Semántica Denotacional

Idea: "El significado de un programa es la función denotada por el programa"

Componentes del metalenguaje para la definición semántica denotacional de un L.P.:

- *Dominios sintácticos* ➔ sintaxis abstracta del L.S.
- *Dominios semánticos* ➔ significado de constructores
- *Funciones semánticas* ➔ relación entre los dominios sintácticos y semánticos

  F : Sin → Se

Las funciones semánticas se definen mediante *ecuaciones semánticas*

Dominios

- Un dominio es un conjunto (de valores) que incluye además dos valores especiales: ⊥ (indefinido) y ⊤ (error)
- Representación por enumeración:

  T = {true, false}^{o} = {true, false, ⊥, ⊤}

- *Operaciones sobre dominios*

  Dominio producto: P = D1× D2 × ... × Dn
  p = <d1,d2,...,dn>
  p ↓ i = di (i-ésima componente)

  Dominio suma: S = D1 + D2 +... + Dn
  di∈ Di (di **in** S)∈ S (inyección)
  s∈ S (s | Di)∈ Di (proyección)

Dominio función: F = D1→ D2
λp.e (lambda-expresión)

Prioridad operaciones: × , +, →
D1×D2 → D3 + D4× D5 ≡ (D1×D2) → (D3 + (D4× D5))
Asociatividad a derechas de →
D1 → D2 → D3 ≡ D1 → (D2 → D3)

Dominio lista: D* = {nil} + D + D^2 + D^3 +...

hd: D* → D	(cabeza)	hd(nil) = ⊤
tl: D* → D*	(resto)	tl(nil) = ⊤
append: D* × D → D*		append(l,d) = l^<d>
prefix: D × D* → D*		prefix(d,l) = <d>^l

SINTAXIS ABSTRACTA del Lenguaje Sujeto

- Se representa por medio de *dominios sintácticos*

  Ejs: Ψ : Prog Σ : Stmt K : Comp

  metavariable (letra griega mayúscula) — dom.sintáctico (3 o más letras)

- Las definiciones de los dominios sintácticos vienen dadas por *reglas de producción abstractas*

  Ej: Sentencia "while", definiciones equivalentes:

  Σ : Comp × Stmt
  Σ = K × Σ
  Σ ::= `while K do Σ end`

SIGNIFICADO del Lenguaje Sujeto

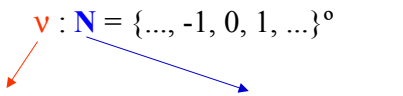
- Se representa el significado de los constructores mediante *dominios semánticos*

  Ejs:

  N = {..., -1, 0, 1, ...}° dominio de los números enteros

  S = Var → **N** dominio de los estados (de memoria)

  Cf = S × Fi × Fi dominio de las configuraciones



σ : S

γ : Cf

(letra griega minúscula) (1 ó 2 letras)

FUNCIONES SEMANTICAS

- Son elementos de algún dominio función de la forma **Sin → Se** (y se representan con una letra mayúscula)

  Ej: *S* : Stmt → (Cf → Cf)

  indica que "el significado de una sentencia es una función de cambio de configuración"

- La definición de una función semántica se hace por medio de *ecuaciones semánticas*

  Ej : *S* |[Ξ := E]| = "expresión de tipo Cf → Cf "

  func.sem. argumento (en sint.abstracta) ≡ < Ξ ,E>

  o también *S* |[Ξ := E]| (γ) = "expresión de tipo Cf "

- Otras notaciones:

  f [a / b] indica la función f1 definida:

  f1 (b) = a

  f1 (x) = f (x), para x ≠ b

  Si f : D → D entonces f^n indica f • ..... • f (composición n veces)

  En las ecuaciones semánticas, se usan expresiones condicionales de la forma:

  (c → e1, e2)

  para indicar:

  "si c es cierto entonces e1 sino e2"

Semántica Denotacional de Pam (tabla 4.3)

- ***Dominios sintácticos***

Ψ : Prog	programas
Σ : Stmt	sentencias
Ξ : Var	variables
Λ : Vars	listas de 1 ó más variables
E : Exp	expresiones
K : Comp	comparaciones
P : Rel	relaciones
Θ : Opr	operadores
N : Num	constantes enteras
Δ : Dig	dígitos decimales

- ***Reglas de producción abstractas*** *(ver tabla 4.3)*

  – Se ha suprimido la "serie de sentencias", ahora se tiene

  $$\Psi ::= \Sigma$$

  permitiendo

  $$\Sigma ::= \Sigma_1 ; \Sigma_2 \mid .....$$

  – Similarmente, para las listas de variables:

  $$\Lambda ::= \Xi \mid \Lambda_1 , \Lambda_2$$

  – No hay definición para Ξ porque la estructura interna de las variables no es relevante semánticamente.

- ***Dominios semánticos***

  – Hay dos dominios primitivos para

  los números enteros $\nu : \mathbf{N} = \{..., -1, 0, 1, ...\}^\circ$

  los valores de verdad $\tau : \mathbf{T} = \{true, false\}^\circ$

  – y tres dominios definidos para

  los estados (de memoria) $\sigma : S = Var \rightarrow \mathbf{N}$

  los ficheros $\phi : Fi = \mathbf{N}^*$

  las configuraciones $\gamma : Cf = S \times Fi \times Fi$

  Dada una configuración $\gamma = <\sigma, \phi_1, \phi_2>$ *recordar que*

  $$\gamma\downarrow 1 = \sigma \qquad \gamma\downarrow 2 = \phi_1 \qquad \gamma\downarrow 3 = \phi_2$$

- ***Funciones semánticas y ecuaciones semánticas***

  (1) $M : Prog \rightarrow (Fi \rightarrow Fi)$

  "El significado de un programa $M\,[\![\Psi]\!]$ es alguna función que aplica ficheros (de entrada) en ficheros (de salida)"

- La definición de $M\,[\![\Psi]\!]$ viene dada por la *ecuación semántica*

  $$M\,[\![\Psi]\!]\,(\phi_i) = (S\,[\![\Psi]\!]\,(\gamma_i))\downarrow 3$$

  donde

  $\gamma_i = <\lambda\Xi.\perp, \phi_i, nil>$ es la configuración inicial respecto al fichero de entrada ϕ_i, con memoria (inicial) $\lambda\Xi.\perp$ y fichero de salida nil

- *Ejemplo:* El significado del programa

  read x,y ; write y

  es la función $M\,[\![\text{read x,y ; write y}]\!] : Fi \rightarrow Fi$

  definida $M\,[\![\text{read x,y ; write y}]\!]\,\phi_i =$
  $(S\,[\![\text{read x,y ; write y}]\!]\,(<\lambda\Xi.\perp, \phi_i, nil>))\downarrow 3$

  Si $\phi_i = <-2,7,1>$ → $M\,[\![\text{read x,y ; write y}]\!]\,\phi_i = <7>$
  y en general :

  $M\,[\![\text{read x,y ; write y}]\!] : Fi \rightarrow Fi$
  $\phi_i \rightarrow <hd(tl(\phi_i))>$

(2) $S : \text{Stmt} \to (\text{Cf} \to \text{Cf})$

"El significado de una sentencia $S\,[\![\Sigma]\!]$ es una función de cambio de configuración"

La definición de $S\,[\![\Sigma]\!]$ viene dada por un conjunto de *ecuaciones semánticas* (una por cada clase distinta de Σ):

- *Secuencia de sentencias:*

$S\,[\![\ \Sigma_1\ ;\ \Sigma_2\ ]\!] = S\,[\![\Sigma_2\ ]\!] \bullet S\,[\![\Sigma_1\ ]\!]$

Ejemplo:

$S\,[\![\text{read x,y ; write y}]\!] = S\,[\![\text{write y}]\!] \bullet S\,[\![\text{read x,y}]\!]$

- *Sentencias de lectura y escritura de listas de variables:*

$S\,[\![\ \text{read}\ \Lambda_1\ ,\ \Lambda_2\ ]\!] = S\,[\![\text{read}\ \Lambda_1\ ;\ \text{read}\ \Lambda_2\ ]\!]$

$S\,[\![\ \text{write}\ \Lambda_1\ ,\ \Lambda_2\ ]\!] = S\,[\![\text{write}\ \Lambda_1\ ;\ \text{write}\ \Lambda_2\ ]\!]$

Ejemplo:

$S\,[\![\text{read x,y}]\!] = S\,[\![\text{read x ; read y}]\!] = S\,[\![\text{read y}]\!] \bullet S\,[\![\text{read x}]\!]$

- *Sentencia de lectura de una variable:*

$$S\,[\![\ \text{read}\ \Xi\ ]\!]\ \langle\sigma, \phi_1, \phi_2\rangle = \begin{cases} \text{error} & \textit{si}\ \ \phi_1 = \text{nil} \\ \langle\sigma[\text{hd}(\phi_1)\ /\ \Xi],\ \text{tl}(\phi_1)\ ,\ \phi_2\rangle & \textit{e.o.c} \end{cases}$$

Ejemplo:

$S\,[\![\text{read x}]\!]\ (\langle\sigma_0, \langle -2,7,1\rangle, \text{nil}\rangle) = \langle\sigma_0[-2\ /\ x],\ \langle 7,1\rangle,\ \text{nil}\rangle$

- *Sentencia de escritura de una variable:*

$$S\,[\![\ \text{write}\ \Xi\ ]\!]\ \langle\sigma, \phi_1, \phi_2\rangle = \begin{cases} \text{error} & \textit{si}\ \ \sigma\,[\![\ \Xi\ ]\!] = \bot \\ \langle\sigma,\ \phi_1,\ \text{append}(\phi_2\ ,\ \sigma\,[\![\ \Xi\ ]\!])\rangle & \textit{e.o.c} \end{cases}$$

Ejemplo:

Dados $\sigma_0 = \lambda\Xi.\bot$

$\sigma_1 = \sigma_0\,[-2\ /\ x]$

$\sigma_2 = \sigma_1[7\ /\ y] = (\ \sigma_0\,[-2\ /\ x]\ )\ [7\ /\ y]$

$S\,[\![\text{write y}]\!]\ (\langle\sigma_2, \langle 1\rangle, \text{nil}\rangle) = \langle\sigma_2\ ,\ \langle 1\rangle\ ,\ \langle 7\rangle\rangle$

- *Sentencia de asignación:*

$S\,[\![\ \Xi := \text{E}\ ]\!]\ \ \langle\sigma, \phi_1, \phi_2\rangle\ =\ \langle\sigma[\text{val}\ /\ \Xi\ ],\ \phi_1,\ \phi_2\rangle$ con

$\text{val} = E\,[\![\text{E}]\!]\ (\sigma)$

- *Bucle definido:*

$$S\,[\![\text{to E do}\ \Sigma\ \text{end}]\!]\ \langle\sigma, \phi_1, \phi_2\rangle\ = \begin{cases} \langle\sigma, \phi_1, \phi_2\rangle & \textit{si}\ \ \nu < 1 \\ (\ S\,[\![\Sigma]\!]\ )^{\nu}\ \langle\sigma, \phi_1, \phi_2\rangle & \textit{e.o.c} \end{cases}$$

con $\nu = E\,[\![\text{E}]\!]\ (\sigma)$

Ambas usan la función semántica $E : \text{Exp} \to (\text{S} \to \mathbf{N})$

$E\,[\![\text{E}]\!]\ (\sigma)$ denota el valor (entero) de E en el estado σ

En las sentencias condicionales y el bucle indefinido se usa la función semántica $C : Comp \rightarrow (S \rightarrow \mathbf{T})$

$C\,[\![K]\!]\,(\sigma)$ denota el valor (de verdad) de K en el estado σ

- *Sentencias condicionales:*

$S\,[\![\text{if K then } \Sigma \text{ fi}]\!]\ \gamma = (\ C\,[\![K]\!]\,(\gamma\downarrow 1) \rightarrow S\,[\![\Sigma]\!]\,\gamma,\ \gamma\ )$

$S\,[\![\text{if K then } \Sigma_1 \text{ else } \Sigma_2 \text{ fi}]\!]\ \gamma =$
$(\ C\,[\![K]\!]\,(\gamma\downarrow 1) \rightarrow S\,[\![\Sigma_1]\!]\,\gamma,\ S\,[\![\Sigma_2]\!]\,\gamma\ )$

- *Bucle indefinido:*

$S\,[\![\text{while K do } \Sigma \text{ end}]\!]\ \gamma =$
$(\ C\,[\![K]\!]\,(\gamma\downarrow 1) \rightarrow (S\,[\![\text{while K do } \Sigma \text{ end}]\!] \bullet S\,[\![\Sigma]\!])\,\gamma,\ \gamma\ )$

(3) $C : Comp \rightarrow (S \rightarrow \mathbf{T})$

"El significado de una comparación $C\,[\![K]\!]$ es una función que aplica estados en valores de verdad"

Como una comparación es de la forma $K ::= E_1\ P\ E_2$,
$C\,[\![K]\!]$ se define en términos de E dependiendo de P

$C\,[\![E_1 = E_2]\!]\,\sigma = (\ E\,[\![E_1]\!]\,\sigma = E\,[\![E_2]\!]\,\sigma \rightarrow \text{true, false}\ )$

$C\,[\![E_1 >= E_2]\!]\,\sigma = (\ E\,[\![E_1]\!]\,\sigma \geq E\,[\![E_2]\!]\,\sigma \rightarrow \text{true, false}\ )$
$C\,[\![E_1 <> E_2]\!]\,\sigma = (\ E\,[\![E_1]\!]\,\sigma \neq E\,[\![E_2]\!]\,\sigma \rightarrow \text{true, false}\ )$

sintaxis (abstracta) operación en $\mathbf{N} = \{..., -1, 0, 1, ...\}^\circ$

(4) $E : Exp \rightarrow (S \rightarrow \mathbf{N})$

"El significado de una expresión $E\,[\![E]\!]$ es una función que aplica estados en valores enteros"

Como una expresión es de la forma $E ::= E_1\ \Theta\ E_2 \mid \Xi \mid N$
$E\,[\![E]\!]$ se define para cada caso de $E_1\ \Theta\ E_2$ dependiendo de Θ, para el caso Ξ y para los casos de constantes enteras N

Destacan dos posibles errores semánticos:

$E\,[\![E_1 / E_2]\!]\,\sigma = (\ E\,[\![E_2]\!]\,\sigma = 0 \rightarrow \text{error}, E\,[\![E_1]\!]\,\sigma \div E\,[\![E_2]\!]\,\sigma\ )$

$E\,[\![\Xi]\!]\,\sigma = (\ \sigma\,[\![\Xi]\!] = \perp \rightarrow \text{error}, \sigma\,[\![\Xi]\!]\ )$

Ejercicio

Dar el significado (denotacional) del siguiente programa Pam:

```
Ψ =  read k;
     while k>0 do k:= k-1 end;
     write k
```

Solución:

Debe darse qué función concreta denota este programa

$M\,[\![\Psi]\!] : Fi \rightarrow Fi$
$\Phi \rightarrow ??$

Semántica Denotacional de Eva (tabla 4.4)

- ***Dominios sintácticos***

Ψ : Prog	programas
B : Blo	bloques
Δ : Dec	declaraciones
I : Ide	identificadores
T : Type	tipos de datos
Σ : Stmt	sentencias
E : Exp	expresiones
Ξ : Str	strings literales
Λ : Let	letras

- ***Reglas de producción abstractas*** *(ver tabla 4.4)*

  – Se han suprimido como dominios sintácticos explícitos la "secuencia de declaraciones", permitiendo

  $$\Delta ::= \Delta_1 \ \Delta_2 \mid .....$$

  y la "secuencia de sentencias", permitiendo:

  $$\Sigma ::= \Sigma_1 ; \Sigma_2 \mid .....$$

  – Las listas de parámetros (resp. de argumentos) se especifican con una herramienta notacional "elíptica" :

  $$\text{proc I } (T_1\ I_1, ..., T_n\ I_n) = \Sigma \qquad (n \geq 1)$$
  $$\text{call I } (E_1, ..., E_n) \qquad (n \geq 1)$$

- ***Dominios semánticos***

➢ Debido a la existencia de bloques (definiciones locales) debemos distinguir entre *entorno* y *(estado de) memoria:*

Ide(ntificador) $\xrightarrow{\rho}$ L(ugar) $\xrightarrow{\sigma}$ V(alor)

I $\qquad$ $\rho\,|[I]|$ $\qquad$ $\sigma(\,\rho\,|[I]|\,)$

$\rho\,|[I]|$ = lugar de memoria asociado a I en el entorno ρ

$\sigma(\alpha)$ = valor contenido en el lugar α del estado de memoria σ

Se necesita dar dos pasos (mediante ρ y σ) para alcanzar el valor asociado a un identificador I de tipo char ó string.

Dominios semánticos necesarios para Eva (1)

Los estados (de memoria)	$\sigma : S = L \rightarrow V$
los lugares (de memoria)	$\alpha : L = \{1,2,3,....\}^o$
los valores (de memoria)	$\beta : V = H + Z + \{unallocated\}^o$
los valores carácter	$\eta : H = \{a,b, ..., z, space\}^o$
los valores string	$\zeta : Z = H^*$

unallocated es un valor especial para indicar un lugar sin asignar:

" Si para todo entorno ρ y todo identificador I se tiene que $\rho\,|[I]| \neq \alpha$ entonces $\sigma(\alpha) =$ *unallocated "*

- Debido a la existencia de procedimientos, debemos asociar a un identificador I de tipo proc un "valor de procedimiento" π: P

  Estos valores no serán almacenables (en memoria) sino entidades asociadas a los identificadores por medio del entorno (como ocurre con los lugares).

  Ide(ntificador) $\xrightarrow{\rho}$ P(rocedimiento)
  I $\qquad$ ρ [[I]]

  Esto implica definir el dominio U de los entornos así:

  $U = Ide \rightarrow D$ (en lugar de $U = Ide \rightarrow L$)

  siendo $D = L + P$ el dominio de los "valores denotables"

Dominios semánticos necesarios para Eva (2)

Los entornos $\rho : U = Ide \rightarrow D$

los valores denotables $\delta : D = L + P$

las configuraciones $\gamma : Cf = S \times Z \times Z$

y los valores de procedimiento

$\pi : P = L^* \rightarrow (Cf \rightarrow Cf)$

donde

lista de lugares que se asociarán a los parámetros formales (en la llamada al procedimiento)

cambio de configuración que producirá el cuerpo del procedimiento

• *Funciones semánticas y ecuaciones semánticas*

(1) $\quad M : Prog \rightarrow (Z \rightarrow Z)$

"El significado de un programa M [[Ψ]] es alguna función que aplica ficheros (de entrada) en ficheros (de salida)"

- La definición de M [[Ψ]] viene dada por la *ecuación semántica*

  M [[Ψ]] (ζ_i) = (S [[Ψ]] (ρ_{ini}) (γ_i)) $\downarrow$ 3

  donde

  $\gamma_i = < \lambda\alpha.\ unallocated, \zeta_i, nil >$ es la configuración inicial respecto al fichero de entrada ζ_i

  y siendo el entorno inicial $\rho_{ini} = \lambda I.\ \perp$

- El significado de una sentencia S [[Σ]] y el de una expresión E [[E]] son como en Pam, salvo que en Eva se definen relativas a un entorno U

(2)	$S : Stmt \rightarrow (U \rightarrow (Cf \rightarrow Cf))$
(3)	$E : Exp \rightarrow (U \rightarrow (S \rightarrow V))$

S [[Σ]] $\rho : Cf \rightarrow Cf$ "significado de Σ en el entorno ρ"

S [[Σ]] $\rho\ \gamma$ = " configuración obtenida al ejecutar Σ en el entorno ρ y sobre la configuración γ "

E [[E]] $\rho\ \sigma$ = " valor obtenido al evaluar E en el entorno ρ y sobre la memoria σ"

- La definición de $S\,[\![\Sigma]\!]$ vendrá dada por un conjunto de *ecuaciones semánticas* (una por cada clase distinta de Σ)

- El significado de un bloque (begin Δ Σ end) es el significado del cuerpo del bloque (Σ), pero relativo a un entorno y a una memoria modificados por las declaraciones (Δ) del bloque.

$$S\,[\![\text{begin } \Delta\ \Sigma \text{ end}]\!]\ \rho\ \langle\sigma, \zeta_1, \zeta_2\rangle = S\,[\![\ \Sigma\ ]\!]\ \rho'\ \langle\sigma', \zeta_1, \zeta_2\rangle$$

donde ρ' y σ' se obtienen al modificar ρ y σ según Δ

$$\langle\rho', \sigma'\rangle = D\,[\![\ \Delta\ ]\!]\ \rho\ \langle\rho, \sigma\rangle$$

¿ ?

- ***Necesidad de la función semántica D para el significado de*** Δ

Supongamos la función semántica D de la forma:

(4*) $\quad D : \text{Dec} \rightarrow (U \times S \rightarrow U \times S)$

"El significado de una declaración $D\,[\![\Delta\ ]\!]$ sería una función de modificación de pares (entorno, memoria) "

Las ecuaciones semánticas correspondientes serían:

- $D\,[\![\text{char I}]\!]\ \langle\rho,\sigma\rangle = \langle\ \rho\ [\alpha \text{ in D / I}\ ]\ ,\ \sigma\ [\perp / \alpha]\ \rangle$

  donde α *es un lugar "nuevo":* $\sigma\ (\alpha) =$ *unallocated*

- $D\,[\![\text{string I}]\!]\ \langle\rho,\sigma\rangle = \langle\ \rho\ [\alpha \text{ in D / I}\ ]\ ,\ \sigma\ [\text{nil} / \alpha]\ \rangle$

  donde α *es un lugar "nuevo":* $\sigma\ (\alpha) =$ *unallocated*

- $D\,[\![\ \Delta_1\ \Delta_2\ ]\!] = D\,[\![\ \Delta_2\ ]\!] \bullet D\,[\![\ \Delta_1\ ]\!]$

  es decir: $D\,[\![\ \Delta_1\ \Delta_2\ ]\!]\ \langle\rho, \sigma\rangle = D\,[\![\ \Delta_2\ ]\!]\ (\ D\,[\![\ \Delta_1\ ]\!]\ \langle\rho, \sigma\rangle\ )$

- $D\,[\![\text{proc I} = \Sigma]\!]\ \langle\rho, \sigma\rangle = \langle\ \rho\ [\pi \text{ in D / I}\ ]\ ,\ \sigma\ \rangle$

  donde al procedimiento (sin parámetros) de nombre I se le asigna en el nuevo entorno el "valor"

$$\pi : \ L^* \rightarrow (\text{Cf} \rightarrow \text{Cf})$$

tal que π (nil) denota el cambio de configuración de Σ (cuerpo de I) relativo a ¿qué entorno?

a) $\pi\ (\text{nil}) = S\,[\![\Sigma]\!]\ \rho$ ← *entorno "antes de I"*

b) $\pi\ (\text{nil}) = S\,[\![\Sigma]\!]\ \rho\ [\pi \text{ in D / I}]$ ← *entorno "con I"*

Problema:

Ni en *a)* $\quad \pi\ (\text{nil}) = S\,[\![\Sigma]\!]\ \rho$

ni en *b)* $\quad \pi\ (\text{nil}) = S\,[\![\Sigma]\!]\ \rho[\pi \text{ in D / I}]$

se tiene en cuenta que en el cuerpo Σ se puede hacer referencia a variables declaradas tras proc I = Σ (en la misma Δ)

- ***Necesidad de cambiar la función semántica D para el significado de*** Δ

(4) $\quad D : \text{Dec} \rightarrow U \rightarrow (U \times S \rightarrow U \times S)$

"El significado de una declaración $D\,[\![\Delta]\!]$ es una función de modificación de pares (entorno, memoria) relativa al entorno correspondiente a la secuencia entera de declaraciones que contiene a Δ"

- Por eso, en el significado de los bloques se necesita poner:

S [[begin Δ Σ end]] ρ <σ, ζ_1, ζ_2> = S [[Σ]] ρ' <σ', ζ_1, ζ_2>

donde $< \rho', \sigma' > = D$ [[Δ]] ρ' <ρ, σ>

y las ecuaciones semánticas para D son ahora las siguientes:

- D [[char I]] ρ' <ρ,σ> = < ρ [α in D / I] , σ [$\perp$ / α] >
- D [[string I]] ρ' <ρ,σ> = < ρ [α in D / I] , σ [nil / α] >

  en ambos casos con $\sigma(\alpha) = unallocated$

- D [[Δ_1 Δ_2]] ρ' = D [[Δ_2]] ρ' • D [[Δ_1]] ρ'
- D [[proc I = Σ]] ρ' <ρ,σ> = < ρ [π in D / I] , σ >

  siendo π (nil) = S [[Σ]] ρ'

Por último, para los procedimientos con parámetros:

- D [[proc I (T_1 I_1,...,T_n I_n) = Σ]] ρ' <ρ,σ> = < ρ [π in D / I] , σ >

siendo

π (<α_1,..., α_n>) = S [[Σ]] (ρ' [α_1 in D / I_1 ,..., α_n in D / I_n])

Esto es, el valor π toma una lista de lugares (que se conocerán en el momento de cada llamada) y produce la función

π (<α_1,...,α_n>) : Cf $\rightarrow$ Cf

denotada por el cuerpo del procedimiento relativo al entorno:

ρ' [α_1 in D / I_1 ,..., α_n in D / I_n]

que es el entorno ρ' actualizado con la asignación de dichos lugares a los parámetros formales.

Ejemplo Consideramos el programa Ψ siguiente:

```
begin
    proc p (char x) = Σ3          Δ1
    string a
    begin
        char a                    Δ2
        char b
        Σ5                        Σ1
        call p (b)    Σ2
        Σ6
    end
    Σ4
end
```


Ejemplo (cont)

Sean $\rho_0 = \lambda I.\perp$ y $\sigma_0 = \lambda\alpha.unallocated$

S [[Ψ]] ρ_0 <σ_0, ζ_1, ζ_2>
= S [[begin Δ_1 Σ_1 end]] ρ_0 <σ_0, ζ_1, ζ_2 >
= S [[Σ_1]] ρ_1 <σ_1, ζ_1, ζ_2 >

donde

<ρ_1 , σ_1> =
D [[Δ_1]] ρ_1 <ρ_0 , σ_0> =
D [[string a]] ρ_1 (D [[proc p (char x) = Σ_3]] ρ_1 <ρ_0 , σ_0>) =
D [[string a]] ρ_1 <ρ_0 [π/p] , σ_0> =
siendo π (<α>) = S [[Σ_3]] (ρ_1[α /x])
= <ρ_0 [π/p, 1/a] , σ_0[nil /1] >
ρ_1 **σ_1**

Ejemplo (cont)

$S\,[\![\Psi]\!]\,\rho_0\,\langle\sigma_0, \zeta_1, \zeta_2\rangle = \ldots = S\,[\![\Sigma_1]\!]\,\rho_1\,\langle\sigma_1, \zeta_1, \zeta_2\rangle =$
$= S\,[\![\Sigma_1]\!]\,(\rho_0\,[\pi/p,\ 1/a]\,)\ \langle\sigma_0[nil/1], \zeta_1, \zeta_2\rangle =$

siendo

$\pi\,(\langle\alpha\rangle) = S\,[\![\Sigma_3]\!]\,(\rho_1\,[\alpha/x]\,)$
$= S\,[\![\Sigma_3]\!]\,(\rho_0\,[\pi/p,\ 1/a\ ,\ \alpha/x])$

$= S\,[\![\text{begin } \Delta_2\ \Sigma_2 \text{ end } \Sigma_4]\!]\,(\rho_0\,[\pi/p,\ 1/a])\ \langle\sigma_0[nil/1], \zeta_1, \zeta_2\rangle =$
$= S\,[\![\Sigma_4]\!]\,(\rho_0\,[\pi/p,\ 1/a])$
$(\ S\,[\![\text{begin } \Delta_2\ \Sigma_2 \text{ end}]\!]\,(\rho_0\,[\pi/p,\ 1/a])\ \langle\sigma_0[nil/1], \zeta_1, \zeta_2\rangle\ )$

- $S\,[\![\Sigma_1\ \Sigma_2]\!]\,\rho = S\,[\![\Sigma_2\,]\!]\,\rho\ \bullet\ S\,[\![\Sigma_1\,]\!]\,\rho$ *secuencia de sentencias*

Ejemplo (cont)

$S\,[\![\Psi]\!]\,\rho_0\,\langle\sigma_0, \zeta_1, \zeta_2\rangle = \ldots\ldots =$
$= S\,[\![\Sigma_4]\!]\,(\rho_0\,[\pi/p,\ 1/a])$
$(\ S\,[\![\text{begin } \Delta_2\ \Sigma_2 \text{ end}]\!]\,(\underbrace{\rho_0\,[\pi/p,\ 1/a]}_{\rho_1})\ \langle\underbrace{\sigma_0[nil/1]}_{\sigma_1}, \zeta_1, \zeta_2\rangle\ ) =$
$= S\,[\![\Sigma_4]\!]\,(\rho_0\,[\pi/p,\ 1/a])\ (\ S\,[\![\Sigma_2]\!]\,\rho_2 < \sigma_2, \zeta_1, \zeta_2\rangle\ )$

donde

$\langle\rho_2\ ,\ \sigma_2\rangle =$
$D\,[\![\Delta_2]\!]\,\rho_2\,\langle\rho_1\ ,\ \sigma_1\rangle = D\,[\![\text{char a}\ \ \text{char b}]\!]\,\rho_2\,\langle\rho_1\ ,\ \sigma_1\rangle =$
$D\,[\![\text{char b}]\!]\,\rho_2\,(\ D\,[\![\text{char a}]\!]\,\rho_2\,\langle\rho_1\ ,\ \sigma_1\rangle\ ) =$
$D\,[\![\text{char b}]\!]\,\rho_2\ \ \langle\rho_0\,[\pi/p\ ,\ 2/a],\ \sigma_0[nil/1\ ,\ \perp/2]\rangle =$
$= \langle\underbrace{\rho_0\,[\pi/p,\ 2/a,\ 3/b]}_{\rho_2}\ ,\ \underbrace{\sigma_0[nil/1,\ \perp/2,\ \perp/3]}_{\sigma_2}\rangle$

Ejemplo (cont)

Lo obtenido hasta ahora es:

$S\,[\![\Psi]\!]\,\rho_0\,\langle\sigma_0, \zeta_1, \zeta_2\rangle = \ldots\ldots =$
$= S\,[\![\Sigma_4]\!]\,\rho_1\,(\ S\,[\![\Sigma_2]\!]\,\rho_2\,\langle\sigma_2, \zeta_1, \zeta_2\rangle\ )$

donde

$\rho_1 = \rho_0\,[\pi/p,\ 1/a]$
$\rho_2 = \rho_0\,[\pi/p,\ 2/a,\ 3/b]$ y $\sigma_2 = \sigma_0[nil/1,\ \perp/2,\ \perp/3]$

- Nota: El significado de Σ_2 se define relativo al entorno ρ_2 debido a las declaraciones locales Δ_2 , mientras que el significado de Σ_4 se define relativo al entorno ρ_1 porque dichas declaraciones locales pierden su efecto al salir del bloque interno.

- La ecuación semántica correspondiente a la *llamada a un procedimiento con parámetros* es:

- $S\,[\![\text{call I } (E_1,\ldots,E_n)]\!]\,\rho\,\langle\sigma, \zeta_1, \zeta_2\rangle$
$= (\underbrace{\rho\,[\![I]\!]\,|\,P}_{\pi})\ (\langle\alpha_1,\ldots,\alpha_n\rangle)\ (\langle\sigma', \zeta_1, \zeta_2\rangle)$

donde

α_i son lugares nuevos: $\sigma\,(\alpha_i) = \textit{unallocated}$ $(i=1\ldots n)$
y distintos entre sí: $\alpha_i \neq \alpha_j$ $(i \neq j)$

y σ' se obtiene al actualizar σ asignando a cada α_i el valor del correspondiente argumento E_i

$\sigma' = \sigma[\ E\,[\![E_1]\!]\,\rho\,\sigma/\alpha_1\ ,\ \ldots.,\ E\,[\![E_n]\!]\,\rho\,\sigma/\alpha_n]$

Ejemplo (cont) Siguiendo con el ejemplo:

$S\,[\![\Psi]\!]\,\rho_0\,\langle\sigma_0, \zeta_1, \zeta_2\rangle =$

$= S\,[\![\Sigma_4]\!]\,\rho_1\,(\,S\,[\![\Sigma_5 \text{ call p (b) } \Sigma_6]\!]\,\rho_2\,\langle\sigma_2, \zeta_1, \zeta_2\rangle\,) =$

$= S\,[\![\Sigma_4]\!]\,\rho_1\,(\,S\,[\![\Sigma_6]\!]\,\rho_2\,(\,S\,[\![\text{call p (b)}]\!]\,\rho_2$

$(\,S\,[\![\Sigma_5]\!]\,\rho_2\,\langle\sigma_2, \zeta_1, \zeta_2\rangle\,)\,)\,)$

suponemos que da

$= S\,[\![\Sigma_4]\!]\,\rho_1\,(\,S\,[\![\Sigma_6]\!]\,\rho_2\,(\,S\,[\![\text{call p (b)}]\!]\,\rho_2\,\langle\sigma_3, \zeta_3, \zeta_4\rangle\,))$

$= S\,[\![\Sigma_4]\!]\,\rho_1\,(\,S\,[\![\Sigma_6]\!]\,\rho_2\,(\pi\,(\langle 4\rangle)\,\langle\sigma_3\,[E\,[\![b]\!]\,\rho_2\,\sigma_3 / 4], \zeta_3, \zeta_4\rangle))$

suponemos nuevo

y como $\pi\,(\langle\alpha\rangle) = S\,[\![\Sigma_3]\!]\,(\rho_0\,[\pi/p,\ 1/a\ ,\ \alpha/x])$

$= S\,[\![\Sigma_4]\!]\,\rho_1\,(\,S\,[\![\Sigma_6]\!]\,\rho_2$

$(\,S\,[\![\Sigma_3]\!]\,(\rho_0\,[\pi/p,\ 1/a\ \ 4/x])\,\langle\sigma_3\,[E\,[\![b]\!]\,\rho_2\,\sigma_3 / 4], \zeta_3, \zeta_4\rangle\,))$

- La ecuación semántica correspondiente a la *llamada a un procedimiento sin parámetros* es:

- $S\,[\![\text{call I}]\!]\,\rho\,\langle\sigma, \zeta_1, \zeta_2\rangle = (\rho\,[\![I]\!]\,|\,P)\,(\text{nil})\,(\langle\sigma, \zeta_1, \zeta_2\rangle)$

- Las demás ecuaciones semánticas para S son:

- $S\,[\![\text{input I}]\!]\,\rho\,\langle\sigma, \zeta_1, \zeta_2\rangle =$
  $(\,\zeta_1 = \text{nil} \rightarrow \text{error}, \langle\sigma\,[\text{hd}(\zeta_1) \text{ in V} / (\rho\,[\![I]\!]\,|\,L)], \text{tl}(\zeta_1), \zeta_2\rangle\,)$

- $S\,[\![\text{output E}]\!]\,\rho\,\langle\sigma, \zeta_1, \zeta_2\rangle = \langle\sigma, \zeta_1, \textit{append}(\zeta_2, E\,[\![E]\!]\rho\,\sigma\,|\,H)\rangle$

- $S\,[\![\text{eq } E_1, E_2 : \Sigma]\!]\,\rho\,\gamma = (E[\![E_1]\!]\rho\,\sigma = E\,[\![E_2]\!]\rho\,\sigma) \rightarrow S\,[\![\Sigma]\!]\rho\,\gamma, \gamma\,)$
  (neq) (≠) *donde* $\sigma = \gamma \downarrow 1$

- $S\,[\![\text{cons E,I}]\!]\,\rho\,\langle\sigma, \zeta_1, \zeta_2\rangle =$
  $\langle\sigma[\,\textit{prefix}(E\,[\![E]\!]\rho\,\sigma\,|\,H, \sigma(\rho\,[\![I]\!]\,|\,L)\,|Z) \text{ in V} / (\rho\,[\![I]\!]\,|\,L)],$
  $\zeta_1, \zeta_2\rangle$

- Finalmente, la función semántica de las expresiones es

(3) $E : \text{Exp} \rightarrow (\,\text{U} \rightarrow (\,\text{S} \rightarrow \text{V}\,))$

y sus ecuaciones son:

- $E\,[\![I]\!]\,\rho\,\sigma = (\,\sigma(\rho\,[\![I]\!]\,|\,L) = \perp \rightarrow \text{error},\ \sigma\,(\rho\,[\![I]\!]\,|\,L)\,)$
- $E\,[\![\text{space}]\!]\,\rho\,\sigma = \text{space in V}$ *(para el carácter blanco)*
- $E\,[\![\text{“a”}]\!]\,\rho\,\sigma = \text{a in V}$ *(para el resto de caracteres)*
- $E\,[\![\text{“”}]\!]\,\rho\,\sigma = \text{nil in V}$ *(para el string vacío)*
- $E\,[\![\text{“Ξa”}]\!]\,\rho\,\sigma = \textit{append}(\,E\,[\![\text{“Ξ”}]\!]\,\rho\,\sigma\,|\,Z\,,\,a\,) \text{ in V}$
- $E\,[\![\text{head E}]\!]\,\rho\,\sigma = (\,s \neq \text{nil} \rightarrow \text{hd(s)}, \text{error}\,) \text{ in V}$
- $E\,[\![\text{tail E}]\!]\,\rho\,\sigma = (\,s \neq \text{nil} \rightarrow \text{tl(s)}, \text{error}\,) \text{ in V}$

ambas con $s = E\,[\![E]\!]\,\rho\,\sigma\,|\,Z$