XI

SEMANTICA ALGEBRICA DELLO SMALL

Diamo ora la semantica, secondo lo schema già adot= tato per l'$\underline{\underline{A}}$-While e l'$\underline{\underline{A}}$-McCarthy, di un linguaggio più complesso che è sostanzialmente lo SMALL definito in [18].

Sia $\underline{\underline{A}}=(A, f^A, c^A)$ un prefissato tipo di dati

Sintassi astratta

La sintassi astratta di SMALL è definibile in BNF come segue:

I, L, P, F: id

C, C_1, C_2: cmd

E, E_1, E_2: exp

D, D_1, D_2: dec

Pr : prg

I::= $I_1 | I_2 | \ldots$

C::= (I:=E) | (output E) | P(E) | (if E then C_1 else C_2) | (C_1;C_2) | (While E do C) | (I:C) | (go to L) | (begin D;C end)

E::= I | read | True, False | f(E) | $\underline{\underline{c}}$ | F(E) | (if E then E_1 else E_2)

D::= (const I=E) | (var I=E) | (D_1;D_2) | (proc P(I)= E) | (fun F(I)= E)

Pr::= (Program C)

ID, CMD, EXP, DEC, PRG sono gli insiemi di termini relativi alle sorte id, cmd, exp, dec, prg di una segna=

tura Σ_o, facilmente definibile, che esprima la sintassi sopra descritta.

Definizione della base B

Consideriamo i seguenti domini semantici ove + indica somma disgiunta e (A → B)l'insieme delle funzioni da A in B,

A : l'insieme dei valori del tipo di dati A

Bv : {True, False}

Loc = insieme di locazioni, l ∈ Loc

Ev = A + Bv + Loc ,valori esprimibili, e ∈ Ev

Iv = A^* sequenze finite di valori, valori di input, i∈ Iv

Ov = A^* x {Stop} , valori di output , o ∈ Ov

$$\begin{cases} Env = \bigcup_{i\in N} Env_i \text{ , ambienti , } \delta \in Env \\ Env_o = (ID \longrightarrow Ev + CMD + \{unbound\}) \\ Env_{i+1} = (ID \longrightarrow Ev + CMD + Proc_i + Fun_i + \{unbound\}) \end{cases}$$

$$\begin{cases} Proc = \bigcup_{i\in N} Proc_i \\ Proc_i = CMD \times ID \times Env_i \end{cases}$$

$$\begin{cases} Fun = \bigcup_{i\in N} Fun_i \\ Fun_i = EXP \times ID \times Env_i \end{cases}$$

Store = (Loc → (Ev + {unused })) , memorie

Sta = Store x Iv x Ov , stati , ρ ∈ Sta

Seq = {Jump, Continue} , marche di sequenza

Oltre ai domini semantici sopra dati in $\underline{B}$ vi sono le componenti di tali domini semantici e gli insiemi ottenuti da queste per prodotto cartesiano x , somma disgiunta $+$, esponenziazione insiemistica $\rightarrow$.

Su tali domini assumiamo le seguenti operazioni semantiche:

1) **funzioni di accoppiamento** $(-,-)$ e **proiezioni** tali che se

$$y \in A \times B \times \ldots \times C \quad , \quad A \neq B \neq \ldots \neq C$$

allora

$$y\downarrow A \ , \ y\downarrow B \ , \ \ldots \ y\downarrow C$$

sono le proiezioni di y su A, B, ...C rispetivamente

2) **funzioni di estrazione** e di **immersione** per somme disgiunte, che non indicheremo esplicitamente cioè se $a \in A$ allora a continua a indicare l'elemento corrispondente di A in ogni somma disgiunta contenente A (sarà il contesto a disambiguare)

3) **operazioni di applicazione funzionale,** tali che se $f \in (A \rightarrow B)$ e $a \in A$ allora fa è il valore di f su a.

funzioni di aggiornamento tali che

$$f[a/x]y = \begin{cases} fy & \text{se } x \neq y \\ a & \text{se } x = y \end{cases}$$

$$f[g]\ y = \begin{cases} fy & \text{se } gy = \text{unbound} \\ gy & \text{altrimenti} \end{cases}$$

ove unbound è un particolare elemento del codominio di f e g (vedi sopra)

4) le usuali **operazioni su sequense : first, rest, giustaposizione** (indicata con $\smile$) e **sostituzione** in=

dicata ancora con [/] ove

$$(x_1 \ldots x_n)[y_i/x_i] = (x_1 \ldots . y_i \ldots . x_n)$$

5) **funzioni condizionali** e **test booleani** di uguaglianza e appartenenza a componenti di somme disgiunte

6) λ : stringa vuota

() : ambiente vuoto

[ ] : memoria vuota

new: Store $\longrightarrow$ Loc , funzione tale che new x for= nisce la prima (in un prefissato ordine) loca= zione che nello Store x vale unused, error se tale locazione non c'è .

J : CMD $\longrightarrow$ Env , funzione di legame per la eti= chetta go to come vedremo avanti

7) **Tutte le operazioni** di sopra si intendono **error-estese** cioè forniscono error se prendono error come argo= mento (error è un elemento particolare che sarà intro= dotto fra poco .

Osserviamo che la semantica del go to sarà trat= tata in base ai seguenti principi:

— La semantica di un comando in un dato stato e in un dato ambiente fornisce uno stato e una marca di se= quenza (Jump o Continue) che informa se nella seman= tica del comando sono utilizzate equazioni semantiche di go to .

— Vi è una funzione J: CMD $\longrightarrow$ Env che lega ogni eti= chetta di go to con un opportuno comando

— Il campo di legame di ogni etichetta è il più pic= colo blocco begin/end in cui essa compare .

— Si può saltare entro le alternative di un condizio= nale ed entro il campo di un While

— Se un'etichetta compare in entrambi le alternative di un condizionale o in entrambi i comandi di un co= mando composto $C_1;C_2$ allora è considerata solo la seconda occorrenza dell'etichetta.

— A ogni etichetta L è legato

i) il comando C che segue L nel suo campo di le= game se L non compare nel corpo di un comando (<u>While</u> ...)

ii) C;<u>While</u> (...) altrimenti

La definizione di J è quindi esprimibile per in= duzione come segue:

$J(I:= E) = J(\ P(E)\ ) = J(\underline{outputE}) = J(\underline{goto}\ L) =$

$= J(\underline{begin}\ D;C\ \underline{end}) = ()$

$J(I:C) = J(C)[C/I]$

$J(\underline{if}\ E\ \underline{then}\ C_1\ \underline{else}\ C_2) = J(C_1)\ [J(C_2)]$

$J(C_1;C_2)L = J(C_2)L = \text{unbound} \supset [\ J(C_1)L = \text{unbound} \supset \text{unbound},$

$(\ J(C_1)L;C_2\ )],\ J(C_2)L$

$J(\underline{While}\ E\ \underline{do}\ C) = J(C)L = \text{unbound} \supset \text{unbound},$

$(\ J(C)L;(\underline{While}\ E\ \underline{do}\ C)\ )$

È una cosa ovvia definire una segnatura Σ disgiunta dalla segnatura sintattica Σ_0 tale che $\underline{B}$ risulti una Σ-algebra.

Sia Σ' la segnatura ottenuta congiungendo Σ_0 e Σ e aggiungendo degli operatori interpretativi $\underline{\mathcal{E}}$, $\underline{\mathcal{C}}$, $\underline{\mathcal{R}}$, $\underline{\mathcal{D}}$, $\underline{\mathcal{J}}$, $\underline{\mathcal{P}}$ relativi alle seguenti funzioni di interpretazione

$$\underline{\mathcal{E}} : EXP \times Env \times Sta \longrightarrow (Ev \times Sta) + \{error\}$$

$$\underline{\mathcal{R}} : EXP \times Env \times Sta \longrightarrow Ev + \{error\}$$

$$\underline{\mathcal{C}} : CMD \times Env \times Sta \longrightarrow (Sta \times Seq) + \{error\}$$

$$\underline{\mathcal{D}} : DEC \times Env \times Sta \longrightarrow (Env \times Sta) + \{error\}$$

$$\underline{\mathcal{P}} : PRG \times Iv \longrightarrow Ov + \{error\}$$

Equazioni semantiche

Nel seguito abbrevieremo $(x,(y\ldots z))$ con $(x,y\ldots z)$

$$\underline{\mathcal{E}}(\underline{c})\delta\varphi = c$$

$$\underline{\mathcal{E}}(I)\delta\varphi = \delta I = \text{unbound} \supset \text{error}, \delta I$$

$$\underline{\mathcal{E}}(\underline{True})\delta\varphi = True$$

$$\underline{\mathcal{E}}(\underline{False})\delta\varphi = False$$

$$\underline{\mathcal{R}}(E)\delta\varphi = \underline{\mathcal{E}}(E)\delta\varphi \in Loc \supset \Big[(\varphi \downarrow Store)(\underline{\mathcal{E}}(E)\delta\varphi) = unused \supset \text{error}, (\varphi \downarrow Store)(\underline{\mathcal{E}}(E)\delta\varphi)\Big], \underline{\mathcal{E}}(E)\delta\varphi$$

$$\underline{\mathcal{E}}(\underline{f}E)\delta\varphi = \underline{\mathcal{R}}(E)\delta\varphi \in A \supset f(\underline{\mathcal{R}}(E)\delta\varphi), error$$

$$\underline{\mathcal{E}}(\underline{read})\delta\varphi = \varphi\downarrow Iv = \lambda \supset error, (first(\varphi \downarrow Iv), \varphi[rest(\varphi \downarrow Iv)/\ \varphi \downarrow Iv])$$

$$\underline{\mathcal{E}}(\underline{\text{if}}\ E\ \underline{\text{then}}\ E_1\ \underline{\text{else}}\ E_2)\delta\varrho = \underline{\mathcal{R}}(E) \in Bv \supset \big[\underline{\mathcal{R}}(E)\delta\varrho \supset \underline{\mathcal{E}}(E_1)\delta\varrho,\ \underline{\mathcal{E}}(E_2)\delta\varrho\big]\ ,\text{error}$$

$$\underline{\mathcal{C}}(I:=E)\delta\varrho = \varrho I \in Loc \supset (\ (\varrho \downarrow Store)\big[\underline{\mathcal{R}}(E)\delta\varrho/\varrho I\big]\ ,\ \varrho \downarrow Iv,\ \varrho \downarrow Ov,\ Continue\ ),\ error$$

$$\underline{\mathcal{C}}(\underline{\text{Output}}\ E)\delta\varrho = (\varrho\big[\varrho \downarrow Ov \cup \underline{\mathcal{R}}(E)\delta\varrho/\varrho \downarrow Ov\big]\ ,\ Continue\ )$$

$$\underline{\mathcal{C}}(\underline{\text{if}}\ E\ \underline{\text{then}}\ C_1\ \underline{\text{else}}\ C_2)\delta\varrho = \underline{\mathcal{R}}(E)\delta\varrho \in Bv \supset \big[\underline{\mathcal{R}}(E)\delta\varrho \supset \underline{\mathcal{C}}(C_1)\delta\varrho,\ \underline{\mathcal{C}}(C_2)\delta\varrho\ \big],\ error$$

$$\underline{\mathcal{C}}(I:C)\delta\varrho = \underline{\mathcal{C}}(C)\delta\varrho$$

$$\underline{\mathcal{C}}(\underline{\text{goto}}\ I)\delta\varrho = \delta I \in CMD \supset (\ \varkappa(\delta I)\delta\varrho \downarrow Sta\ \ ,Jump),\ error$$

$$\underline{\mathcal{C}}(C_1;C_2)\delta\varrho = \underline{\mathcal{C}}(C_1)\delta\varrho \downarrow Seq = Jump \supset \underline{\mathcal{C}}(C_1)\delta\varrho,\ \underline{\mathcal{C}}(C_2)\delta(\ \varkappa(C_1)\delta\varrho \downarrow Sta\ )$$

$$\underline{\mathcal{C}}(\underline{\text{While}}\ E\ \underline{\text{do}}\ C)\delta\varrho = \underline{\mathcal{R}}(E)\delta\varrho \in Bv \supset \big[\underline{\mathcal{R}}(E)\delta\varrho \supset \big[\underline{\mathcal{C}}(C)\delta\varrho \downarrow Seq=Jump \supset \underline{\mathcal{C}}(C)\delta\varrho\ ,\ \underline{\mathcal{C}}(\underline{\text{While}}\ E\ \underline{\text{do}}\ C)\delta(\ \underline{\mathcal{C}}(C)\delta\varrho \downarrow Sta\ )\big]\ ,\ (\varrho, Continue)\ \big],\ error$$

$$\underline{\mathcal{C}}(P(E))\delta\varrho = \varkappa(\delta P \downarrow CMD)\ \Psi\ (P, \Phi(E), \delta, \varrho)$$

$$\underline{\mathcal{E}}(F(E))\delta\varrho = \mathcal{E}(\delta F \downarrow EXP)\ \Psi\ (F, \Phi(E), \delta, \varrho)$$

ove Ψ fornisce ambiente e stato secondo il passaggio dei parametri e la valutazione del parametro attuale, $\Phi(E)$ (cfr. oltre)

$$\underline{\mathcal{C}}(\underline{\text{begin}}\ D;C\ \underline{\text{end}})\delta\varrho = \underline{\mathcal{C}}(C)(\ (\underline{\mathcal{D}}(D)\delta\varrho \downarrow Env)\big[\mathcal{J}(C)\big]\big)(\underline{\mathcal{D}}(D)\delta\varrho \downarrow Sta)$$

$$\underline{\mathcal{D}}(\underline{\text{const}}\ I=E)\delta\varrho = (\ \delta\big[\underline{\mathcal{R}}(E)\delta\varrho/I\big]\ ,\varrho\ )$$

$$\underline{\mathbb{D}}(\underline{\text{var}}\ I{=}E)\delta\varrho = new(\varrho \downarrow Store){=} error \supset error,$$
$$(\ \delta[\ I/new(\varrho \downarrow Store)]\ ,$$
$$\varrho[\underline{\mathbb{R}}(E)\delta\varrho/new(\varrho \downarrow Store)]\ )$$

$$\underline{\mathbb{D}}(\underline{\text{proc}}\ P(I);C)\delta\varrho = (\delta[\ (C,P,I,\delta)/I\ ],\varrho\ )$$

$$\underline{\mathbb{D}}(\underline{\text{fun}}\ F(I);E)\delta\varrho = (\ \delta[\ (E,F,I,\delta)/I\ ],\varrho\ )$$

$$\underline{\mathbb{D}}(D_1;D_2)\delta\varrho = \underline{\mathbb{D}}(D_2)\ (\delta\ [\underline{\mathbb{D}}(D_1)\delta\varrho \downarrow Env\ ])(\ \varrho[\underline{\mathbb{D}}(D_1)\delta\varrho \downarrow Sta\ ])$$

$$\underline{\mathbb{P}}(\underline{\text{program}}\ C)i = (\ (\underline{\mathbb{C}}(C)()([\,],i,\lambda\ )\ )\downarrow OV,\ Stop\ )$$

Osserviamo che nella semantica della chiamata di procedura la definizione di Φ determina il tipo di valutazione e quella di Ψ il tipo di passaggio dei parametri secondo una **valutazione per valore** e un **passaggio per risultato** , *cioè* :

$$\begin{cases} \mathbb{D}(E)\delta\varrho = \underline{\mathbb{E}}(E)\delta\varrho \quad \text{(valutazione del parametro attuale)} \\ \Psi(P,\Phi(E),\delta,\varrho) = (\delta P \downarrow Env\ [\mathbb{D}(E)\delta\varrho/I]\ ,\varrho\ ) \end{cases}$$

Si può comunque definire $\mathbb{D}$ in altri modi ad esempio **per testo** o **per nome** ponendo rispettivamente $\mathbb{D}(E)\delta\varrho = E$ o $\mathbb{D}(E)\delta\varrho = (E,\delta)$, in tal caso si deve però ammettere che gli identificatori (parametri formali) possano essere legati ad entità quali testi o nomi.

Bisogna quindi estendere l'equazione semantica di $\underline{\mathbb{E}}(I)\delta\varrho$ (e conseguentemente i domini semantici)ponendo:

per testo

$$\underline{\mathbb{E}}(I)\delta\varrho = \delta I = unbound \supset error, (\delta I \in EXP \supset \underline{\mathbb{E}}(\delta I)\delta\varrho,\ \delta I)$$

per nome

$\underline{\xi}(I)\delta\varrho = \delta I = \text{unbound} \supset \text{error},$

$(\delta I = (E,\delta') \in EXP \times Env \supset \underline{\xi}(E)\delta'\varrho,\ \delta I)$

Analogamente data una definizione di $\mathfrak{D}$ si può de=finire la semantica con altri tipi di passaggi, per esem=pio **per valore** o **valore/risultato** ponendo:

i) **per valore** $\underline{\xi}(P(E))\delta\varrho =$

$= \xi(\delta P{\downarrow}CMD)(\delta P{\downarrow}Env\,[\,new(\varrho{\downarrow}Store)/\delta P{\downarrow}ID\,],$

$\varrho\,[\,\varrho{\downarrow}Store\,[\,\mathfrak{D}(E)\delta\varrho/new(\varrho{\downarrow}Store)\,]/\varrho{\downarrow}Store\,])$

ii) **per valore/risultato**

$\underline{\xi}(P(E))\delta\varrho = \underline{\xi}(E)\delta\varrho \in Loc \supset$

$\supset\ \#\,[\#{\downarrow}Store\,[\#{\downarrow}Store(new(\#{\downarrow}Store))/\underline{\xi}(E)\delta\varrho]]$

error

(# è il 2º membro di i))

Notiamo ancora che se nelle definizioni di Ψ si pone δ al posto di $\delta P{\downarrow}Env$ si ha una chiamata di pro=cedura con binding dinamico piuttosto che statico.

Il discorso è analogo nel caso della chiamata di funzione laddove τ è sostituito da ξ e CMD da EXP.

Tale specifica algebrica della semantica dello SMALL è consistente come si può verificare utilizzando metodi descritti in [26], [27] e inoltre esiste (cfr. [27]) l'algebra $\underline{\underline{I}}$ iniziale nella classe di tutte le Σ'_B-alge=bre che soddisfano le date equazioni semantiche.

A partire da $\underline{\xi}^I, \mathfrak{R}^I, \underline{\tau}^I, \mathfrak{D}^I, \mathfrak{P}^I$ si possono quindi definire le funzioni di interpretazione $\xi, \mathfrak{R}, \tau, \mathfrak{D}, \mathfrak{P}$, secondo quanto si è detto precedentemente (cfr. $\underline{\underline{A}}$-While).

Inoltre a partire da $\mathcal{P}$ si può definire una funzione di interpretazione $\mathcal{P}_\infty$ per programmi che non terminano fornendo una sequenza infinita di valori.

Basta porre a tal fine

$$\mathcal{P}_\infty(\underline{\text{program}}\ C)i = (\ \mathcal{P}(\underline{\text{program}}\ C)i \in A^* \times \{\text{Stop}\} \supset$$
$$\supset \mathcal{P}(\underline{\text{program}}\ C)i,$$
$$(\{t_o^I \mid \underline{I} \models \mathcal{C}(C)()(\square, i, \lambda) =$$
$$= (t_p, t_i, t_o, t_s)\}, \sqsubset)$$

ove t_o^I è la valutazione di t_o in $\underline{I}$ e $\sqsubset$ è l'ordinamento per lunghezza di A^* (t_o^I è un elemento di A^*).