

# Fórmulas parcial 1

jueves, 25 de abril de 2024 8:55

semántica denotacional = cualquier función (total?) de los árboles en algo

Categoría sintáctica = la cosa entre  $\langle \rangle$

Función semántica = Función de la categoría sintáctica en algo

Ecuación semántica = ????

Ecuaciones dirigidas por sintaxis = ecuación recursiva en la gramática que solo usa los resultados en las sobreexpresiones directas (y nada mas)

Función dirija por sintaxis = Ecuaciones dirigidas por sintaxis que dan una única función

Semántica composicional = semántica denotacional en el que el significado de una frase depende solo del significado de sus subfrases

## Lifting:

Sea  $X$  un conjunto

$$X_{\perp} = X \cup \{\perp\}$$

$$X^{\infty} = X \cup \{\infty\}$$

$(X, =)$  es el orden discreto

Sean:

$\leq_X$  un orden parcial sobre  $X$

$x, y \in X$

$$x \leq_{X_{\perp}} y \Leftrightarrow x \leq_X y \vee x = \perp$$

$$x \leq_{X^{\infty}} y \Leftrightarrow x \leq_X y \vee y = \infty$$

$(X_{\perp}, =_{\perp})$  es el orden llano

## Orden de funciones:

Sean:

$$f, g : X \rightarrow Y$$

$\leq_Y$  un orden parcial sobre  $Y$

$$f \leq g \Leftrightarrow \forall x \in X : f(x) \leq g(x)$$

## Cadena:

Sean:

$(P, \leq)$  un poset

$p_0, p_1, \dots \in P$

$$p_0 p_1 \dots \text{ es una cadena} \Leftrightarrow p_0 \leq p_1 \leq \dots$$

Sea  $p_0 p_1 \dots$  una cadena

$$p_0 p_1 \dots \text{ es interesante} \Leftrightarrow \{p_0, p_1, \dots\} \text{ es infinito}$$

## Predominio y dominio:

Sea  $\mathbb{P}$  un poset

$\mathbb{P}$  es un predominio  $\Leftrightarrow \forall s$  cadena de  $\mathbb{P}$   $s$  tiene supremo

$\mathbb{P}$  es un dominio  $\Leftrightarrow \mathbb{P}$  es un predominio  $\wedge \mathbb{P}$  tiene mínimo

## Funciones monótonas y continuas:

Sean:

$(P, \leq_P), (Q, \leq_Q)$  posets

$f : P \rightarrow Q$

$f$  es monotona  $\Leftrightarrow \forall x, y \in P . x \leq_P y \Rightarrow f(x) \leq_Q f(y)$

Sean  $P$  y  $Q$  predominios

$f$  es continua  $\Leftrightarrow$

$\forall p_0 p_1 \dots$  cadena de  $P$ .

$\{f(p_0), f(p_1), \dots\}$  tiene supremo  $\wedge f(\sup\{p_0, p_1, \dots\}) = \sup\{f(p_0), f(p_1), \dots\}$

$[P \rightarrow Q] = \{f \in (P \rightarrow Q) : f \text{ es continua}\}$

Sean  $P$  y  $Q$  dominios

$f$  es estricta  $\Leftrightarrow f(\perp_P) = \perp_Q$

## Extensiones:

Sean:

$P, P'$  predominios

$D$  un dominio

$f : P \rightarrow P'$

$g : P \rightarrow D$

$f_\perp : P_\perp \rightarrow P'_\perp$

$f_\perp \perp = \perp$

$f_\perp p = f p$

$g_\perp : P_\perp \rightarrow D$

$g_\perp \perp = \perp_D$

$g_\perp p = g p$

## Comandos:

```

⟨comm⟩ ::= skip | ⟨var⟩ := ⟨intexp⟩ | ⟨comm⟩; ⟨comm⟩
        | if ⟨boolexp⟩ then ⟨comm⟩ else ⟨comm⟩
        | newvar ⟨var⟩ := ⟨intexp⟩ in ⟨comm⟩
        | while ⟨boolexp⟩ do ⟨comm⟩
    
```

$\Sigma = (\langle var \rangle \rightarrow \mathbb{Z})$

$$\begin{aligned}
\llbracket \cdot \rrbracket \langle comm \rangle &\rightarrow \Sigma \rightarrow \Sigma_\perp \\
\llbracket \text{skip} \rrbracket \sigma &= \sigma \\
\llbracket v := e \rrbracket \sigma &= [\sigma \mid v : \llbracket e \rrbracket \sigma] \\
\llbracket c_0 ; c_1 \rrbracket \sigma &= \llbracket c_1 \rrbracket \llbracket c_0 \rrbracket \sigma \\
\llbracket \text{if } e \text{ then } c_0 \text{ else } c_1 \rrbracket \sigma &= \begin{cases} \llbracket e \rrbracket \sigma & \rightarrow \llbracket c_0 \rrbracket \sigma \\ \text{si no} & \rightarrow \llbracket c_1 \rrbracket \sigma \end{cases} \\
\llbracket \text{newvar } v := e \text{ in } c \rrbracket \sigma &= \text{rest}_{v,\sigma} \llbracket c \rrbracket \sigma \mid v : \llbracket e \rrbracket \sigma] \\
\text{rest}_{v,\sigma} &: \Sigma \rightarrow \Sigma \\
\text{rest}_{v,\sigma} \sigma' &= [\sigma' \mid v : \sigma \ v] \\
\llbracket \text{while } b \text{ do } c \rrbracket \sigma &= \left( \coprod_{i \in \mathbb{N}} F^i \perp_{\Sigma \rightarrow \Sigma_\perp} \right) \sigma \\
F : (\Sigma \rightarrow \Sigma_\perp) &\rightarrow \Sigma \rightarrow \Sigma_\perp \\
F f \sigma' &= \begin{cases} \llbracket b \rrbracket \sigma' & \rightarrow f \llbracket c \rrbracket \sigma' \\ \text{si no} & \rightarrow \sigma' \end{cases}
\end{aligned}$$

### Lenguaje imperativo con fallas:

$\langle comm \rangle ::= \dots \mid \text{fail} \mid \text{catchin } \langle comm \rangle \text{ with } \langle comm \rangle$

$$\begin{aligned}
\hat{\Sigma} &= \Sigma \cup (\{\text{abort}\} \times \Sigma) \\
\Box_* : (\Sigma \rightarrow \hat{\Sigma}_\perp) &\rightarrow \hat{\Sigma}_\perp \rightarrow \hat{\Sigma}_\perp \\
f_* \sigma &= \begin{cases} \sigma \in \Sigma & \rightarrow f \sigma \\ \text{si no} & \rightarrow \sigma \end{cases} \\
\Box_\dagger : (\Sigma \rightarrow \Sigma) &\rightarrow \hat{\Sigma}_\perp \rightarrow \hat{\Sigma}_\perp \\
f_\dagger \sigma &= \begin{cases} \sigma \in \Sigma & \rightarrow f \sigma \\ \sigma \in (\{\text{abort}\} \times \Sigma) & \rightarrow \langle \text{abort}, f \sigma_1 \rangle \\ \text{si no} & \rightarrow \perp \end{cases} \\
\Box_+ : (\Sigma \rightarrow \hat{\Sigma}_\perp) &\rightarrow \hat{\Sigma}_\perp \rightarrow \hat{\Sigma}_\perp \\
f_+ \sigma &= \begin{cases} \sigma \in (\{\text{abort}\} \times \Sigma) & \rightarrow f \sigma_2 \\ \text{si no} & \rightarrow \sigma \end{cases}
\end{aligned}$$

La definición es la misma de antes pero cambiando los  $\Sigma_\perp$  por  $\hat{\Sigma}_\perp$  y los  $\llbracket \cdot \rrbracket$  por  $\llbracket \cdot \rrbracket_*$  salvo en newvar en donde se lo cambia por  $\dagger$  y agregando:

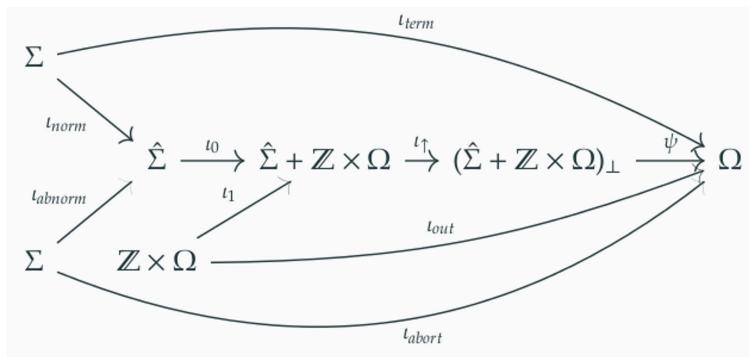
$$\begin{aligned}
\llbracket \text{fail} \rrbracket \sigma &= \langle \text{abort}, \sigma \rangle \\
\llbracket \text{catchin } c_0 \text{ with } c_1 \rrbracket \sigma &= \llbracket c_1 \rrbracket_+ (\llbracket c_0 \rrbracket \sigma)
\end{aligned}$$

### Imperativo con input:

$\langle comm \rangle ::= \dots \mid ! \langle intexp \rangle$

$$\Omega = (\hat{\Sigma} + \mathbb{Z} \times \Omega)$$

$$\begin{aligned}
\Box_* : (\Sigma \rightarrow \Omega) &\rightarrow \Omega \rightarrow \Omega \\
f_*(\iota_{term} \sigma) &= f \sigma
\end{aligned}$$



$$\begin{aligned}\square_* &: (\Sigma \rightarrow \Omega) \rightarrow \Omega \rightarrow \Omega \\ f_*(\iota_{term} \sigma) &= f \sigma \\ f_*(\iota_{out}(n, \omega)) &= \iota_{out}(n, f_* \omega) \\ f_* \omega &= \omega\end{aligned}$$


$$\begin{aligned}\square_\dagger &: (\Sigma \rightarrow \Sigma) \rightarrow \Omega \rightarrow \Omega \\ f_\dagger(\iota_{term} \sigma) &= \iota_{term}(f \sigma) \\ f_\dagger(\iota_{out}(n, \omega)) &= \iota_{out}(n, f_\dagger \omega) \\ f_\dagger(\iota_{abort} \sigma) &= \iota_{abort}(f \sigma) \\ f_\dagger \perp &= \perp\end{aligned}$$

$$\begin{aligned}\square_+ &: (\Sigma \rightarrow \Omega) \rightarrow \Omega \rightarrow \Omega \\ f_+(\iota_{abort} \sigma) &= f \sigma \\ f_+ \omega &= \omega\end{aligned}$$

$$\begin{aligned}\llbracket \langle comm \rangle &\rightarrow \Sigma \rightarrow \Omega \\ \llbracket !x \rrbracket \sigma &= \iota_{out}(\sigma x, \sigma)\end{aligned}$$

### Agregando input:

$$\langle comm \rangle ::= \dots \mid ? \langle var \rangle$$

$$\Omega = (\hat{\Sigma} + \mathbb{Z} \times \Omega + \mathbb{Z} \rightarrow \Omega)$$

$$\begin{aligned}\square_* &: (\Sigma \rightarrow \Omega) \rightarrow \Omega \rightarrow \Omega \\ f_*(\iota_{term} \sigma) &= f \sigma \\ f_*(\iota_{out}(n, \omega)) &= \iota_{out}(n, f_* \omega) \\ f_*(\iota_{in}(g)) &= \iota_{in}(\lambda n . f_*(g n)) \\ f_* \omega &= \omega\end{aligned}$$

$$\begin{aligned}\square_\dagger &: (\Sigma \rightarrow \Sigma) \rightarrow \Omega \rightarrow \Omega \\ f_\dagger(\iota_{term} \sigma) &= \iota_{term}(f \sigma) \\ f_\dagger(\iota_{out}(n, \omega)) &= \iota_{out}(f_\dagger \omega) \\ f_\dagger(\iota_{abort} \sigma) &= \iota_{abort}(f \sigma) \\ f_\dagger(\iota_{in}(g)) &= \iota_{in}(\lambda n . f_\dagger(g n)) \\ f_\dagger \perp &= \perp\end{aligned}$$

$$\llbracket ?x \rrbracket \sigma = \iota_{in}(\lambda n . [\sigma \mid x : n])$$

1. Determinar si es verdadero o falso. Justificar la respuesta.

- Sean  $p, q$  predicados. Si  $\llbracket p \rrbracket = \llbracket q \rrbracket$ , entonces para toda sustitución  $\delta$  se tiene  $\llbracket p/\delta \rrbracket = \llbracket q/\delta \rrbracket$ .
- Sea  $\Omega$  el dominio del lenguaje imperativo con fallas y output. Si  $\sigma$  es un estado, entonces existe una cadena interesante que tiene como supremo a  $\iota_{out}(1, \iota_{term}\sigma)$ .
- Sea  $f, g \in \mathbf{Z} \rightarrow \mathbf{Z}_\perp$ . Entonces existe  $h \in \mathbf{Z} \rightarrow \mathbf{Z}_\perp$  tal que  $f \leq h$  y  $g \leq h$ .
- En el lenguaje imperativo simple, si  $\llbracket c \rrbracket \sigma = \langle \text{abort}, \sigma' \rangle$ , entonces

$$\llbracket \text{catchin } c \text{ with } c' \rrbracket \sigma = \llbracket c; c' \rrbracket \sigma.$$

a) Verdadero

Demostración suponiendo el antecedente:

$$\begin{aligned}
 & \llbracket p/\delta \rrbracket \sigma \\
 &= \llbracket p \rrbracket \llbracket x : \llbracket \delta x \rrbracket \sigma \quad \forall x \in \langle \text{var} \rangle \rrbracket \quad \xrightarrow{\text{Lema auxiliar 2}} \\
 &= \llbracket q \rrbracket \llbracket x : \llbracket \delta x \rrbracket \sigma \quad \forall x \in \langle \text{var} \rangle \rrbracket \quad \xrightarrow{\text{Antecedente}} \\
 &= \llbracket q/\delta \rrbracket \sigma \quad \xrightarrow{\text{Lema auxiliar 2}}
 \end{aligned}$$

Lema auxiliar 1:

Se da:

$$e \in \langle \text{intexp} \rangle$$

$$\delta \in (\langle \text{var} \rangle \rightarrow \langle \text{intexp} \rangle)$$

$$\sigma \in \Sigma$$

$$\llbracket e/\delta \rrbracket \sigma = \llbracket e \rrbracket \llbracket x : \llbracket \delta x \rrbracket \sigma \quad \forall x \in \langle \text{var} \rangle \rrbracket$$

Se demuestra fácilmente por inducción

Lema auxiliar 2:

Son:

$$p \in \langle \text{assert} \rangle$$

$$\delta \in (\langle \text{var} \rangle \Rightarrow \langle \text{intexp} \rangle)$$

$$\sigma \in \Sigma$$

$$[\![p/\delta]\!] \sigma = [\![p]\!][x : [\![\delta x]\!] \sigma \quad \forall x \in \langle \text{var} \rangle]$$

Se demuestra fácilmente por inducción usando el lema auxiliar 1

b) Falso

Los únicos elementos menores a  $\text{Lout}(t, \text{term } \sigma)$  son:

$$\text{Lout}(t, \perp)$$

$\perp$

c) Falso, por ejemplo:

$$f \times = 0$$

$$g \times = 1$$

d) Falso

Si  $c = \text{fail}$ ,  $c' = \text{skip}$  tenemos:

$$\llbracket \text{catchin } c \text{ with } c' \rrbracket \sigma = \sigma$$

$$\llbracket c ; c' \rrbracket \sigma = \langle \text{abort}, \sigma \rangle$$

2. Considere el lenguaje aplicativo con fallas, output e input. Analice utilizando la semántica denotacional la equivalencia entre los siguientes comandos:

- a) **newvar**  $v := e \text{ in } ?v; !v \equiv ?v; !v$   
 b) Si  $FA\ c \cap FA\ c' = \emptyset$  entonces  $c; c' \equiv c'; c$

d) Falso

Porque:

$$\llbracket \text{newvar } v := e \text{ in } ?v; !v \rrbracket \sigma = \text{in}(\lambda x. \text{out}(x, \sigma))$$

$$\llbracket ?v \text{ j!v} \rrbracket \sigma = \text{in}(\lambda x. \text{out}(x, [\sigma \mid v : x]))$$

b) Falso

Por ejemplo:

$$c = x := y$$

$$c' = y := x$$

Tenemos:

$$[\underline{c}; c']\sigma = [\sigma \mid x : \sigma y]$$

$$[\underline{c'}; c]\sigma = [f \mid y : f x]$$

3. Considere el lenguaje imperativo simple.

- Dé la semántica denotacional de **while**  $b$  **do**  $c$ .
- Pruebe que la función  $F$  que define la semántica de **while**  $b$  **do**  $c$  es continua.
- De ejemplo de un comando  $c$  de la forma **while**  $b$  **do**  $c$  tal que  $[c] = F^3 \perp_{\Sigma \rightarrow \Sigma_\perp}$  pero  $[c] \neq F^2 \perp_{\Sigma \rightarrow \Sigma_\perp}$ .

a)

$$[\text{while } b \text{ do } c]\sigma = \bigsqcup_{i \in N} F^i \perp \sigma$$

Donde:

$$F : (\Sigma \rightarrow \Sigma_\perp) \rightarrow \Sigma \rightarrow \Sigma_\perp$$

$$F f \sigma = \begin{cases} [\underline{b}]\sigma \rightarrow f \sigma \\ \text{si no } \rightarrow \sigma \end{cases}$$

b)

Serán  $g_0, g_1, \dots$  una cadena de  $\Sigma \Rightarrow \Sigma_\perp$

Hay que probar:

$$F \left( \bigsqcup_{i \in N} g_i \right) = \bigsqcup_{i \in N} F g_i$$

Figurando  $\sigma \in \Sigma$  hay que probar:

$$\begin{aligned} F\left(\bigsqcup_{i \in N} g_i\right) \sigma &= \left(\bigsqcup_{i \in N} Fg_i\right) \sigma \\ \Leftrightarrow F\left(\bigsqcup_{i \in N} g_i\right) \sigma &= \bigsqcup_{i \in N} Fg_i \sigma \end{aligned} \quad \Rightarrow \text{Supremo de funciones}$$

(así  $\Rightarrow$   $\llbracket b \rrbracket \sigma$ :

$$\begin{aligned} F\left(\bigsqcup_{i \in N} g_i\right) \sigma &\quad \text{Def } F \\ = \sigma & \\ = \bigsqcup_{i \in N} \sigma &\quad \text{Def } F \\ = \bigsqcup_{i \in N} Fg_i \sigma & \end{aligned}$$

(así  $\llbracket b \rrbracket \sigma$ :

$$\begin{aligned} F\left(\bigsqcup_{i \in N} g_i\right) \sigma &\quad \text{Def } F \\ = \left(\bigsqcup_{i \in N} g_i\right) \sigma & \\ = \bigsqcup_{i \in N} g_i \sigma &\quad \text{Def } F \\ = \bigsqcup_{i \in N} Fg_i \sigma & \end{aligned}$$

c)

while  $b \neq 0$  do

if  $x > 2 \vee x \leq 0$

then  $x := 2$

else  $x := x - 1$

5. Considere la función  $F : (\mathbf{Z} \rightarrow \mathbf{Z}_+) \rightarrow (\mathbf{Z} \rightarrow \mathbf{Z}_+)$  dada por:

$$Ffn = \begin{cases} n & n = 0, 1, 2 \\ f(n-3) & n > \cancel{X}2 \\ f(-n) & n < 0 \end{cases}$$

a) ¿Cuánto vale  $F^5 \perp_{\mathbf{Z} \rightarrow \mathbf{Z}_+} (-10)$ ?

b) ¿Cuánto vale el menor punto fijo de  $F$  en  $-10$ ? Justifique su respuesta.

c) Justifique la siguiente afirmación:  $F^2 \perp_{\mathbf{Z} \rightarrow \mathbf{Z}_+} \leq F^3 \perp_{\mathbf{Z} \rightarrow \mathbf{Z}_+}$ .

d) Pruebe que  $F$  es continua.

2)



$$F^5 \perp (-10)$$

$$= F(F^4 \perp)(-10)$$

$$= (F^4 \perp) 10$$

$$= F(F^3 \perp) 10$$

$$= (F^3 \perp)(10 - 3)$$

$$= F(F^2 \perp) 7$$

$$= (F^2 \perp)(7 - 3)$$

$$= F(F \perp) 4$$

$$= F \perp (4-3)$$

$$= 1$$

b) Vale 1 porque el cálculo con un punto fijo se desarrolla igual que en  $\mathbb{R}$

c) Es porque sigue dando lo mismo en donde  $d_{\text{bbo}} \neq 1$  y algunos valores para los que  $d_{\text{bbo}} \perp$  lejos de darlo

d)

Ser  $g_0, g_1, \dots$  una familia de  $\mathbb{Z} \rightarrow \mathbb{Z}_+$

# Parcial 1 2018

Lunes, 22 de abril de 2024 10:41

2. Determinar si es verdadero o falso. Justificar la respuesta.

- (a) Si  $P$  es un poset finito con menor elemento  $\perp$ , y  $F \in P \rightarrow P$  es monótona creciente, entonces  $F$  tiene un punto fijo.
- (b) Sea  $p$  predicado y  $\delta$  una sustitución. Si  $\llbracket p/\delta \rrbracket \sigma = V$ , entonces existe  $\sigma'$  tal que  $\llbracket p \rrbracket \sigma' = V$ .
- (c) La cadena  $F^0 \perp_{\Sigma \rightarrow \Sigma_\perp}, F^1 \perp_{\Sigma \rightarrow \Sigma_\perp}, \dots$  correspondiente a un programa de la forma **while true do**  $c$  es siempre interesante.

a) Verdadero

Porque cualquier  $x$  maximal tiene que ser punto fijo  
y al ser finito tiene que tener maximal

b) Verdadera

En particular  $\Gamma(x) \supseteq \llbracket s x \rrbracket \sigma$

c) Falso

Cuando  $c = \text{while true do skip}$

La cadena da todo  $\perp$

3. Considere el dominio  $D = (\mathbf{Z} \rightarrow \mathbf{Z}_\perp)$ . Justifique las respuestas.

- (a) Muestre una función  $F \in D \rightarrow D$  que satisfaga simultáneamente:
  - (i) tiene infinitos puntos fijos;
  - (ii) posee un menor punto fijo  $h$ ;
  - (iii)  $h(x)$  es distinto de  $\perp$  en los enteros negativos.
- (b) Muestre una función  $F \in D \rightarrow D$  que tenga puntos fijos pero no tenga un menor punto fijo.

a)

$$F f x = \begin{cases} x < 0 \Rightarrow 0 \\ \text{Si } h_0 \Rightarrow f(x) \end{cases}$$

Cumple i ya que  $\forall f \in (Z \rightarrow Z_+) . (\forall x < 0 . f x = 0) \Rightarrow f$  es punto fijo de  $F$

Cumple ii ya que el menor punto fijo es:

$$h x = \begin{cases} x < 0 \Rightarrow 0 \\ \text{Si } h_0 \Rightarrow \perp \end{cases}$$

Claramente  $h$  cumple iii

b)

$$F f x = \begin{cases} f x = \perp \Rightarrow 0 \\ \text{Si } h_0 \Rightarrow f x \end{cases}$$

id y  $\lambda x. x + 1$  son claramente puntos fijos

pero son incomparables entre si

$\Rightarrow$  No tiene menor punto fijo

4. Considere el lenguaje imperativo simple. Enuncie el Teorema de Coincidencia, y luego utilice el mismo para probar que si  $v$  no es libre en  $c$ , entonces los comandos  $c$  y **newvar**  $v := e$  in  $c$  son equivalentes.

Teorema de coincidencia (TC):

Sedh:

$$c \in \langle \text{com} \rangle$$

$$\sigma, \sigma' \in \Sigma \text{ tales que } \forall v \notin FV(c) . \sigma v = \sigma' v$$

Entonces:

$$1) \llbracket c \rrbracket_\sigma = \perp \Leftrightarrow \llbracket c \rrbracket_{\sigma'} = \perp$$

$$2) \llbracket c \rrbracket_\sigma \neq \perp \Rightarrow \forall v \notin FV(c) . \llbracket c \rrbracket_\sigma v = \llbracket c \rrbracket_{\sigma'} v$$

Queremos provar:

Sedh:

$$c \in \langle \text{com} \rangle$$

$$v \in \langle \text{var} \rangle - FV(c)$$

$$e \in \langle \text{intexp} \rangle$$

$$c \equiv \text{newvar } v := e \text{ in } c$$

Demonstracióh, se d  $\sigma \in \Sigma, x \in \langle \text{var} \rangle$ :

$$\llbracket \text{newvar } v := e \text{ in } c \rrbracket_\sigma x = \llbracket c \rrbracket_\sigma x$$

$$\Leftrightarrow \text{rest}_{v, \sigma v} (\llbracket c \rrbracket [\sigma / v := \llbracket e \rrbracket \sigma]) = \llbracket c \rrbracket_\sigma x$$

El caso  $x = v$  es claramente cierto por el resto

Si  $x \neq v$ :

$$\text{rest}_{v,v} ([c][\sigma] \vee [e][\sigma])x = [c]\sigma x$$

$$\Leftrightarrow [c][\sigma \mid v : [e]\sigma]x = [c]\sigma x$$

Si  $x \notin FV(c)$  esto es cierto por el TC

Si no es cierto porque:

$$\forall v \notin FV(c). \sigma v = [c]\sigma v$$

Y a que si  $v$  no aparece libre en  $c$ , entonces  $c$  no puede modificar  $v$

5. Considere el comando

**while**  $x \neq 0$  **do if**  $x > 10$  **then**  $x := -x$  **else**  $x := x - 1$ .

(a) Muestre la función  $F$  que define la semántica denotacional del while, expresándola de la manera más sencilla posible.

(b) ¿Qué función de  $\Sigma \rightarrow \Sigma_\perp$  es la función  $F(\lambda\sigma' \in \Sigma. [\sigma' | x : 0])$ ?

(c) Caracterice el conjunto de estados que satisfacen  $F^3 \perp_{\Sigma \rightarrow \Sigma_\perp} \sigma = [c]\sigma$

a) 
$$F f \sigma = \begin{cases} x = 0 \rightarrow \sigma \\ x > 10 \rightarrow f[\sigma \mid x : \neg \sigma x] \\ \text{si no } \rightarrow f[\sigma \mid x : \sigma x - 1] \end{cases}$$

b)

$$\lambda \sigma \in \Sigma. [\sigma | x:0]$$

c)

$$\{\sigma \in \Sigma : \sigma x \leq 0 \vee \sigma x > 10\}$$

# Parcial 1 2022-4-13

sábado, 20 de abril de 2024 16:55

- (1) Determinar si las siguientes afirmaciones son verdaderas o falsas. Justificar acabadamente sus respuestas.
- Sea  $p$  el predicado  $\forall x. (1 < x \vee \exists y. z = y + 1) \wedge \exists y. ((\forall z. z + x = y) \rightarrow z > y)$  y sea  $\delta = [z \rightarrow z + 1, y \rightarrow x, x \rightarrow (x + y)]$ . Entonces al hacer  $p/\delta$ , en todos los cuantificadores  $\forall v.p$  se puede tomar  $v = v_{new}$
  - Para toda  $g$ , la cadena  $f_0, f_1, \dots$  es interesante, donde  $f_i \in \mathbb{Z} \rightarrow \mathbb{Z}_\perp$  se define mediante  $f_i n = \begin{cases} g n & -i \leq n \leq i \\ \perp & \text{cc} \end{cases}$
  - Si  $c_w = \text{while true do } c$ , entonces en el lenguaje imperativo con fallas se satisface  $\llbracket c_w \rrbracket = \llbracket c; c_w \rrbracket$
  - Si  $\mathbb{Z}$  tiene el orden discreto, entonces  $(\mathbb{Z} \rightarrow \mathbb{Z}_\perp)$  tiene infinitos elementos maximales.

a) Falso

Si hicieramos eso la y del  $\exists y$  atropellaría a la y del reemplazo  $x \rightarrow (x + y)$

b) Falso

Si  $g n = \perp$  tenemos que

$\forall i, f_i n = \perp$

y por ende  $f_0 = f_1 = \dots$

c) Es verdadero porque  $c_w$  es esencialmente  $c; c; c; \dots$

entonces agregar una  $c$  antes de infinitos  $c$  no cambia en nada

d) Verdadero

Por ejemplo contiene la siguiente familia de elementos

maximales:

$$f_i : \mathbb{Z} \rightarrow \mathbb{Z}_+$$

$$f_i(x) = \begin{cases} x = i \rightarrow 1 \\ \text{si } n_0 \rightarrow 0 \end{cases}$$

Para que para  $i \neq j$   $f_i$  es incomparable con  $f_j$   
porque 0 y 1 son incomparables

(2) Sea  $h \in \mathbb{Z} \rightarrow \mathbb{Z}_+$ . Considere la siguiente ecuación recursiva:

$$f_n = \begin{cases} h & n \leq 0 \vee n \text{ es impar} \\ f(n-2) & \text{si } n > 0 \wedge n \text{ es par} \end{cases}$$

Sea  $F_h$  la función asociada a la ecuación recursiva, cuyo menor punto fijo es la menor solución.

- (a) Escriba la definición explícita de  $F_h$ .
- (b) Defina  $h_0, h_1 \in \mathbb{Z} \rightarrow \mathbb{Z}_+$  tales que:  
 $F_{h_0} h_0 = h_0$  (o sea,  $F_{h_0}$  tiene a  $h_0$  como punto fijo), y  
 $h_1$  no es punto fijo de  $F_{h_1}$ .
- (c) Calcule  $F_h^3 : \mathbb{Z} \rightarrow \mathbb{Z}_+$  de la manera más clara posible.
- (d) Sin necesidad de calcularlo, proponga una expresión para la función  $\cup_{k \geq 0} F_h^k : \mathbb{Z} \rightarrow \mathbb{Z}_+$ . Describa en castellano qué denota ese supremo (en términos de  $h$ ). Vale intentar esto aunque no haya escrito una expresión matemática para el supremo.
- (e) El calculado en (d), ¿es el único punto fijo, o hay más?

2)

$$F_h : (\mathbb{Z} \rightarrow \mathbb{Z}_+) \rightarrow \mathbb{Z} \rightarrow \mathbb{Z}_+$$

$$F_h f n = \begin{cases} n \leq 0 \vee n \text{ es impar} \rightarrow h n \\ \text{si } n_0 \rightarrow f(n-2) \end{cases}$$

b)

$$h_0, h_1 : \mathbb{Z} \rightarrow \mathbb{Z}_+$$

$$h_0 x = 0$$

$$h_1 x = x$$

c)

$$F_h \perp n$$

$$= \begin{cases} n \leq 0 \vee h \text{ es impar} \Rightarrow h n \\ \quad \text{si } h_0 \\ \quad \Rightarrow \perp (n-2) \end{cases}$$

$$= \begin{cases} n \leq 0 \vee h \text{ es impar} \Rightarrow h n \\ \quad \text{si } h_0 \\ \quad \Rightarrow \perp \end{cases}$$

$$F_h^2 \perp n$$

$$= F_h (F_h \perp) n$$

$$= \begin{cases} n \leq 0 \vee h \text{ es impar} \Rightarrow h n \\ \quad \text{si } h_0 \\ \quad \Rightarrow (F_h \perp) (n-2) \end{cases}$$

$$= \begin{cases} n \leq 0 \vee h \text{ es impar} \Rightarrow h n \\ n-2 \leq 0 \vee \cancel{(n-2)} \text{ es impar} \Rightarrow h(n-2) \\ \quad \text{si } h_0 \\ \quad \Rightarrow \perp \end{cases}$$

$$F_h^3 \perp n$$

$$= F_h (F_h^2 \perp) n$$

$$= \begin{cases} n \leq 0 \vee n \text{ es impar} \Rightarrow h n \\ \text{si } h_0 \end{cases} \rightarrow (F_h^2 \perp) n$$

$$= \begin{cases} n \leq 0 \vee h \text{ es impr} \Rightarrow h n \\ h - 2 \leq 0 \\ n - 1 \leq 0 \\ \text{Si } h_0 \end{cases} \rightarrow \begin{cases} h (h - 2) \\ h (n - 1) \\ \perp \end{cases}$$

1)

$$\left( \bigsqcup_{k \geq 0} F_h^k \perp \right) n = \begin{cases} n \leq 0 \vee n \text{ es impar} \Rightarrow h n \\ \text{si } h_0 \end{cases} \rightarrow h 0$$

No entiendo que mas quiere que describa en castellano

(3) Analice la equivalencia de los siguientes comandos. Utilice la semántica denotacional para justificar la respuesta.

1 newvar  $x := 0$  in catchin (while  $x = 0$  do fail) with skip  $\vdash C''$   
 2 skip  $\vdash C'$

$C \equiv \text{skip}$

Demos fndrión:

$\llbracket C \rrbracket \sigma$

$$\begin{aligned} &= \text{rest}_{x, \sigma +} (\llbracket C' \rrbracket [\sigma | x : 0]) \\ &= \text{rest}_{x, \sigma +} (\llbracket \text{skip} \rrbracket_+ (\llbracket C' \rrbracket [\sigma | x : 0])) \\ &= \text{rest}_{x, \sigma +} (\llbracket \text{skip} \rrbracket_+ (\text{abort}, [\sigma | x : 0])) \\ &= \text{rest}_{x, \sigma +} (\llbracket \text{skip} \rrbracket [\sigma | x : 0]) \\ &= \text{rest}_{x, \sigma +} [\sigma | x : 0] \\ &= \sigma \\ &= \llbracket \text{skip} \rrbracket \sigma \end{aligned}$$

4) Enunciar un Teorema de Coincidencia para LIS + FALLAS (sólo enunciarlo)

Señ:

$c \in \langle \text{comm} \rangle$

$\sigma, \sigma' \in \Sigma : \forall v \in \text{fv}(c) . \sigma v = \sigma' v$

Ejemplos:

$$[c]_{\sigma} = \perp \Leftrightarrow [c]_{\sigma'} = \perp$$

$$[c]_{\sigma} = \langle \text{abort}, s \rangle \Leftrightarrow [c]_{\sigma'} = \langle \text{abort}, s' \rangle$$

↳ en este caso  $\forall v \in \text{fv}(c) . sv = s'v$

(o contrario:

$$\forall v \in \text{fv}(c) . [c]_{\sigma} v = [c]_{\sigma'} v$$

# Parcial 1 2023-05-03

domingo, 21 de abril de 2024 10:10

- La siguiente gramática abstracta corresponde a un lenguaje para dar órdenes a un robot.

|  |  |
|--|--|
| $\langle ord \rangle ::= \text{mover } \langle int \rangle$            | ver ejemplo  |
| girar  | intercambia la dirección actual                    |
| si pos = $\langle coord \rangle$ hacer $\langle ord \rangle$           | ejecuta la orden si la posición actual es la dada  |
| si dir = $\langle dir \rangle$ hacer $\langle ord \rangle$             | ejecuta la orden si la dirección actual es la dada |
| $\langle ord \rangle; \langle ord \rangle$                             | ejecuta la primera orden y luego la segunda        |
| $\langle int \rangle ::= \dots   -2   -1   0   1   2   \dots$          |  |
| $\langle dir \rangle ::= \text{EO}   \text{NS}$                        |  |
| $\langle coord \rangle ::= (\langle int \rangle, \langle int \rangle)$ |  |

Sea  $D = \{\text{NS}, \text{EO}\}$  el conjunto de direcciones y sea  $\Sigma = (\mathbb{Z} \times \mathbb{Z}) \times D$  el conjunto de estados. Escribí las ecuaciones semánticas con el siguiente tipo:  $\llbracket \_ \rrbracket : \langle ord \rangle \rightarrow \Sigma \rightarrow \Sigma$ . Por ejemplo,

$$\llbracket \text{mover } k \rrbracket((x, y), d) = \begin{cases} ((x + k, y), d) & \text{si } d = \text{EO} \\ ((x, y + k), d) & \text{si } d = \text{NS} \end{cases}$$

$$\llbracket \_ \rrbracket : \langle ord \rangle \rightarrow \Sigma \rightarrow \Sigma$$

$$\llbracket \text{mover } k \rrbracket((x, y), d) = \begin{cases} d = \text{EO} \rightarrow ((x + k, y), d) \\ \text{si no} \rightarrow ((x, y + k), d) \end{cases}$$

$$\llbracket \text{girar} \rrbracket(v, d) = \begin{cases} d = \text{EO} \rightarrow (v, \text{NS}) \\ \text{si no} \rightarrow (v, \text{EO}) \end{cases}$$

$$\llbracket \text{si pos} = v \text{ hacer } o \rrbracket(v', d) = \begin{cases} v = v' \rightarrow \llbracket o \rrbracket(v', d) \\ \text{si no} \rightarrow (v', d) \end{cases}$$

$$\llbracket \text{si dir} = d \text{ hacer } o \rrbracket(v, d') = \begin{cases} d = d' \rightarrow \llbracket o \rrbracket(v, d') \\ \text{si no} \rightarrow (v, d') \end{cases}$$

$$\llbracket o ; o' \rrbracket_\sigma = \llbracket o' \rrbracket(\llbracket o \rrbracket_\sigma)$$

2. Considerá la siguiente ecuación recursiva:

$$f(x) = \begin{cases} 0 & \text{si } x = 0 \\ 8 - f(x-2) & \text{si } x \neq 0 \end{cases}$$

Sea  $F: (\mathbb{Z} \rightarrow \mathbb{Z}_{\perp}) \rightarrow (\mathbb{Z} \rightarrow \mathbb{Z}_{\perp})$  el funcional asociado a esa ecuación.

¿Existe  $x \in \mathbb{Z}$  tal que  $F^3(\perp)(x) = 10$ ?

No existe

3. Decidí si cada una de las siguientes afirmaciones es verdadera o falsa. Justificá tu respuesta.

(a) En cualquier predominio infinito siempre hay cadenas interesantes.

(b) Sea  $f: P \rightarrow P'$  una función continua entre los predominios  $P$  y  $P'$ , entonces  $f(\sqcup_i x_i) \leq \sqcup_i (f(x_i))$ , asumié que  $x_i$  es una cadena interesante.

a) Falso

por ejemplo en  $\mathbb{Z}$  con el orden discreto no hay  
y d que todas las cadenas son repeticiones del mismo  
elemento

b) Verdadero

por definición de función continua:

$$f(\bigcup_i x_i) \supseteq \bigcup_i f(x_i)$$

$\Rightarrow$

$$f(\bigcup_i x_i) \leq \bigcup_i f(x_i)$$

4. Considerá el lenguaje imperativo simple con fallas. Sea  $c$  el programa siguiente

while  $x \neq 0$  do if  $x > 0$  then  $d := 1 + d$ ;  $x := x - 1$  else fail

(a) Escribí de la forma más sencilla posible la ecuación para  $F(f)(\sigma)$  donde  $F$  es el funcional asociado al ciclo de ese programa.

(b) Proponé un estado  $\sigma$  (dando los valores de  $x$  y  $d$ ) tal que  $F^1(\perp)(\sigma) \neq \perp$ .

a)

$$F_f \sigma \equiv \begin{cases} x = 0 & \rightarrow \sigma \\ x > 0 & \rightarrow f[\sigma \mid d : d+1, x : x-1] \\ \text{si no} & \rightarrow (\text{abort}, \sigma) \end{cases}$$

b)

$$\sigma = \{x:0, d:0\}$$

Luego  $f^1 \perp \sigma = \sigma$

5. Decidí si las siguientes equivalencias son correctas. Si no lo son proponé contraejemplos concretos. Si lo son, hacé la demostración.

- (a)  $\text{catchin } c \text{ with } (\text{fail}; c') \equiv c; c'$ .
- (b)  $\text{catchin } (c; \text{fail}) \text{ with } c' \equiv \text{catchin } c \text{ with } c'$ .

Recordá que en  $\text{catchin } c \text{ with } c'$  se ejecuta  $c$  y si se produce una falla, entonces se ejecuta  $c'$  en el estado donde se produjo la falla.

a) Falso

Si tenemos:

$$c = c' = 5 \text{ Kip}$$

Tenemos

$$\llbracket \text{catchin } (c \text{ with } (\text{fail}; c')) \rrbracket \sigma \simeq \langle \text{abort}, \sigma \rangle$$

$$\llbracket c; c' \rrbracket \sigma = \sigma$$

b) Falso, por ejemplo:

$$c = 5 \text{ Kip}$$

$$c' = x := 0$$

Tenemos:

$$\begin{aligned} &\llbracket \text{catchin } ((c \text{ fail}) \text{ with } c') \rrbracket \sigma \\ &= \llbracket c' \rrbracket_+ (\llbracket c \text{ fail} \rrbracket \sigma) \end{aligned}$$

$$\begin{aligned}
&= \llbracket c' \rrbracket_+ (\llbracket f \circ i \rrbracket_* (\llbracket c \rrbracket \sigma)) \\
&= \llbracket c' \rrbracket_+ (\llbracket f \circ i \rrbracket \sigma) \\
&= \llbracket c' \rrbracket_+ \langle \text{short}, \sigma \rangle \\
&= \llbracket c' \rrbracket \sigma \\
&= [\sigma | x : 0]
\end{aligned}$$

Pero:

$$\begin{aligned}
&\llbracket \text{catchin } c \text{ with } c' \rrbracket \sigma \\
&= \llbracket c' \rrbracket_+ (\llbracket c \rrbracket \sigma) \\
&= \llbracket c' \rrbracket_+ \sigma \\
&= \sigma
\end{aligned}$$

1. Considere la expresión  $e = (\lambda x. \lambda y. x(e_0 y))(\lambda x. \lambda y. y(xy))(\lambda x. \lambda y. y)$ . Para cada evaluación (normal o eager) determine si se puede definir  $e_0$  de manera que la expresión  $e$  no se pueda evaluar (o sea no está definida la semántica big-step).

En eager si se puede por ejemplo:

$$e_0 = \Delta\Delta$$

E

$$(\lambda x. y. x(e_0 y))(\lambda x. y. y(xy))$$

$$\lambda x. y. x(e_0 y) \Rightarrow_E \dots$$

(pongo... cuando se repite la expresión anterior)

$$\lambda x. y. y(xy) \Rightarrow_E \dots$$

$$\lambda y. (\lambda x. y(xy))(e_0 y) \Rightarrow_E \dots$$

$$\Rightarrow_E \dots$$

$$\lambda x. y \Rightarrow_E \dots$$

$$(\lambda x. y(xy))(e_0 (\lambda x. y))$$

$$\lambda x. y(xy) \Rightarrow_E \dots$$

$$e_0 (\lambda x. y)$$

$$e_0$$

$e_0$  no evalua a nada

En normal no se puede

Vemos que para cualquier  $e_0$  se llega a una forma (ancha):

E

$$(\lambda x. y. x(e_0 y))(\lambda x. y. y(xy))$$

$$\lambda x. y. x(e_0 y) \Rightarrow_N \dots$$

..... 1 0 1 1

$$\lambda x y. x (\rho_0 y) \Rightarrow_N \dots$$

$$\lambda y. (\lambda x y. y (x y)) (\rho_0 y) \Rightarrow_N \dots$$

$$\Rightarrow_N \dots$$

$$(\lambda x y. y (x y)) (\rho_0 (\lambda x y. y))$$

$$\lambda x y. y (x y) \Rightarrow_N \dots$$

$$\lambda y. y ((\rho_0 (\lambda x y. y)) y) \Rightarrow_N \dots$$

$$\Rightarrow_N \dots$$

$$\Rightarrow_N \dots$$

2. Determine si es Verdadero o Falso. Justifique su respuesta. Enuncie todo resultado teórico que utilice.

- Si  $e \rightarrow^* z$  entonces  $e \Rightarrow_N z$  o  $e \Rightarrow_E z$ . (Aquí  $z$  es una forma canónica).
- Si  $e \rightarrow^* e'$  entonces  $[e]^E = [e']^E$ .
- Si existe  $z'$  tal que  $e' \Rightarrow_E z'$ , entonces vale la regla  $\beta$ , o sea las expresiones  $(\lambda v. e)e'$  y  $(e/v \mapsto e')$  tienen la misma semántica eager.

a) Falso

Por ejemplo:

$$e = \lambda x. (\lambda y. y) x$$

$$z = \lambda x. x$$

Tenemos que

$$e \rightarrow^* z \quad \text{y que} \quad e \Rightarrow z \quad \text{y que} \quad (\lambda y. y) x \rightsquigarrow x$$

$$e \Rightarrow_N e \quad \text{y que} \quad e \in \langle \text{(nf)} \rangle$$

$$e \Rightarrow_E e \quad \text{y que} \quad e \in \langle \text{(nf)} \rangle$$

b) Falso

Por ejemplo:

$$e = (\lambda x. x) (\lambda x. x)$$

Por ejemplo:

$$e \equiv (\lambda x. y) (A A)$$

$$e' = \lambda y. y$$

Tenemos que:

$$e \Rightarrow^* e' \quad y \text{ d que } e \rightarrow e'$$

$$\llbracket e \rrbracket^E \eta$$

$$= (\varphi_e(\llbracket \lambda x. y \rrbracket \eta)) \perp (\llbracket A A \rrbracket \eta)$$

$$= (\varphi_e(\llbracket \lambda x. y \rrbracket \eta)) \perp_{\mathbb{E}}$$

$$= \perp_{\mathbb{E}}$$

$$\llbracket e' \rrbracket \eta$$

$$= \zeta_e(\psi(\lambda v \in V^E. \llbracket y \rrbracket[\eta \mid y : v]))$$

$$= \zeta_e(\psi(\lambda v \in V^E. \zeta_e v))$$

$$\neq \perp_{\mathbb{E}}$$

c) Verdadero

Demostración:

$$\text{Supongamos } e' \Rightarrow_E z'$$

Por semántica operacional  $(\lambda v. e)e' \quad y \quad (\lambda v. e)z'$  tienen la misma semántica

Como  $z' \in \langle \text{nf} \rangle$  la semántica es la misma que la de  $e / [v : z']$

Dado que  $e' \Rightarrow_E z' \quad \forall e \quad \llbracket e' \rrbracket^E = \llbracket z' \rrbracket^E$  por ende  $\perp$

$\vdash L^* \vdash e : t$

Dado que  $e' \Rightarrow_e z'$  vale  $\llbracket e' \rrbracket^e = \llbracket z' \rrbracket^e$  por donde por el teorema de sustitución finita y el de coincidencia la semántica es la misma que la de  $e / [v : e']$

3. Considere la expresión  $e = ((\lambda v.e_0)e_1, \langle e_2, e_3 \rangle . 1, e_4)$ , donde  $\llbracket e_0 \rrbracket^\eta = \iota_{norm} z$  y  $\llbracket e_4 \rrbracket^\eta = \perp_D$ . Calcule la semántica denotacional eager y normal de  $e.0$ , suponiendo que  $v \notin FV e_0$ .

Eager:

$$\llbracket e.0 \rrbracket^\eta$$

$$= \left( \lambda t. \begin{cases} 0 < |t| \rightarrow \iota_{norm} t_0 \\ \text{si } n_0 \rightarrow t \text{ y err} \end{cases} \right)_{\text{tuple}*} (\llbracket e \rrbracket^\eta)$$

$$\llbracket e \rrbracket^\eta$$

$$= (\lambda z_0. (\lambda z_1. (\lambda z_2. \text{tuple} \langle z_0, z_1, z_2 \rangle)_* (\llbracket e_1 \rrbracket^\eta))_* \llbracket \langle e_2, e_3 \rangle . 1 \rrbracket^\eta)_* (\llbracket (\lambda v. e_0) e_1 \rrbracket^\eta)$$

$$\llbracket \langle e_2, e_3 \rangle . 1 \rrbracket^\eta$$

$$= \left( \lambda t. \begin{cases} 1 \leq |t| \rightarrow \iota_{norm} t_1 \\ \text{Si } n_0 \rightarrow t \text{ y err} \end{cases} \right)_{\text{tuple}*} (\llbracket \langle e_2, e_3 \rangle \rrbracket^\eta)$$

$$\llbracket \langle e_2, e_3 \rangle \rrbracket^\eta$$

=

2. Considerá el cálculo lambda puro.

- (a) Proponé una expresión  $e$  que cumpla las siguientes condiciones simultáneamente:
  - i. bajo evaluación eager,  $e e'$  diverge para cualquiera expresión  $e'$ .
  - ii. bajo evaluación normal,  $e e'$  evalúa a una forma canónica para cualquier expresión  $e'$ .
- (b) Realizá una reducción de  $e I$  hasta llegar a una abstracción. Recordá que una reducción es una secuencia de  $\beta$ -contracciones.

a)

$$e = \lambda x. (\lambda y. \lambda z. z) \Delta\Delta$$

Prueba de i:

figuemos  $e'$

Si  $e'$  no tiene forma canónica es trivialmente cierto

Si  $e' \Rightarrow_E \hat{e}$ :

$$e e'$$

$$e \Rightarrow_E e$$

$$e' \Rightarrow_E \hat{e}$$

$$(\lambda y. \lambda z. z)(\Delta\Delta)$$

$$\lambda y. \lambda z. z \Rightarrow_E \lambda y. \lambda z. z$$

$\Delta\Delta$  no tiene forma canónica

Prueba de ii:

$$e e'$$

$e e'$

$$e \Rightarrow_N e$$

$$(\lambda y. \lambda z. z)(\Delta\Delta)$$

$$\lambda y. \lambda z. z \Rightarrow_N \lambda y. \lambda z. z$$

$$\lambda z. z \Rightarrow_N \lambda z. z$$

$$\Rightarrow_N \lambda z. z$$

$$\Rightarrow_N \lambda z. z$$

b)

$e I$

$$\Rightarrow (\lambda y. \lambda z. z)(\Delta\Delta)$$

$$\rightarrow \lambda z. z$$

3. La contracción ( $\eta$ ) se especifica como  $\overline{(\lambda x. e x) \rightarrow_\eta e} x \notin FV(e)$ .

Considerá la semántica denotacional normal del cálculo lambda. Recordá que  $V^N \cong [D^N \rightarrow D^N]$  y  $D^N = V_\perp^N$ . Tu tarea es dar una expresión  $e$  concreta y mostrar que la semántica denotacional normal no respeta la contracción  $\eta$ . Para ello calculá la semántica de  $\lambda x. e x$  y la de  $e$  para ver que son distintas.

$$e = \Delta\Delta$$

$$\llbracket e \rrbracket \eta = \perp_D$$

$$\llbracket \lambda x. e x \rrbracket \eta$$

$$= \perp (\psi (\lambda d. \llbracket e \rrbracket [\eta | x : d]))$$

$$= \perp (\psi (\lambda d. \varphi_{\perp} (\llbracket e \rrbracket [\eta | x : d]) (\llbracket x \rrbracket [\eta | x : d])))$$

$$= \perp (\psi (\lambda d. \varphi_{\perp} \perp_D \nu))$$

$$= \perp (\psi (\lambda d. \perp_{D \Rightarrow D} d))$$

$$= \perp (\psi (\lambda d. \perp_D))$$

$$= \perp (\psi \perp_{D \Rightarrow D})$$

4. Ahora nos pasamos al lenguaje aplicativo eager.

(a) Proponé una regla de evaluación para expresiones con pattern-matching muy sencillo:

$$\text{let } \langle \langle var \rangle, \langle var \rangle \rangle \equiv \langle exp \rangle \text{ in } \langle exp \rangle$$

(b) Evaluá la siguiente expresión usando esa regla.

$$\text{let } \langle x, y \rangle \equiv \underbrace{(\lambda n. \langle n, n * (-1) \rangle)(-4)}_{e_1} \text{ in if } x < y \text{ then } -1 \text{ else if } y = x \text{ then } 0 \text{ else } 1$$

$e_2$

a)

$$\hat{\epsilon} \Rightarrow \langle z_0, z_1 \rangle \quad \epsilon / [x_0 : z_0, x_1 : z_1] \Rightarrow z$$

$$\text{let } \langle x_0, x_1 \rangle \equiv \hat{\epsilon} \text{ in } \epsilon \Rightarrow z$$

b)

e<sub>2</sub>

e<sub>1</sub>

$$\lambda n. \langle n, n \cdot (-1) \rangle \Rightarrow \lambda n. \langle n, n \cdot (-1) \rangle$$

$$-4 \Rightarrow -4$$

$$\langle -4, (-4) \cdot (-1) \rangle$$

$$-4 \Rightarrow -4$$

$$(-4) \cdot (-1)$$

$$-4 \Rightarrow -4$$

$$-1 \Rightarrow -1$$

$$\Rightarrow 4$$

$$\Rightarrow \langle -4, 4 \rangle$$

$$\Rightarrow \langle -4, 4 \rangle$$

if  $-4 < 4$  then  $-1$  else if  $-4 = 4$  then  $0$  else  $1$

$$-4 < 4$$

$$-4 \Rightarrow -4$$

$$4 \Rightarrow 4$$

$$\Rightarrow \text{true}$$

$$-1 \Rightarrow -1$$

$$\Rightarrow -1$$

$\Rightarrow -1$

5. Considerá la siguiente serie de naturales:

$$S = 1, 2, 3, 1, 2, 3, \dots$$

Dar una expresión  $e$  en el lenguaje aplicativo eager (sin división ni módulo) tal que la denotación de  $e e'$  sea  $\iota_{\text{int}}(S(n))$  si la semántica de  $e'$  es  $\iota_{\text{int}}(n)$  con  $n > 0$ .

$$e = \text{letrec } f \equiv \lambda x. \underbrace{\text{if } x \leq 3 \text{ then } x \text{ else } f(x-3)}_{\hat{e}} \text{ in } f$$

Demarcación:

$$\text{Supongamos } \llbracket e' \rrbracket \eta \simeq c_{\text{int}}(n)$$

$$\llbracket e \rrbracket \eta$$

$$\begin{aligned} & \text{sea } F \vdash y = \llbracket \hat{e} \rrbracket [\eta \mid f : \text{fun } h, x : y] \\ & = \llbracket f \rrbracket [\eta \mid f : \text{fun } (Y F)] \\ & = \text{fun } (Y F) \end{aligned}$$

Propongo:

$$F = \left( \lambda x \in V_{int} . \begin{cases} x \leq 3 & \rightarrow \underline{\text{int}} x \\ \text{sino} & \rightarrow \underline{\text{int}}((x-1) \% 3 + 7) \end{cases} \right)_{int*}$$

Demos tráigh:

$$Fg$$

$$= \lambda y. [\hat{e}] [\eta] \mid f : \text{fun } g, x : y]$$

$$\text{[Llamemos } \eta' = [\eta] \mid \text{fun } g, x : y]$$

$$= \lambda y. \left( \lambda b. \begin{cases} b & \rightarrow [x]\eta' \\ \text{sino} & \rightarrow [f(x-3)]\eta' \end{cases} \right)_{b \in \mathbb{N}*} [\underline{x \leq 3}]\eta'$$

$$= \left( \lambda y. \begin{cases} y \leq 3 & \rightarrow \underline{\text{int}} y \\ \text{sino} & \rightarrow g(\underline{\text{int}}(y-3)) \end{cases} \right)_{int*}$$

$$= \left( \lambda y. \begin{cases} y \leq 3 & \rightarrow \underline{\text{int}} y \\ y-3 \leq 3 & \rightarrow \underline{\text{int}}(y-3) \\ \text{sino} & \rightarrow \underline{\text{int}}((y-3+1)\% 3 + 7) \end{cases} \right)_{int*}$$

$$= g$$

$$= g$$

$$[\underline{ee'}]\eta$$

$$\begin{aligned}
 & [[e^1]]\eta \\
 &= (\lambda f. f_*([[e^1]]\eta))_{\text{fun}_*} ([[e]]\eta) \\
 &= g(c_{\text{int}}(\eta)) \\
 &= \begin{cases} h \leq 3 \rightarrow c_{\text{int}} h \\ \text{Si } h_0 \rightarrow c_{\text{int}}((h-1)\% 3 + 1) \end{cases} \\
 &= \begin{cases} h \leq 3 \rightarrow c_{\text{int}} h \\ \text{Si } h_0 \rightarrow c_{\text{int}}(S(h)) \end{cases}
 \end{aligned}$$

6. Calcular la semántica denotacional eager de la expresión  $e$ .

Hecho en la sección 5

1. Considerá la siguiente ecuación recursiva.

$$g(x) = \begin{cases} 1 & \text{si } x = 0 \\ g(|x| - 2) & \text{si } x \neq 0 \end{cases}$$

Sea  $F: (\mathbb{Z} \rightarrow \mathbb{Z}_\perp) \rightarrow \mathbb{Z} \rightarrow \mathbb{Z}_\perp$  definido por:

$$F f x = \begin{cases} x + 1 & \text{si } x \in \{0, 1\} \\ f(|x| - 2) & \text{si } x \notin \{0, 1\} \end{cases}$$

- a) Escribí de la manera más clara posible  $\bigsqcup_k (F^k \perp)$ .
- b) ¿Es ese supremo una solución para  $g$ ? Justificá tu respuesta.

$$\left( \bigsqcup_{k \in \mathbb{N}} (F^k \perp) \right) x = \begin{cases} x \text{ es par} \Rightarrow 1 \\ \text{si no} \Rightarrow 2 \end{cases}$$

3. Considerá el cálculo lambda puro. Proponé una expresión  $e$  que tenga distintas formas canónicas en los órdenes de evaluación. Justificá tu respuesta haciendo las evaluaciones.

$$e = (\lambda x. y. x y) ((\lambda x. x)(\lambda x. x))$$

En normal:

$e$

$$\lambda x. y. x y \Rightarrow_N \lambda x. y. x y$$

$$\lambda y. ((\lambda x. x)(\lambda x. x)) \Rightarrow_N \lambda y. ((\lambda x. x)(\lambda x. x))$$

$$\Rightarrow_N \lambda y. ((\lambda x. x)(\lambda x. x))$$

En eager:

$e$

$$\lambda x. y. x y \Rightarrow_E \lambda x. y. x y$$

$$\lambda x.y \cdot xy \Rightarrow_E \lambda x.y \cdot xy$$

$$(\lambda x.x)(\lambda x.x)$$

$$\lambda x.x \Rightarrow_E \lambda x.x$$

$$\lambda x.x \Rightarrow_E \lambda x.x$$

$$\lambda x.x \Rightarrow_E \lambda x.x$$

$$\Rightarrow_E \lambda x.x$$

$$\lambda y.(\lambda x.x)y \Rightarrow_E \lambda y.(\lambda x.x)y$$

$$\Rightarrow_E \lambda y.(\lambda x.x)y$$

4. Considerá el lenguaje eager con recursión y la expresión

$$e = \lambda y.\text{letrec } f \equiv \lambda x.\text{if } x < y \text{ then } x \text{ else } f(x - y) \text{ in } f$$

a) Evalúa  $e 5 9$ .

b) ¿Cuál es la semántica denotacional de  $e (-3) 3?$

$$e \ 5 \ 9$$

$$e \ 5$$

$$e \Rightarrow_E e$$

$$5 \Rightarrow_E 5$$

$$\text{letrec } f \equiv \lambda x. \underbrace{\text{if } x < 5 \text{ then } x}_{e^1} \text{ else } f(x - 5) \text{ in } f$$

$$\underbrace{\lambda x. \text{letrec } f \equiv \lambda x. e^1 \text{ in } e^1}_{e^1} \Rightarrow_E \hat{e}$$

$$\Rightarrow_E \hat{e}$$

$$9 \Rightarrow_E 9$$

$$\text{letrec } f \equiv \lambda x. e^1 \text{ in if } 9 < 5 \text{ then } 9 \text{ else } f(9 - 5)$$

$$\text{if } 9 < 5 \text{ then } 9 \text{ else } \underbrace{(\lambda x. \text{letrec } f \equiv \lambda x. e^1 \text{ in } e^1)(9 - 5)}_{\hat{e}}$$

if  $9 < 5$  then  $\emptyset$  else  $\underbrace{(\lambda x. \text{letrec } f = \lambda x. e' \text{ in } e') (9 - 5)}_{\overline{e}}$

$9 < 5$

$9 \Rightarrow_E 9$

$5 \Rightarrow_E 5$

$\Rightarrow_E \text{False}$

$\overline{e}$

letrec  $f = \lambda x. e'$  in if  $9 - 5 < 5$  then  $9 - 5$  else  $f((9 - 5) - 5)$

if  $9 - 5 < 5$  then  $9 - 5$  else ...

$9 - 5 < 5$

$9 - 5$

$9 \Rightarrow_E 9$

$5 \Rightarrow_E 5$

$\Rightarrow_E \emptyset$

$5 \Rightarrow_E 5$

$\Rightarrow_E \text{True}$

$9 - 5$

$9 \Rightarrow_E 9$

$5 \Rightarrow_E 5$

$\Rightarrow_E \emptyset$

$\Rightarrow_E \emptyset$

$\Rightarrow_E \emptyset$

$\Rightarrow_E \emptyset$

$\Rightarrow_E \emptyset$

$\Rightarrow_E \emptyset$

4  
木