Como repeat S until b es semánticamente equivalente a $S; \text{while } \neg b \text{ do } S$:

$$\langle S; \text{while } \neg b \text{ do } S, s \rangle \rightarrow S' \text{ y } \langle S; \text{while } \neg b \text{ do } S, s \rangle \rightarrow S''.$$

Como esta semántica sí es determinista, entonces $S' = S''$.

Como esta semántica repeat S until b es determinista.

Por tanto, la semántica repeat S until b es determinista.

2.11) Es posible utilizar semánticas operacionales también en la función of y definir NS para expresiones aritméticas. Tendrá dos tipos de configuraciones:

$\langle a, s \rangle$ donde a debe evaluarse en el entorno s ,
 z dando como resultado el valor final.

Luego la relación de transición $\rightarrow_{\text{Aexp}}$ tendrá la forma:

$$\langle a, s \rangle \xrightarrow{\text{Aexp}} z$$

Completa la especificación del sistema y utiliza inducción en Aexp para demostrar que la especificación es análoga a la definición de of .

$$[\text{Anots}_s] \quad \langle n, s \rangle \xrightarrow{\text{Aexp}} \mathcal{N}[[n]]$$

$$[\text{Avors}_s] \quad \langle x, s \rangle \xrightarrow{\text{Aexp}} s \ x$$

$$[\text{Asum}_{ns}] \quad \frac{\langle a_1, s \rangle \xrightarrow{\text{Aexp}} z_1, \langle a_2, s \rangle \xrightarrow{\text{Aexp}} z_2}{\langle a_1 + a_2, s \rangle \xrightarrow{\text{Aexp}} z}, \text{ donde } z = z_1 + z_2$$

$$[\text{Asub}_{ns}] \quad \frac{\langle a_1, s \rangle \xrightarrow{\text{Aexp}} z_1, \langle a_2, s \rangle \xrightarrow{\text{Aexp}} z_2}{\langle a_1 - a_2, s \rangle \xrightarrow{\text{Aexp}} z}, \text{ donde } z = z_1 - z_2$$

$$[\text{Amult}_{ns}] \quad \frac{\langle a_1, s \rangle \xrightarrow{\text{Aexp}} z_1, \langle a_2, s \rangle \xrightarrow{\text{Aexp}} z_2}{\langle a_1 * a_2, s \rangle \xrightarrow{\text{Aexp}} z}, \text{ donde } z = z_1 * z_2$$

Demonstración por inducción: Sea $s \in \text{State}$

CB: Sea $n \in \text{Num}$, entonces $\langle n, s \rangle \xrightarrow{\text{Aexp}} \mathcal{N}[[n]]$ y $\text{of}[\mathcal{N}[[n]]]s = \mathcal{N}[[n]]$
 Sea $x \in \text{Var}$, análogo ✓

PI: Supongamos que $\langle a_1, s \rangle \xrightarrow{\text{Aexp}} z_1, \langle a_2, s \rangle \xrightarrow{\text{Aexp}} z_2, \text{of}[a_1]s = z_1, \text{of}[a_2]s = z_2$. Entonces

$\langle a_1 + a_2, s \rangle \xrightarrow{\text{Aexp}} z$ donde $z = z_1 + z_2$, y $\text{of}[a_1 + a_2]s = \text{of}[a_1]s + \text{of}[a_2]s = z_1 + z_2 = z$ ✓

Con los demás operadores es análogo.

(2.12) De forma similar podemos especificar NS para expresiones booleanas, con la transición $\langle b, s \rangle \xrightarrow{B_{exp}} t$, con $t \in T$. Complete la configuración y demuestre que es análogo a \mathcal{B} .

$[B_{true_{ns}}]$	$\langle \text{true}, s \rangle \xrightarrow{B_{exp}} tt$
$[B_{false_{ns}}]$	$\langle \text{false}, s \rangle \xrightarrow{B_{exp}} ff.$
$[B_{eq_{ns}^{tt}}]$	$\langle z_1 = z_2, s \rangle \xrightarrow{B_{exp}} tt \quad \text{si } z_1 = z_2, \text{ donde } \langle z_1, s \rangle \xrightarrow{A_{exp}} z_1$
$[B_{eq_{ns}^{ff}}]$	ANÁLOGO $\langle z_1 = z_2, s \rangle \xrightarrow{A_{exp}} z_2$
$[B_{design_{ns}^{tt}}]$	ANÁLOGO
$[B_{neg_{ns}^{tt}}]$	$\langle \neg b, s \rangle \xrightarrow{B_{exp}} ff \quad \text{si } \langle b, s \rangle \xrightarrow{B_{exp}} tt$
$[B_{neg_{ns}^{ff}}]$	ANÁLOGO
$[B_{and_{ns}^{tt}}]$	$\langle b_1 \wedge b_2 \rangle \xrightarrow{B_{exp}} tt \quad \text{si } \langle b_1, s \rangle \xrightarrow{B_{exp}} tt \text{ y } \langle b_2, s \rangle \xrightarrow{B_{exp}} tt$
$[B_{and_{ns}^{ff}}]$	ANÁLOGA $\langle b_1, s \rangle \xrightarrow{B_{exp}} ff \quad \text{y } \langle b_2, s \rangle \xrightarrow{B_{exp}} ff.$

Demostremos que es análogo al 2.11.

LA DE ESTUDIAR, AHORA
MISMO TE LA SABES.
LA DE TRABAJAR, VAMOS
A VERLO.

InfoJobs



O

13/04/21 (2)

PRO

E.16) Construye una secuencia de denivación por el statement.

$z := 0 ; \text{while } y \leq x \text{ do } (z := z + 1 ; x := x - y)$

Ejecutado desde un estado donde x tiene el valor de 17 e y de 5.

Luego determine un estado cuya secuencia sea infinita.

Sea s_0 state, con $s = [x \mapsto 17, y \mapsto 5]$.

$$[\text{comp}^2_{\text{sos}}] \quad \frac{\langle z := 0, s \rangle \Rightarrow^{17, 5} s_0}{\langle z := 0 ; \text{while } y \leq x \text{ do } (z := z + 1 ; x := x - y) \rangle \Rightarrow \subset T_1}$$

Donde T_1 tiene la denivación: (*) $\stackrel{17, 5}{\Rightarrow} s_0$ [while sas]

$T_1 = \langle \text{while } y \leq x \text{ do } (z := z + 1 ; x := x - y), s_0 \rangle \Rightarrow$

$\Rightarrow \langle \text{if } y \leq x \text{ then } (z := z + 1 ; x := x - y) ; \text{while } y \leq x \text{ do } (z := z + 1 ; x := x - y) \text{ else skip}, s_0 \rangle \Rightarrow$

$$\Rightarrow \frac{\langle (z := z + 1 ; x := x - y) ; \text{while } y \leq x \text{ do } (z := z + 1 ; x := x - y), s_0 \rangle \Rightarrow^{17, 5} T_2}{T_2}$$

$$[\text{ptt}_{\text{sos}}] \quad \frac{\langle z := z + 1, s_0 \rangle \Rightarrow^{17, 5} s_1}{\langle z := z + 1 ; x := x - y, s_0 \rangle \Rightarrow \langle x := x - y, s_1 \rangle}$$

$$[\text{comp}^2_{\text{sos}}] \quad \frac{\langle x := x - y, s_1 \rangle \Rightarrow^{12, 5} s_2}{\langle x := x - y ; \text{while } y \leq x \text{ do } (z := z + 1 ; x := x - y), s_1 \rangle \Rightarrow^{17, 5} T_4}$$

Donde T_3 :

$$[\text{comp}^2_{\text{sos}}] \quad \frac{\langle x := x - y ; \text{while } y \leq x \text{ do } (z := z + 1 ; x := x - y), s_1 \rangle \Rightarrow^{17, 5} s_3}{\langle x := x - y, s_1 \rangle \Rightarrow^{12, 5} s_2}$$

Donde T_4 : $\langle \text{while } y \leq x \text{ do } (z := z + 1 ; x := x - y), s_1 \rangle \Rightarrow^{17, 5} s_3$

Análogo a lo anterior

(*)

Luego el estado final es $s_3 = S[x \mapsto 2, y \mapsto 5, z \mapsto 3]$

Un estado con denivación infinita sería: y negativa y x mayor que y , por ejemplo:

$$S^* = [x \mapsto 5, y \mapsto -1]$$

Tenemos + 13.000 ofertas
de trabajo en Madrid para tí.
Un besito.



(2.17) Exienda WHILE con el statement repeat S until b en SOS.

[repeat_{sos}^{ff1}]: $\frac{<S, s> \Rightarrow S'}{<\text{repeat } S \text{ until } b, s> \Rightarrow <\text{repeat } S \text{ until } b, S'>} \text{ si } \beta[b] = ff$.

[repeat_{sos}^{ff2}]: $\frac{<S, s> \Rightarrow <S', s'>}{<\text{repeat } S \text{ until } b, s> \Rightarrow <S'; \text{repeat } S \text{ until } b, S'>} \text{ si } \beta[b] = ff$.

[repeat_{sos}^{tt}]: $<\text{repeat } S \text{ until } b, s> \Rightarrow <S, s> \text{ if } \beta[b] = tt$.

↓ también:

[repeat_{sos}]: $\text{repeat } S \text{ until } b, s \Rightarrow <S; \text{if } b \text{ then skip else (repeat } b \text{ until } S), S>$.

(2.18) Exienda WHILE con el statement for := a₁ to a₂ en SOS.

[for_{sos}]: $<\text{for } x := a_1 \text{ to } a_2 \text{ do } S, s> \Rightarrow$

$\Rightarrow <x := a_1; \text{if } x \leq a_2 \text{ then } (S; \text{for } x := x+1 \text{ to } a_2 \text{ do } S) \text{ else skip}, S>$

(2.20)

Supá que $(S_1; S_2, s) \Rightarrow^* <S_2, S'_1>$. Prueba que no necesariamente tiene que ocurrir que $<S_1, s> \Rightarrow^* S'_1$.

Para ver un contrarejemplo:

Es necesario que S_2 sea un ciclo de al menos dos vueltas, para hacer el lío:

$S_1 = \text{skip}$, $S_2 = \text{while } \neg(x=0) \text{ do } x=x-1$.

Sea $s \neq s_{x=2}$, entonces:

$(S_1, s) \Rightarrow S$.

$(S_1; S_2, s) \Rightarrow (S_2, s) \Rightarrow \dots \Rightarrow (S_2, s[x \leftarrow 1])$. Luego $(S_1; S_2, s) \Rightarrow (S_2, S')$ pero $(S_1, s) \neq S'$

#

WUOLAH

*
2.21.
*

PRUEBA QUE :

$$\boxed{\langle S_1, s \rangle \Rightarrow^k s' \quad \text{entonces} \quad \langle S_1; S_2, s \rangle \Rightarrow^k \langle S_2, s' \rangle}$$

(es decir, S_1 no es influenciado por ningún statement después de él)

DEMOSTRACIÓN: (SOBRE LA LONGITUD DEL ÁRBOL DE DERIVACIÓN)

- CB:
- Si $K=0 \rightarrow$ la premisa es FALSA \rightarrow CONSECUENCIA CIERTA ✓.
 - Si $K=1 \rightarrow \langle S_1, s \rangle \Rightarrow s' \rightarrow \langle S_1; S_2, s \rangle \Rightarrow \langle S_2, s' \rangle$ ✓
[comp²]

PI:

H1:

(*) Supongamos que se cumple para K y demostremos para $K+1$.

Asumimos que $\langle S_1, s \rangle \Rightarrow^{K+1} s'$.

Como $K+1 \geq 2 \rightarrow \langle S_1, s \rangle \Rightarrow \underbrace{\langle S_1'', s'' \rangle}_{(*)} \Rightarrow^k s'$.

Usando H1 con (**), llegamos a que $\langle S_1''; S_2, s'' \rangle \Rightarrow^k \langle S_2, s' \rangle$.

Desarrollando $\langle S_1; S_2, s \rangle$ por [comp¹]:

$$\frac{\langle S_1, s \rangle \Rightarrow \langle S_1'', s'' \rangle}{\langle S_1; S_2, s \rangle \Rightarrow \langle S_1''; S_2, s'' \rangle} \Rightarrow^k \langle S_2, s' \rangle ..$$

Luego $\langle S_1; S_2, s \rangle \Rightarrow^{K+1} \langle S_2, s' \rangle$

* 2.22 *

Demostrar que SOS es DETERMINISTA, deduciendo que existen solo UNA secuencia de derivación en $\langle S, s \rangle$ y argumentando por qué un S no puede ocurrir y borrar a la vez.

Supongamos que $\langle S, s \rangle \Rightarrow \gamma'$ y que $\langle S, s \rangle \Rightarrow \gamma''$. Demostremos que $\gamma' = \gamma''$. Según S , se le puede aplicar:

- S es $x := a$, luego por [ass_{SOS}]:

$$\langle x := a, s \rangle \Rightarrow s[x \mapsto \alpha[a]s] = \gamma'.$$

Al ser el único símbolo utilizable, si $\langle x := a, s \rangle \Rightarrow \gamma''$, entonces no queda otra que $\gamma'' = s[x \mapsto \alpha[a]s]$, luego $\gamma' = \gamma''$.

- S es skip análogo.

- S es $S_1; S_2$. Si $\langle S_1; S_2, s \rangle \Rightarrow \gamma'$, puede ser que:

1) Hayamos usado [comp¹_{SOS}]:

$$[\text{comp}^1_{\text{SOS}}]: \frac{\langle S_1, s \rangle \Rightarrow \langle S'_1, S'_1 \rangle}{\langle S_1; S_2, s \rangle \Rightarrow \langle S'_1; S'_2, S'_1 \rangle} = \gamma'.$$

Como $\langle S_1; S_2, s \rangle \Rightarrow \gamma''$, también hemos podido usar:

1.1) [comp¹_{SOS}]: Entonces $\gamma' = \gamma''$ análogo a los anteriores.

1.2) [comp²_{SOS}]: Entonces:

$$[\text{comp}^2_{\text{SOS}}]: \frac{\langle S_1, s \rangle \Rightarrow S'}{\langle S_1; S_2, s \rangle \Rightarrow \langle S_2, S' \rangle}, \text{ pero no es posible}$$

que $\langle S_1, s \rangle \Rightarrow S'$ y a la vez $\langle S_1, s \rangle \Rightarrow \langle S'_1, S'_1 \rangle$,
Estados Configuración.

y lo que una configuración no es lo mismo que un estado.

2) [comp²_{SOS}]: Análogo a los demás.

- Los demás puentes son totalmente análogos y usando los mismos razonamientos.

**“SOY CREW DE McDONALD'S,
Y POR SUPUESTO QUE
MI TRABAJO Y MI PASIÓN
SON COMPATIBLES”**



My CREW
Mi trabajo. Mi pasión. Mi gente.

PRO

14/04/21.

2.23) Demuestre que los statements siguientes son equivalentes:

- $S; skip \quad y \quad S$. Dado se State.

Demostremos que $\langle S; skip, S \rangle \Rightarrow^* S'$ si y sólo si $\langle S, S \rangle \Rightarrow^* S'$.

\Rightarrow Supongamos que $\langle S; skip, S \rangle \Rightarrow^* S'$.

Por el lema 2.19:

$\langle S, S \rangle \Rightarrow^{K_1} S''$, y $\langle skip, S'' \rangle \Rightarrow^{K_2} S'$, Aplicando [skip_{sos}] a (*):

$[skip_{sos}] \langle skip, S'' \rangle \Rightarrow S''$, luego $K_2 = 1$ y $S'' = S'$.

Como $\langle S, S \rangle \Rightarrow^{K_1} S''$ y $S'' = S'$, entonces

$\boxed{\langle S, S \rangle \Rightarrow^* S'}$

\Leftarrow Supongamos que $\langle S, S \rangle \Rightarrow^* S'$.

Por el lema 2.21:

$\langle S; skip, S \rangle \Rightarrow^* \langle skip, S' \rangle \Rightarrow S'$, luego $\boxed{\langle S; skip, S \rangle \Rightarrow^* S'}$

- while b do S e if b then (S; while b do S) else skip

\Rightarrow Supongamos que $\langle \text{while } b \text{ do } S, S \rangle \Rightarrow^* S'$.

Usando [while_{sos}]:

$\langle \text{while } b \text{ do } S, S \rangle \Rightarrow \langle \text{if } b \text{ then } (S; \text{while } b \text{ do } S) \text{ else skip}, S \rangle \Rightarrow^* S'$,

luego $\langle \text{if } b \text{ then } (S; \text{while } b \text{ do } S) \text{ else skip}, S \rangle \Rightarrow^* S'$.

\Leftarrow Supongamos que $\langle \text{if } b \text{ then } (S; \text{while } b \text{ do } S) \text{ else skip}, S \rangle \Rightarrow^* S'$.

Usando [while_{sos}]:

$\langle \text{while } b \text{ do } S, S \rangle \Rightarrow (\star\star) \Rightarrow^* S'$, luego $\langle \text{while } b \text{ do } S, S \rangle \Rightarrow^* S'$.

My CREW
Mi trabajo. Mi pasión. Mi gente.

¿TE VIENES?



WUOLAH

- $S_1 ; (S_2 ; S_3)$ y $(S_1 ; S_2) ; S_3$.

\Rightarrow Supongamos que $\langle S_1 ; (S_2 ; S_3), s \rangle \Rightarrow S'$.

Entonces, por el lema 2.19:

$\boxed{\langle S_1, s \rangle \Rightarrow^{(*)} S_1}$ y $\langle S_2 ; S_3, S_1 \rangle \Rightarrow^{(*)} S'$, y otra vez por el lema 2.19:

$\boxed{\langle S_2, S_1 \rangle \Rightarrow^{(*)} S_2}$ y $\boxed{\langle S_3, S_2 \rangle \Rightarrow^{(*)} S'}$.

Supongamos que $\langle (S_1 ; S_2) ; S_3, s \rangle \Rightarrow^{(*)} S''$. Por el lema 2.19:

$\langle (S_1 ; S_2), s \rangle \Rightarrow^{(*)} S'_1$ y $\boxed{\langle S_3, S'_1 \rangle \Rightarrow^{(*)} S''}$, y otra vez por el lema 2.19:

$\boxed{\langle S_1, s \rangle \Rightarrow^{(*)} S'_1}$ y $\boxed{\langle S_2, S'_1 \rangle \Rightarrow^{(*)} S'_2}$.

Como $\langle S_1, s \rangle \Rightarrow^{(*)} S_1$
 $\langle S_1, s \rangle \Rightarrow^{(*)} S'_1$ entonces $S_1 = S'_1$.
sos es determinista

Como $\langle S_2, S_1 \rangle \Rightarrow^{(*)} S_2$
 $\langle S_2, S'_1 \rangle \Rightarrow^{(*)} S'_2$ entonces $S_2 = S'_2$.
 $S_1 = S'_1$
sos es det.

Como $\langle S_3, S_2 \rangle \Rightarrow^{(*)} S'$
 $\langle S_3, S'_2 \rangle \Rightarrow^{(*)} S''$ entonces $\boxed{\begin{array}{c} S' = S'' \\ \hline \end{array}}$ #
 $S_2 = S'_2$
sos det.

(2.119) Dada la siguiente expresión sintáctica:

$\text{forVar } x \text{ do } S$

Se pide:

- 1) Dar una semántica operacional para dicha expresión. Se deberá ejecutar S siempre que el valor de la variable x no sea 0 y tras la ejecución de S deberá aumentar de forma automática en uno el valor de la variable x .

$$\text{[forVar}_{ns}^{tt}\text{]} \frac{\langle S, s \rangle \rightarrow s', \langle \text{forVar } (x+1) \text{ do } S, s' \rangle \rightarrow s''}{\langle \text{forVar } x \text{ do } S, s \rangle \rightarrow s''} \quad \text{si } \mathcal{R}[x]s \neq 0$$

$$\text{[forVar}_{ns}^{ff}\text{]} \quad \langle \text{forVar } x \text{ do } S, s \rangle \rightarrow s \quad \text{si } \mathcal{R}[x]s = 0.$$

- 2) Dar una expresión equivalente a dicha semántica en el lenguaje While.

La expresión equivalente será $\text{while } \neg(x=0) \text{ do } (S; x := x + 1)$

- 3) Demostrar formalmente la equivalencia propuesta en el apartado anterior:

Sean $s, s' \in \text{State}$, hay que demostrar que:

$$(\langle \text{forVar } x \text{ do } S, s \rangle \rightarrow s') \Leftrightarrow (\text{while } \neg(x=0) \text{ do } (S; x := x + 1))$$

\Rightarrow Supongamos que $\langle \text{forVar } x \text{ do } S, s \rangle \rightarrow s'$. Para construir su árbol son posibles dos reglas:

- $\text{[forVar}_{ns}^{tt}\text{]}$: Entonces $\mathcal{R}[x]s \neq 0$, y el árbol será:

$$\text{[forVar}_{ns}^{tt}\text{]} \frac{\langle S, s \rangle \rightarrow s'', \langle \text{forVar } (x+1) \text{ do } S, s'' \rangle \rightarrow s' \equiv T_1}{\langle \text{forVar } x \text{ do } S, s \rangle \rightarrow s'}$$

Como T_2 es la raíz de un árbol de tamaño menor al original, podemos usar la hipótesis de inducción:

(H1) $\langle \text{while } \neg(x=0) \text{ do } (S; x:=x+1), S'' \rangle \rightarrow S'$

Como $\text{cf}[x]s \neq 0$, entonces $\beta[\neg(x=0)]s = \text{tt}$, luego podemos usar $(\text{while}^{\text{tt}}_{\text{ns}})$

Con (H1):

$$\frac{\begin{array}{c} [\text{compos}] \quad \langle S, S'' \rangle \rightarrow S'', \langle x := x + 1, S'' \rangle \rightarrow S \\ [\text{while}^{\text{tt}}_{\text{ns}}] \quad \frac{\langle (S; x := x + 1), S'' \rangle \rightarrow S'''}{\langle \text{while } \neg(x=0) \text{ do } (S; x := x + 1), S'' \rangle \rightarrow S'} \end{array}}{T_2}$$

Donde $S' = S'' [x \mapsto \text{cf}[x+1]s'']$.

↳ ??

• $[\text{forVar}^{\text{ff}}_{\text{ns}}]$: Entonces $\text{cf}[x]s = 0$ y el símbolo es f:

$\langle \text{forVar } x \text{ do } S', s \rangle \rightarrow S'$, y por otro lado, como $\beta[\neg(x=0)]s = \text{tt}$,

aplicando $(\text{while}^{\text{ff}}_{\text{ns}})$:

$\langle \text{while } \neg(x=0) \text{ do } S; x := x + 1, s \rangle \rightarrow S$ ✓

Importante

Puedo eliminar la publi de este documento con 1 coin

¿Cómo consigo coins? → Plan Turbo: barato
→ Planes pro: más coins

pierdo
espacio



Necesito
concentración

ali ali ooooh
esto con 1 coin me
lo quito yo...

wuolah

15/03/21 (2)

pro
• Supongamos que $\langle \text{while } \neg(x=0) \text{ do } (S; x:=x+1), S \rangle \rightarrow S'$.
Construyendo su árbol, podemos usar las reglas:

• $[\text{while}_{ns}^{\{\}}]$: Entradas $\mathcal{B}[\neg(x=0)]S = \{\}$ y su árbol será:

$\langle \text{while } \neg(x=0) \text{ do } (S; x:=x+1), S \rangle \rightarrow S$
y por otro lado, como $\mathcal{A}[x]=0$, el árbol de la otra expresión, usando
 $[\text{forVar}_{ns}^{\{\}}]$ será:

$\langle \text{forVar } x \text{ do } S, S \rangle \rightarrow S \quad \checkmark$

• $[\text{while}_{ns}^{tt}]$: → Mismo problema con (II) → ??