

Hoja-4-Resuelta.pdf



DEYORS



Teoria de la Programacion



4º Grado en Matemáticas



Facultad de Ciencias Matemáticas
Universidad Complutense de Madrid

DUA LIPA · AVRIL LAVIGNE · RELS B · MANESKIN · MOCHAKK · PEARL JAM
JANELLE MONÁE · MICHAEL KIWANUKA · ARLO PARKS · BOMBA ESTÉREO
BLACK PUMAS · THE BREEDERS · NOTHING BUT THIEVES · ASHNIKKO
THE GASLIGHT ANTHEM · TOM ODELL · KEANE · TOM MORELLO
BONOBO dj set · PAUL KALKBRENNER · KENYA GRACE · LARKIN POE
SOCCER MOMMY · THE BLESSED MADONNA · CARLITA · SOFI TUKKER

2024

10-13 JULY



MADRID
#MadCool2024

AND MANY MORE...

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



InfoJobs

O

TP20

HOTA 4

29/04/21

(EJEMPLO 4.1) Considera:

PUSH-1 : FETCH-X : ADD : STORE-X

Desde un estado s con $S[X] = 3$, y desarrollalo:

$\langle \text{PUSH-1} : \text{FETCH-X} : \text{ADD} : \text{STORE-X}, \epsilon, s \rangle \triangleright$

$\triangleright \langle \text{FETCH-X} : \text{ADD} : \text{STORE-X}, 1, s \rangle \triangleright \langle \text{ADD} : \text{STORE-X}, 3 : 1, s \rangle \triangleright$

$\langle \text{STORE-X}, 4, s \rangle \triangleright \langle \epsilon, \epsilon, s[x \mapsto 4] \rangle$.

Esto es equivalente a $x := x + 1$.

(EJEMPLO 4.2) Considera:

Loop (TRUE, NOOP) y desarrollalo:

$\langle \text{LOOP}(\text{TRUE}, \text{NOOP}), \epsilon, s \rangle \triangleright \langle \text{TRUE} : \text{BRANCH}(\text{NOOP} : \text{LOOP}(\text{TRUE}, \text{NOOP}), \epsilon), s \rangle \triangleright$

$\triangleright \langle \text{BRANCH}(\text{NOOP} : \text{LOOP}(\text{TRUE}, \text{NOOP}), \text{NOOP}), \text{tt}, s \rangle \triangleright \langle \text{NOOP} : \text{LOOP}(\text{TRUE}, \text{NOOP}), \epsilon, s \rangle \triangleright$

$\triangleright \langle \text{LOOP}(\text{TRUE}, \text{NOOP}), \epsilon, s \rangle \triangleright \dots$

Esto es equivalente a while true do skip.

(4.3) Considera:

PUSH-0 : STORE-Z : FETCH-X : STORE-R : LOOP (FETCH-R : FETCH-Y : LE, FETCH-Y : FETCH-R :
SUB : STORE-R : PUSH-1 : FETCH-Z : ADD : STORE-Z)

$\langle \text{PUSH-0} : c, \epsilon, s \rangle \triangleright \langle \text{STORE-Z} : c_1, 0, s \rangle \triangleright \langle \text{FETCH-X} : c_2, \epsilon, s[z \mapsto 0] \rangle \triangleright$

$\triangleright \langle \text{STORE-R} : c_3, (s x), s[z \mapsto 0] \rangle \triangleright \langle \text{LOOP}(c_4, c_5), \epsilon, s[z \mapsto 0, r \mapsto (s x)] \rangle \triangleright$

$\triangleright \langle \text{FETCH-R} : \text{FETCH-Y} : \text{LE} : \text{BRANCH}(\text{FETCH-Y} : c_6 : \text{LOOP}(c_4, c_5), \text{NOOP}), \epsilon, s[z \mapsto 0, r \mapsto (s x)] \rangle \triangleright$

$\triangleright \langle \text{BRANCH}(\text{FETCH-Y} : c_6 : \text{LOOP}(c_4, c_5), \text{NOOP}), ((s y) \leq (s x)), s[z \mapsto 0, r \mapsto (s x)] \rangle \triangleright \dots$

Esto es equivalente a:

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

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



(4.4)* De muestra que:

Si

$$\langle c_1, e_1, s \rangle \Delta^k \langle c'_1, e'_1, s' \rangle \quad \text{entonces} \quad \langle c_1 : c_2, e_1 : e_2, s \rangle \Delta^k \langle c'_1 : c'_2, e'_1 : e'_2, s' \rangle$$

Inducción sobre K:

(CASO BASE)

$\langle c'_1, e'_1, s' \rangle$

$$K=0: \quad \langle c_1, e_1, s \rangle \Delta^0 \langle c'_1, e'_1, s' \rangle, \text{ luego: } \underbrace{c_1 = c'_1, e_1 = e'_1, s = s'}_{\text{y}}$$

$$\langle c_1 : c_2, e_1 : e_2, s \rangle \Delta^0 \langle c_1 : c_2, e_1 : e_2, s \rangle = \langle c'_1 : c'_2, e'_1 : e'_2, s' \rangle \quad \checkmark$$

$$K=1: \quad \boxed{\langle c_1, e_1, s \rangle \Delta^1 \langle c'_1, e'_1, s' \rangle}. \quad \text{Entonces } \langle c_1 : c_2, e_1 : e_2, s \rangle \text{ puede ser:}$$

$$C_1 = \text{PUSH-}n : C'$$

$$\text{Entonces: } \langle \text{PUSH-}n : c'_1, e_1, s \rangle \Delta \langle c'_1, \underbrace{\text{N}(n) : c_1}_{e''}, \underbrace{s}_{s'} \rangle, \text{ luego}$$

$$2) \quad \langle \text{PUSH-}n : c'_1 : c_2, e_1 : e_2, s \rangle \Delta \langle c'_1 : c_2, \underbrace{\text{N}(n) : e_1 : e_2}_{e''}, s \rangle = \langle c'_1 : c_2, e'_1 : e_2, s' \rangle \quad \checkmark$$

$$\boxed{C_1 = OP : c'_1}, \text{ donde } OP = \text{ADD, MULT o SUB}; \text{ y } e_2 = z_1 : z_2 : e_1$$

$$\text{Entonces: } 1) \quad \langle OP : c'_1, z_1 : z_2 : e_1, s \rangle = \langle c'_1, \underbrace{(z_1 \text{OP} z_2)}_{e''} : e_1, \underbrace{s}_{s'} \rangle, \text{ luego}$$

$$2) \quad \langle OP : c'_1 : c_2, z_1 : z_2 : e_1 : e_2, s \rangle \Delta \langle c'_1 : c_2, (z_1 \text{OP} z_2) : e_1 : e_2, s \rangle = \\ = \langle c'_1 : c_2, e'_1 : e_2, s' \rangle \quad \checkmark$$

$$\boxed{C_1 = LE : c'_1}, \boxed{C_1 = EQ : c'_1}, \boxed{C_1 = \text{TRUE} : c'_1}, \boxed{C_1 = \text{FALSE} : c'_1} \text{ ANÁLOGAS.}$$

$$\boxed{C_1 = \text{AND} : c'_1}, \text{ donde } e_1 = t_1 : t_2 : e_1'; z_1, z_2 \in T.$$

$$\text{Entonces: } 1) \quad \langle \text{AND} : c'_1, t_1 : t_2 : e_1', s \rangle \Delta \langle c'_1, \underbrace{(t_1 \text{AND} t_2)}_{e'} : e_1', \underbrace{s}_{s'} \rangle, \text{ luego}$$

$$\langle \text{AND} : c'_1 : c_2, t_1 : t_2 : e_1' : e_2, s \rangle \Delta \langle c'_1 : c_2, (t_1 \text{AND} t_2) : e_1' : e_2, s \rangle = \\ = \langle c'_1 : c_2, e'_1 : e_2, s' \rangle \quad \checkmark$$

El resto análogos, se lo desarrolla con BRANCH.

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



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

InfoJobs

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



PASO INDUCTIVO:

Sabemos que se cumple:

HI: \exists

si $\langle c_1, e_1, s \rangle \Delta^K \langle c', e', s' \rangle$ entonces $\langle c_1; c_2, e_1; e_2, s \rangle \Delta^K \langle c'; c_2, e'; e_2, s' \rangle$

Entonces lo demostraremos por K+1:

Sabemos $\langle c_1, e_1, s \rangle \Delta^{K+1} \langle c', e', s' \rangle$. Entonces:

$\langle c_1; c_2, e_1; e_2, s \rangle \Delta^K \langle c'_1; c_2, e'_1; e_2, s'_1 \rangle \Delta \langle c'; c_2, e'; e_2, s' \rangle$ ✓
 [HI]

↳ Este es el mismo caso que
 $K=1$, ($c'_1 = \text{push}_n : c'$, etc...).

Luego $\langle c_1; c_2, e_1; e_2, s \rangle \Delta^{K+1} \langle c'; c_2, e'; e_2, s' \rangle$ ✓

Luego se cumple ✓

(4.5) Demuestre que:

si $\langle c_1; c_2, e_1, s \rangle \Delta^K \langle e, e'', s'' \rangle$, entonces existe una configuración

$\langle e, e', s' \rangle$ y $K_1, K_2 \in \mathbb{N}$ tq $K = K_1 + K_2$ tal que:

$\langle c_1, e, s \rangle \Delta^{K_1} \langle e, e', s' \rangle$ y $\langle c_2, e', s' \rangle \Delta^{K_2} \langle e, e'', s'' \rangle$.

Sabemos que $\boxed{\langle c_1; c_2, e, s \rangle \Delta^K \langle e, e'', s'' \rangle}$

Inducción sobre K:

CASO BASE /

$K=0$

Entonces $\langle c_1; c_2, e, s \rangle \Delta^0 \langle e, e'', s'' \rangle$. Luego $c_1 = c_2 = e$, $e = e''$

y $s = s''$.

$\langle c_1, e, s \rangle = \langle e, e, s \rangle \Delta^{K_1} \langle e, e', s' \rangle$

↳ No queda otra que $K_1 = 0$,
 $e = e'$, $s = s'$

$\langle c_2, e', s' \rangle = \langle e, e, s \rangle \Delta^{K_2} \langle e, e'', s'' \rangle$

↳ No queda otra que $K_2 = 0$
 $\langle e, e, s \rangle$

Luego $K = K_1 + K_2 = 0 + 0$, y se cumple ✓.

* $c_1 = e$ * Es importante porque luego obligamos a que $c_1 \neq \underline{e}$.

Sabemos: $\langle c_1; c_2, e, s \rangle = \boxed{\langle c_2, e, s \rangle \Delta^K \langle e, e'', s'' \rangle}$, $K = K_1 + K_2$.

$\langle c_1, e, s \rangle = \langle e, e, s \rangle \Delta^{K_1} \langle e, e', s' \rangle$ Luego $\underline{K_1 = 0}$, $\underline{e = e'}$, $\underline{s = s'}$.

$\langle c_2, e', s' \rangle = \langle c_2, e, s \rangle \Delta^{K_2} \langle e, e'', s'' \rangle$, si $K_2 = K$, por *, lo tenemos *

PASO INDUCCIVO:

[HI] Suponemos cierto: $\langle C_1^*: c_1^*, e_*, s_* \rangle \Delta^{K_1} \langle \epsilon, e''_*, s''_* \rangle$ entonces $\langle C_2^*: c_2^*, e_*, s_* \rangle \Delta^{K_2} \langle \epsilon, e''_*, s''_* \rangle$.

$$\langle C_1^*: c_1^*, e_*, s_* \rangle \Delta^{K_1} \langle \epsilon, e'_*, s'_* \rangle$$

$$\langle C_2^*: c_2^*, e_*, s_* \rangle \Delta^{K_2} \langle \epsilon, e''_*, s''_* \rangle.$$

Demostremos para $K+1$:

Se demuestra $\langle C_1: c_1, e, s \rangle \Delta^{K+1} \langle \epsilon, e'', s'' \rangle$. Entonces:

$$\langle C_1, e, s \rangle \Delta^{K_1} \langle \epsilon, e', s' \rangle ? \text{ Suponemos } c_1 \neq \epsilon, \text{ luego } K_1 \neq 0.$$

$$\langle C_2, e', s' \rangle \Delta^{K_2} \langle \epsilon, e'', s'' \rangle ? \text{ (o } K+1 = K_1 + K_2).$$

Como $c_1 \in \text{Code}$, $c_1 = \text{inst}: c_1'$, luego:

Si $\text{inst} = \text{PUSH-}n$:

$$\langle \text{PUSH-}n: c_1': c_2, e, s \rangle \Delta \langle C_1': c_2, \text{dr}[n]: e, s \rangle \Delta^K \langle \epsilon, e'', s'' \rangle. \text{ Por lo HI,}$$

si $c_1' = C_1^*$, $c_2 = C_2^*$, $\text{dr}[n]: e = e'_*$, $s = s_*$, $e'' = e''_*$, $s'' = s''_*$, entonces:

$$\langle C_1', e, s \rangle \Delta^{K_1} \langle \epsilon, e', s' \rangle \text{ y } \langle C_2, e', s' \rangle \Delta^{K_2} \langle \epsilon, e'', s'' \rangle, \quad K = K_1 + K_2$$

Luego:

$$\langle \text{PUSH-}n: c_1', e, s \rangle \Delta \langle C_1', e, s \rangle \Delta^{K_1} \langle \epsilon, e', s' \rangle. \text{ Si } K_1' = K_1 + 1 \text{ y } K_2 = K_2'$$

$$\langle C_1, e, s \rangle \Delta^{K_1'} \langle \epsilon, e', s' \rangle. \text{ y } \langle C_2, e', s' \rangle \Delta^{K_2'} \langle \epsilon, e'', s'' \rangle \checkmark$$

TODOS LOS DEMÁS CASOS SON ANÁLOGOS.

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

InfoJobs



O

TPro

29/04/21(3)

④ Dmostrar qe AM es determinista.

S: $\langle \text{inst}: c, e, s \rangle \Delta \gamma_1$, c entonces $\gamma_1 = \gamma_2$?
 $\langle \text{inst}: c, e, s \rangle \Delta \gamma_2$

Hay que hacerlo para todos los posibles inst.

$\boxed{\text{inst} = \text{PUSH-}n}$

$\langle \text{PUSH-}n: c, e, s \rangle \Delta \gamma_1$, luego $\gamma_1 = \langle c, \text{dil}(n): e, s \rangle$. $\Rightarrow \gamma_1 = \gamma_2 \checkmark$.

$\langle \text{PUSH-}n: c, e, s \rangle \Delta \gamma_2$, luego $\gamma_2 = \langle c, \text{dil}(n): e, s \rangle$

$\boxed{\text{ADD, SUB, MUL, TRUE, FALSE, EQ, LE}}$ | Auslogos

$\boxed{\text{inst} = \text{AND}}$

Entonces $e = t_1 : t_2 : e_1$

$\langle \text{AND}: c, t_1 : t_2 : e_1, s \rangle \Delta \gamma_1$, luego $\gamma_1 = \begin{cases} \langle c, \text{tt}: e_1, s \rangle & \text{si } B[t_1] = \text{tt} \text{ y } B[t_2] = \text{tt} \\ \langle c, \text{ff}: e_1, s \rangle & \text{en otro caso} \end{cases} \Rightarrow$

$\langle \text{AND}: c, t_1 : t_2 : e_1, s \rangle \Delta \gamma_2$, luego $\gamma_2 = \begin{cases} \langle c, \text{tt}: e_1, s \rangle & \text{si } B[t_1] = \text{tt} \text{ y } B[t_2] = \text{tt} \\ \langle c, \text{ff}: e_1, s \rangle & \text{en otro caso} \end{cases}$

$\Rightarrow \gamma_1 = \gamma_2$.

$\boxed{\text{Todos los demás auslogos}}$ #

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



WUOLAH

4.7) Sea AM_1 una máquina igual que AM pero con config. $\langle c, e, m \rangle$, con m la memoria, una lista finita de valores ($m \in \mathbb{Z}^*$), y donde $\text{FETCH-}x$ y $\text{STORE-}x$ son $\text{GET-}n$ y $\text{PUT-}n$, con n n° natural, y donde necesitas coger $m[n]$ para seleccionar el elemento n -ésimo ($n \in \{1, \dots, \text{len}(m)\}$). Escribe su semántica.

$\text{GET-}n : \langle \text{GET-}n : c, e, m \rangle \rightarrow \langle c, m[n] : e, m \rangle \text{ if } n \in \{1, \dots, \text{len}(m)\}$.

$\text{PUT-}n : \langle \text{PUT-}n : c, z, e, m \rangle \rightarrow \langle c, e, m[m[n] \rightarrow z] : e, m \rangle \text{ if } n \in \{1, \dots, \text{len}(m)\}$

4.8) Sea AM_2 una máquina con config. $\langle pc, c, e, m \rangle$, donde se DEFINEN y se SALTA HACIA ETIQUETAS. c, e y m son como en AM_1 , y pc es un n° natural que APUNTA HACIA UNA INSTRUCCIÓN c .

Las instrucciones BRANCH y LOOP se sustituyen por LABEL- l , JUMP- l y JUMPFALSE- l , donde l es la ETIQUETA ($l \in \mathbb{N}$).

La idea es ejecutar la instrucción c o la que pc apunta. LABEL- l sólo aumenta el contador pc . JUMP- l move el contador a la instrucción LABEL- l (si existe). JUMPFALSE- l sólo move el contador hacia LABEL- l si se encuentra con ff:e. Si se encuentra con tt:e, el contador se incrementará en 1.

Se escribe $c[pc]$ o la instrucción en c operada por pc ($pc \in \{1, \dots, \text{len}(c)\}$). Especifica una semántica.

$\text{LABEL-}l : \langle pc, c, e, m \rangle \rightarrow \langle pc+1, c, e, m \rangle \text{ if } c[pc] = \text{LABEL-}l$.

$\text{JUMP-}l : \langle pc, c, e, m \rangle \rightarrow \langle \text{poslabel}(c), c, e, m \rangle \text{ if } c[pc] = \text{JUMP-}l$.

↳ Necesitamos saber la posición del LABEL ANTES, esto se hace con una FUNCIÓN AUXILIAR.

$\text{poslabel} : \text{CODE} \rightarrow \mathbb{N}$
 $(\text{LABEL-}l : c) \rightarrow 0$
 $(q : c) \rightarrow 1 + \text{poslabel}(c)$.

$\text{JUMPFALSE-}l : \langle pc, c, tt : e, m \rangle \rightarrow \langle pc+1, c, e, m \rangle \text{ if } c[pc] = \text{JUMPFALSE-}l$.

$\langle pc, c, ff : e, m \rangle \rightarrow \langle \text{poslabel}(c), c, e, m \rangle \text{ if } c[pc] = \text{JUMPFALSE-}l$.

(4.9)

Luego el esquema de instrucciones para AM_3 es:

- | | |
|--|---------------------------------|
| $\langle pc, c, e, m \rangle \rightarrow \langle pc+1, c, \text{N}[n]:e, m \rangle$ | si $c[pc] = \text{PUSH-}n$ |
| $\langle pc, c, z_1:z_2:e, m \rangle \rightarrow \langle pc+1, c, (z_1+z_2):e, m \rangle$ | si $c[pc] = \text{ADD}$ |
| $\langle pc, c, z_1:z_2:e, m \rangle \rightarrow \langle pc+1, c, (z_1-z_2):e, m \rangle$ | si $c[pc] = \text{SUB}$ |
| $\langle pc, c, z_1:z_2:e, m \rangle \rightarrow \langle pc+1, c, (z_1 * z_2):e, m \rangle$ | si $c[pc] = \text{MULT}$ |
| $\langle pc, c, e, m \rangle \rightarrow \langle pc+1, c, tt:e, m \rangle$ | si $c[pc] = \text{TRUE}$ |
| $\langle pc, c, e, m \rangle \rightarrow \langle pc+1, c, ff:e, m \rangle$ | si $c[pc] = \text{FALSE}$ |
| $\langle pc, c, z_1:z_2:e, m \rangle \rightarrow \langle pc+1, c, (z_1 = z_2):e, m \rangle$ | si $c[pc] = \text{EQ}$ |
| $\langle pc, c, z_1:z_2:e, m \rangle \rightarrow \langle pc+1, c, (z_1 \leq z_2):e, m \rangle$ | si $c[pc] = \text{LE}$ |
| $\langle pc, c, t_1:t_2:e, m \rangle \rightarrow \begin{cases} \langle pc+1, c, tt:e, m \rangle \text{ si } B[t_1] = tt \text{ y } B[t_2] = tt \\ \langle pc+1, c, ff:e, m \rangle \text{ si } B[t_1] = ff \text{ o } B[t_2] = ff \end{cases}$ | |
| $\langle pc, c, t:e, m \rangle \rightarrow \begin{cases} \langle pc+1, c, ff:e, m \rangle \text{ si } B[t] = tt \\ \langle pc+1, c, tt:e, m \rangle \text{ si } B[t] = ff \end{cases}$ | si $c[pc] = \text{NEG}$ |
| $\langle pc, c, e, m \rangle \rightarrow \langle pc+1, c, m[n]:e, m \rangle$ | si $c[pc] = \text{GET-}n$ |
| $\langle pc, c, z:e, m \rangle \rightarrow \langle pc+1, c, e, m[m[n] \mapsto z] \rangle$ | si $c[pc] = \text{PUT-}n$ |
| $\langle pc, c, e, m \rangle \rightarrow \langle pc+1, c, e, m \rangle$ | si $c[pc] = \text{NOOP}$ |
| $\langle pc, c, e, m \rangle \rightarrow \langle z, c, e, m \rangle$ | si $c[pc] = \text{JUMP-}z$ |
| $\langle pc, c, t:e, m \rangle \rightarrow \begin{cases} \langle pc+1, c, e, m \rangle \text{ si } B[t] = tt \\ \langle z, c, e, m \rangle \text{ si } B[t] = ff \end{cases}$ | si $c[pc] = \text{JUMPFALSE-}z$ |

4.11 Es claro que $CA[(a_1+a_2)+a_3]$ es equivalente a $CA[a_1+(a_2+a_3)]$.

Demostraremos que no es cierto que $CA[(a_1+a_2)+a_3]$ sea equivalente a $CA[a_1+(a_2+a_3)]$. Sin embargo, muestra que sí ocurre de formas similar.

$$CA[(a_1+a_2)+a_3] = CA[a_1+a_2]:CA[a_3]:ADD = CA[a_1]:CA[a_2]:ADD:CA[a_3]:ADD$$

$$CA[a_1+(a_2+a_3)] = CA[a_1]:CA[a_2+a_3]:ADD = CA[a_1]:CA[a_2]:CA[a_3]:ADD:ADD$$

Poniendo el siguiente ejemplo:

$$CA[(x+5)+y] = \text{FETCH}-x:\text{PUSH}-5:\text{ADD}:\text{FETCH}-y:\text{ADD}$$

$$CA[x+(5+y)] = \text{FETCH}-x:\text{PUSH}-5:\text{FETCH}-y:\text{ADD}:\text{ADD}$$

Al no ser el mismo código, queda claro que no hay una igualdad entre las instrucciones, pero sí ocurre de formas similar, ya que producen los mismos resultados.

4.14 Extiende while con la construcción repeat S until b y especifica cómo generar código para ello. La definición deberá ser composicional y que no es necesario extender el conjunto de instrucciones de AM.

Proporcionamos $CS[\text{repeat } S \text{ until } b] =$

$$CS[\text{repeat } S \text{ until } b] = CS[S]:CB[b]:\text{BRANCH}(\text{NOOP}, CS[\text{repeat } S \text{ until } b]).$$

Para una definición no composicional podemos usar la equivalencia

que nos define que $\text{repeat } S \text{ until } b$ equivale a $S; \text{while } \neg b \text{ do } S \text{ else skip.}$ con lo

que quedaría:

$$CS[\text{repeat } S \text{ until } b] = CS[S]:\text{LOOP}(CB[\neg b], CS[S]) =$$

$$= CS[S]:\text{LOOP}(CB[b]:\text{NEG}, CS[S]).$$



CAJA MENSUAL VALORADA

EN 3000€

TPR0

18/05/21.

Q.16 Modifica la gama de código para traducir WHILE para AM₁. Deberás asumir la existencia de una función:

$\text{env} : \text{Var} \rightarrow \mathbb{N}$

que impone las variables e sus direcciones. Aplica esto al statement factorial del Ejemplo 4.12 y se comienza desde x vale 3.

Todos las instrucciones de CA, CB y CS son iguales, salvo,

$$CA[\bar{x}] = GET - (\text{env } x)$$

$$CS[x := a] = CA[a] : \text{PUT-}(\text{env}\ x)$$

FC statement es: $y := 1$; while $\gamma(x=1)$ do ($y := y * x$; $x := x - 1$)

`<CS, [y:=1; while (7x=4) do (y:=y*x ; x:=x-1)], E, [3]>` ▷ *

$CS_1[\lceil y := s; \text{while } (x \neq t) \text{ do } (y := y \wedge x; x := x - 1) \rceil] =$

$$= \text{PUSH-1 : PUT -}(\text{env } y) : \text{LOOP}\left(\text{CA[1]} : \text{CA[x]} : \text{EQ:NEG}, \text{CS[y := y * x]} : \text{CS[x := x - 1]}\right)$$

$$= \text{PUSH-1:PUT-(env y)} : \text{LOOP} \left(\text{PUSH-1:GET-(env x)} : \text{EQ} : \text{NEG}, \text{ GET-(env x)} : \text{GET-(env y)} : \text{MULT} : \text{PUT-(env y)} : \text{CALC-X-Y} \right)$$

$= \text{PUSH-1:PUT-}(\text{env } y)\text{:LOOP} \left(\text{PUSH-1:GET-}(\text{env } x)\text{:EQ:NEG}, \underbrace{\text{GET-}(\text{env } x)\text{:GET-}(\text{env } y)\text{:MULT:PUT-}(\text{env } y)\text{:PUSH-1:EQ:...}}_{**} \right)$

* $\triangleright \langle \text{PUT-env } y : \dots, 1, [3] \rangle \triangleright \langle \text{LOOP}(\dots), \dots, [2, 3], \dots \rangle$

$\triangleright \langle \text{BRANCH}(\dots), \text{ff}, [3,1] \rangle \triangleright \langle **:\text{LOOP}(\dots), \mathcal{E}, [3,1] \rangle \triangleright^* \langle \text{LOOP}(\dots), \mathcal{E}, [2,1] \rangle \triangleright$

$\triangleright^* \langle \text{LOOP}(\dots), \varepsilon, [4, 6] \rangle \triangleright \langle \text{RUSH-1:...:BRANCH}(\dots), \varepsilon, [4, 6] \rangle$

$$\langle \varepsilon, \varepsilon, [1, 6] \rangle \triangleright \langle \varepsilon, \varepsilon, [1, 6] \rangle$$

↳ Hemos llegado a "x" es 6 e "y" es 1.



Promoción válida
del 07 de abril
al 30 de septiembre
de 2024.
Bases depositadas
ante notario.
Consultar en
bifurtas.com

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

4.19 Demuestre que, para todos los expresiones booleanas b , se cumple
 $\langle CB[b], \epsilon, s \rangle \Delta^* \langle \epsilon, B[b]s, s \rangle$.

Además, muestre que todos los configuraciones intermedias tanto stack de evaluación no vacío.

Procedemos por inducción estructural en b :

CASO BASE:

- $b = \text{true}$: $\langle CB[\text{true}], \epsilon, s \rangle = \langle \text{true}, \epsilon, s \rangle \Delta \langle \epsilon, tt, s \rangle$

Por otro lado: $\langle \epsilon, B[\text{true}]s, s \rangle = \langle \epsilon, tt, s \rangle$. Son las mismas config. ✓

- $b = \text{false}$ ANALOGO En los dos casos se ven los estados intermedios sin stack de evaluación vacío.

- $b \equiv \alpha_1 = \alpha_2$: $\langle CB[\alpha_1 = \alpha_2], \epsilon, s \rangle = \langle CA[\alpha_2] : CA[\alpha_1] : EQ, \epsilon, s \rangle \Delta$

$$\Delta^* \langle CA[\alpha_1] : EQ, A[\alpha_2]s, s \rangle \Delta^* \langle EQ, A[\alpha_1]s : A[\alpha_2]s, s \rangle \Delta$$

Lema 4.18 \rightarrow Su stack eval. vacío en \leftarrow Lema 4.18
 + Ej 4.4 estados intermedios. \uparrow + Ej 4.4.

$$\Delta \langle \epsilon, (A[\alpha_1]s = A[\alpha_2]s), s \rangle = \langle \epsilon, B[\alpha_1 = \alpha_2]s, s \rangle \quad \checkmark$$

- $b \equiv \alpha_1 \leq \alpha_2$ ANALOGO.

PASO INDUCTIVO:

(H) Supongamos que dado $b_1, b_2 \in B_{\text{exp}}$, se cumple que:

$$\langle CB[b_1], \epsilon, s \rangle \Delta^* \langle \epsilon, B[b_1]s, s \rangle. \quad \text{sin stacks de eval. intermedios vacíos.}$$

$$\langle CB[b_2], \epsilon, s \rangle \Delta^* \langle \epsilon, B[b_2]s, s \rangle.$$

- $b \equiv \neg b_1$: $\langle CB[\neg b_1], \epsilon, s \rangle = \langle CB[b_1] : \text{NEG}, \epsilon, s \rangle \Delta^*$ (H) + Ej. 4.4.

$$\Delta^* \langle \text{NEG}, B[b_1]s, s \rangle \Delta \begin{cases} \langle \epsilon, ff, s \rangle & \text{si } B[b_1]s = tt \\ \langle \epsilon, tt, s \rangle & \text{si } B[b_1]s = ff \end{cases} = \text{DEF.}$$

$$= \langle \epsilon, B[\neg b_1]s, s \rangle \quad \checkmark$$

$$\bullet b = b_1 \wedge b_2$$

$$\langle CB[b_1 \wedge b_2], \mathcal{E}, s \rangle = \langle CB[b_2] : CB[b_1] : \text{AND}, \mathcal{E}, s \rangle \stackrel{\Delta^*}{\underset{\text{DEF}}{=}}$$

$$\Delta^* \langle CB[b_1] : \text{AND}, CB[b_2]s, s \rangle \stackrel{\Delta^*}{\underset{\text{DEF}}{=}} \langle \text{AND}, CB[b_1]s : CB[b_2]s, s \rangle \stackrel{\Delta^*}{\underset{\text{DEF}}{=}}$$

+ Ej. 4.4.

$$\Delta \langle \mathcal{E}, CB[b_1 \wedge b_2]s, s \rangle \checkmark$$

(4.23) Considera el código "optimizado" para la función CS' donde $CS'[skip] = \mathcal{E}$ y el resto igual que CS . ¿Cúalquier esto la demostración del TH. 4.20?

$$\Rightarrow \text{Supongamos que } \langle skip, s \rangle \rightarrow s' \quad (s' = s)$$

$$\text{Entonces } \langle CS[skip], \mathcal{E}, s \rangle \stackrel{\Delta^*}{\underset{\text{DEF}}{=}} \langle \mathcal{E}, \mathcal{E}, s \rangle \checkmark$$

~~Así~~ Yo si teorizamos el caso $K=0$, luego teorizamos que dentro

~~de~~ todos los propiedades para $K=0$, superponiendo por:

$$\text{Entonces } \langle CS[skip], \mathcal{E}, s \rangle \stackrel{\Delta^*}{\underset{\text{DEF}}{=}} \langle \mathcal{E}, \mathcal{E}, s \rangle.$$

$$\text{Supongamos que } \langle skip, s \rangle \rightarrow s \checkmark$$

$$\text{Entonces } \langle skip, s \rangle \rightarrow s \checkmark$$

Tampoco se podrían aplicar el caso $|K|+1$ al ser skip de 0 pasos.

(4.24) Extiende la demostración del TH. 4.20 para que se cumpla en la instrucción repeat S until b. El código generado está en los ejes.

2.7 y 4.14.

Por el ej. 2.7:

$\text{[repeat}_{ns}^{sp}]$	$\frac{\langle S, s \rangle \rightarrow S^*, \langle \text{repeat } S \text{ until } b, s^* \rangle \rightarrow S'}{\langle \text{repeat } S \text{ until } b, s \rangle \rightarrow S'}$	<small>si $CB[b]s = sp$</small>
$\text{[repeat}_{ns}^{tt}]$	$\frac{\langle S, s \rangle \rightarrow S^*}{\langle \text{repeat } S \text{ until } b, s \rangle \rightarrow S'}$	<small>si $CB[b]s = tt$</small>

$$\boxed{\text{Por el ej. 4.14: } CS[\text{repeat } S \text{ until } b] = CS[S] : \text{LOOP}(CB[b] : \text{NEG}, CS[S])}$$

$\Rightarrow 1$ (caso 4.2f)

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

Este es un caso del PASO INDUCTIVO de la inducción estructural:

(H1) Supongamos que $\langle S, s \rangle \rightarrow S^*$ implica que $\langle CS[S], \epsilon, s \rangle \Delta^* \Delta^* \langle \epsilon, \epsilon, S^* \rangle$.

Entonces $\langle CS[\text{repeat } S \text{ until } b], \epsilon, s \rangle \Delta^* \langle CS[S]: \text{loop}(CB[b]:\text{NEG}, CS[S]), \epsilon, s \rangle \Delta^*$

$\Delta^* \langle \text{loop}(CB[b]:\text{NEG}, CS[S]), \epsilon, S^* \rangle \Delta^* \langle CB[b]:\text{NEG}: \text{BRANCH}(CS[S]:\text{loop}(*), \text{NOOP}), \epsilon, S^* \rangle$

$\Delta^* \langle \text{BRANCH}(CS[S]:\text{loop}(CB[b]:\text{NEG}, CS[S]), \text{NOOP}), CB[b]S^*, S^* \rangle \Delta^* ①$

↑
Ej. 4.19.
+ TABLA 4.1.

• Si $\underline{CB[b]S^* = tt}$:

Por $\langle \text{repeat } ns \rangle$: $\frac{\langle S, s \rangle \rightarrow S^*}{\langle \text{repeat } S \text{ until } b, s \rangle \rightarrow S^*}$, luego por el ENUNCIADO: $S' = \underline{S^*}$

Luego ① $\Delta^* \langle \text{NOOP}, \epsilon, S' \rangle \Delta^* \langle \epsilon, \epsilon, S' \rangle$ ✓

$\boxed{CB[b]S^* = ff, S' = S^*}$

• Si $\underline{CB[b]S^* = ff}$:

Por $\langle \text{repeat } ns \rangle$: $\frac{\langle S, s \rangle \rightarrow S^*, \langle \text{repeat } S \text{ until } b, S^* \rangle \rightarrow S^*}{\langle \text{repeat } S \text{ until } b, s \rangle \rightarrow S^*}$

Aquí es necesaria otra hipótesis de inducción:

(H1 2) Supongamos que $\langle \text{repeat } S \text{ until } b, S^* \rangle \rightarrow S'$. Entonces

$\langle CS[\text{repeat } S \text{ until } b], \epsilon, S^* \rangle \Delta^* \langle \epsilon, \epsilon, S' \rangle$.

Luego: ① $\Delta^* \langle CS[S]: \text{loop}(CB[b]:\text{NEG}, CS[S]), \epsilon, S^* \rangle \Delta^* \text{ ENUNCIADO}$

$\boxed{CB[b]S^* = tt}$

$\Delta^* \langle CS[\text{repeat } S \text{ until } b], \epsilon, S^* \rangle \Delta^* \langle \epsilon, \epsilon, S' \rangle$ ✓

H1 2

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



My CREW
Mi trabajo. Mi pasión. Mi gente.

My CREW
Mi trabajo. Mi pasión. Mi gente.

TP20 → CASO $S' \equiv \text{repeat } S \text{ until } b$:

19/05/21 (2)

1) (caso 6.22) $\frac{S'}{S''}$

Desarrollamos que $\langle CS[\text{repeat } S \text{ until } b], \epsilon, s \rangle \Delta^K \langle \epsilon, e', s' \rangle$. implica que $\langle S', s \rangle \rightarrow s'$ y $e' = \epsilon$.

Usaremos inducción sobre K.

- Para $K=0$ el resultado es correcto porque es imposible que ocurra $CS[\epsilon] = \epsilon$.
- Supongamos que se cumple para todo $K \leq K_0$ y desarrollemos para $K+1$.

H1: Si $\langle CS[\text{repeat } S \text{ until } b], \epsilon, s \rangle \Delta^K \langle \epsilon, e', s' \rangle$, ($K \leq K_0$)

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

Supongamos $\langle CS[\text{repeat } S \text{ until } b], \epsilon, s \rangle \Delta^{K+1} \langle \epsilon, e', s' \rangle$.

Desarrollamos $\langle CS[\text{repeat } S \text{ until } b], \epsilon, s \rangle \Delta^0 \langle CS[S]:\text{LOOP}(CB[b]:\text{NEG}, CS[S]), \epsilon, s \rangle$.

Por el ej. 4.5 (aplicando ①), existen $k_1, k_2 \in \mathbb{N}$ ($\text{con } K+1 = k_1 + k_2$) y config. $\langle \epsilon, e^*, s^* \rangle$ tales que $\langle CS[S], \epsilon, s \rangle \Delta^{k_1} \langle \epsilon, e^*, s^* \rangle$ y $\langle \text{LOOP}(CB[b]:\text{NEG}, CS[S]), e^*, s^* \rangle \Delta^{k_2} \langle \epsilon, e', s' \rangle$

H1 2: Podemos asumir que si $\langle CS[S'], \epsilon, s \rangle \Delta^K \langle \epsilon, e', s' \rangle$, entonces $\langle S', s \rangle \rightarrow S'$ y $e' = \epsilon$ (OBS: S NO ES S' , S' es el repeat y S el statement de dentro).

Usando H1 2, $k_1 \leq K_0$ y ej. 6.6 (Ah determinista), no queda otra que $e^* = \epsilon$ y que $\langle S', s \rangle \rightarrow S^*$.

Por otro lado, $\langle \text{LOOP}(CB[b]:\text{NEG}, CS[S]), \epsilon, s^* \rangle \Delta^0$

$\Delta \langle CB[b]:\text{NEG}:\text{BRANCH}(CS[S']):\text{LOOP}(CB[b]:\text{NEG}, CS[S']), \epsilon, s^* \rangle$

Por el ej. 6.5: existen $k_3, k_4 \in \mathbb{N}$ tq $\underbrace{K_0 + 1 - k_1 - 1}_{\substack{\text{desarrollo} \\ CS[S']}} = k_3 + k_4$ tales que:

$\langle CB[b]:\text{NEG}, \epsilon, s^* \rangle \Delta^{k_3} \langle \epsilon, e'', s'' \rangle$ y $\langle \text{BRANCH}(CS[S']):\text{LOOP}(CB[b]:\text{NEG}, CS[S']), \epsilon, s'' \rangle \Delta^{k_4} \langle \epsilon, e', s' \rangle$.

Por el ej. 6.4, 6.6 (Ah det.), 6.19 y tabla NEG: $e'' = CB[b]s''$ y $s'' = s^*$

- Si $CB[b]s^* = tt \rightarrow CB[b]s^* = ff$:

$\langle \text{BRANCH}(CS[S']):\text{LOOP}(CB[b]:\text{NEG}, CS[S']), \epsilon, s^* \rangle \Delta^{k_3} \langle \epsilon, e'', s'' \rangle$

$\Delta \langle CS[S']::\text{LOOP}(CB[b]:\text{NEG}, CS[S']), \epsilon, s^* \rangle$

¿TE VIENES?



WUOLAH

H3 Podemos asumir que si $\langle CS[\text{repeat } S \text{ until } b], \epsilon, s^* \rangle \rightarrow \langle \epsilon, e'', s'' \rangle$
entonces $\langle \text{repeat } S \text{ until } b, s^* \rangle \rightarrow s''$ y $e'' = \epsilon$.

Si seguimos desarrollando la expresión anterior:

$\langle CS[S]:\text{loop}(CB[b]):\text{NEG}, CS[S'] \rangle, \epsilon, s^* \rightarrow \langle CS[\text{repeat } S \text{ until } b], \epsilon, s^* \rangle \rightarrow^*$
 $\Delta^* \langle \epsilon, e', s' \rangle$.

Premisa inicial

Usando H3: $\langle \text{repeat } S \text{ until } b, s^* \rangle \rightarrow s'$ y $e' = \epsilon$. ✓
y finalmente usando $\langle S, s \rangle \rightarrow s^*$, llegamos a $\langle \text{repeat } S \text{ until } b, s \rangle \rightarrow s'$
↳ (Usando la TABLA NS de repeat).

• Si $CB[b]s^* = ff \rightarrow CB[b]s^* = tt$

$\langle \text{BRANCH}(CS[S']: \text{loop}(CB[b]):\text{NEG}, CS[S']) , \text{NOOP}, CB[b]s, s^* \rangle \rightarrow$
 $\rightarrow \langle \text{NOOP}, \epsilon, s^* \rangle \rightarrow \langle \epsilon, \epsilon, s^* \rangle \Delta^* \langle \epsilon, e', s' \rangle$

Premisa inicial

Queda claro que $e' = \epsilon'$, y como $\langle S, s \rangle \rightarrow s^*$ y $s^* = s'$,

no queda otra que $\langle \text{repeat } S \text{ until } b, s \rangle \rightarrow s'$. ✓

(27) * Demuestre que si $\gamma_{SOS} \approx \gamma_{AM}$ y $\gamma_{SOS} \Rightarrow \gamma'_{SOS}$

entonces existe una configuración γ'_{AM} tal que:

$$\gamma_{AM} \Delta^+ \gamma'_{AM} \quad y \quad \gamma'_{SOS} \approx \gamma'_{AM} \quad \begin{array}{l} (\text{Muestre que si } \langle S, S \rangle \Rightarrow^* S', \text{ entonces}) \\ (\langle CS[\alpha], \epsilon, S \rangle \Delta^* \langle \epsilon, \epsilon, S \rangle) \end{array}$$

Demostremos esto viendo de qué formas puede ocurrir $\gamma_{SOS} \Rightarrow \gamma'_{SOS}$, realizando inducción estructural en los pasos comparsionales.

CASO BASE:

- Si $\gamma_{SOS} = \langle x := a, S \rangle$. Entonces $\gamma_{SOS} \Rightarrow S[x \mapsto \alpha]S = \gamma'_{SOS}$

La relación bisimulacional $\gamma_{SOS} \approx \gamma_{AM}$ se produce cuando

$$\gamma_{AM} = \langle CS[\alpha], \epsilon, S \rangle = \langle CA[a]: STORE - x, \epsilon, S \rangle \xrightarrow{*}$$

$$\Delta^* \langle STORE - x, \alpha]S, S \rangle \xrightarrow[TABLA]{} \langle \epsilon, \epsilon, S[x \mapsto \alpha]S \rangle \approx \gamma'_{SOS}. \checkmark$$

- Si $\gamma_{SOS} = \langle \text{skip}, S \rangle$. Entonces $\gamma_{SOS} \Rightarrow S = \gamma'_{SOS}$

La relación $\gamma_{SOS} \approx \gamma_{AM}$ es si $\gamma_{AM} = \langle CS[\text{skip}], \epsilon, S \rangle = \langle \text{NOOP}, \epsilon, S \rangle \Delta$

$$\Delta \langle \epsilon, \epsilon, S \rangle \approx \gamma'_{SOS} = S. \checkmark$$

PASO INDUCTIVO:

(HI): Supongamos que $\langle S_1, S_1 \rangle \Rightarrow^* S'_1$ y que $\langle S_2, S_2 \rangle \Rightarrow^* S'_2$.
Entonces $\langle CS[S_1], \epsilon, S_1 \rangle \Delta \langle \epsilon, \epsilon, S'_1 \rangle$ y $\langle CS[S_2], \epsilon, S_2 \rangle \Delta \langle \epsilon, \epsilon, S'_2 \rangle$

- Si $\gamma_{SOS} = \langle S_1; S_2, S \rangle$. Entonces $\langle S_1; S_2, S \rangle \Rightarrow^* S' = \gamma'_{SOS}$

$$\Delta \langle S_1, S_2, S \rangle \Rightarrow^* \langle S_2, S \rangle \Rightarrow^* S' = \gamma'_{SOS}$$

Por el ej. 2.19, $\exists k_1, k_2 \in \mathbb{N}$ con $k = k_1 + k_2$ tq $\langle S_2, S \rangle \Rightarrow^k S'$
y $\langle S_2, S \rangle \Rightarrow^{k_2} S'$.

$$\text{Aplicando HI: } \underbrace{\langle CS[S_1], \epsilon, S \rangle \Delta^* \langle \epsilon, \epsilon, S^* \rangle}_{(1)} \quad y \quad \underbrace{\langle CS[S_2], \epsilon, S^* \rangle \Delta^* \langle \epsilon, \epsilon, S' \rangle}_{(2)}$$

Usando el ej. 4.4 con (1) para introducirle (2), obtenemos:

$$\begin{aligned} &\text{Usando el ej. 4.4 con (1) para introducirle (2), obtenemos:} \\ &\langle CS[S_1]; CS[S_2], \epsilon, S \rangle \Delta^* \langle CS[S_2], \epsilon, S^* \rangle \Delta^* \langle \epsilon, \epsilon, S' \rangle \approx \gamma'_{SOS} = S' \\ &\langle CS[S_1]; CS[S_2], \epsilon, S \rangle \Delta^* \langle CS[S_2], \epsilon, S^* \rangle \Delta^* \langle \epsilon, \epsilon, S' \rangle \approx \gamma'_{SOS} = S' \\ &\qquad\qquad\qquad \downarrow \\ &\qquad\qquad\qquad \underline{\text{HI}} \end{aligned}$$

• Si $\gamma_{SOS} = \langle \text{if } b \text{ then } S_1 \text{ else } S_2, S \rangle \Rightarrow^* S' = \gamma'_{SOS}$

Entonces puede ocurrir:

1) $\mathcal{B}[b]S = tt$: Usando [γ_{SOS}^{tt}]: $\gamma_{SOS} \Rightarrow \langle S_1, S \rangle \Rightarrow^* S' = \gamma'_{SOS}$.

Lo anterior se produce cuando $\gamma_{AM} = \langle CS[\text{if } b \text{ then } S_1 \text{ else } S_2], E, S \rangle \triangleright^0$

$\triangleright^0 \langle CB[b] \rangle: \text{BRANCH} \langle S_1, S_2 \rangle, E, S \rangle \triangleright^* \langle \text{BRANCH} \langle S_1, S_2 \rangle, \mathcal{B}[b]S, S \rangle \triangleright^0$
 \uparrow Ej. 4.19
 $\mathcal{B}[b]S = tt$

$\triangleright \langle S_2, E, S \rangle \triangleright^* \langle E, E, S' \rangle \approx \gamma'_{SOS} = S' \quad \checkmark$
HI aplicado a $\langle S_1, S \rangle \Rightarrow^* S'$

2) $\mathcal{B}[b] = ff$: ANÁLOGO.

• Si $\gamma_{SOS} = \text{while } b \text{ do } S \Rightarrow \langle \text{if } b \text{ then } (S; \text{while } b \text{ do } S') \text{ else skip}, S \rangle \Rightarrow^*$
 $\Rightarrow^* S' = \gamma'_{SOS}$.

Entonces puede ocurrir:

1) $\mathcal{B}[b]S = tt$: Usando [while_{SOS}^{tt}]:

$\gamma_{SOS} \Rightarrow^* \langle S; \text{while } b \text{ do } S, S \rangle$
 \uparrow

$\mathcal{B}[b]S = tt$

Por el ej. 2.19, tenemos que $\exists k_1, k_2 \in \mathbb{N}$ tq. $\langle S, S \rangle \Rightarrow^* S^*$ y

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

(HI 2): Suponemos que si $\langle \text{while } b \text{ do } S, S^* \rangle \Rightarrow S'$, entonces

$\langle CS[\text{while } b \text{ do } S], E, S^* \rangle \Rightarrow^* \langle E, E, S' \rangle$

El equivalente es $\gamma_{AM} = \langle CS[\text{while } b \text{ do } S], E, S \rangle \Rightarrow$

$= \langle \text{loop}(\mathcal{B}[b], CS[S]), E, S \rangle \triangleright \langle CB[b]: \text{BRANCH}(S; S); \text{loop}(\mathcal{B}[b], CS[S]), E, S \rangle$

$\triangleright \langle CS[S]; \text{loop}(\mathcal{B}[b], CS[S]), E, S \rangle \Rightarrow^* \langle CS[\text{while } b \text{ do } S], E, S^* \rangle$
 \uparrow HI aplicado a $\langle S, S \rangle \Rightarrow S^*$
 $\mathcal{B}[b]S = tt$ + Ej. 4.4.

$\triangleright^* \langle \text{loop}(\mathcal{B}[b], CS[S]), E, S^* \rangle \triangleright^0 \langle CS[\text{while } b \text{ do } S], E, S^* \rangle \triangleright^* \quad \text{HI 2.}$

$\triangleright^* \langle E, E, S' \rangle \approx \gamma'_{SOS} = S' \quad \checkmark$

2) $\mathcal{B}[b]S = ff$ es muy fácil y directo.