INTRODUZIONE

0.- L'analisi matematica dei linguaggi formali è di interesse cruciale in varie discipline: logica, linguistica, informatica. Nel caso dei linguaggi programmativi (cfr. [22]) i problemi di affidabilità, efficienza e sviluppo della programmazione hanno evidenziato una serie di questioni ben precise e innanzitutto la ricerca di strumenti di specifica formale delle nozioni linguistiche.

Senza addentrarci in sottili problemi di semiotica richiamiamo semplicemente la classica distinzione degli aspetti fondamentali di un linguaggio:

i) sintassi, che riguarda le regole di costituzione delle forme segniche
ii) semantica, che riguarda la determinazione dei significati veicolati dai segni
iii) pragmatica, che riguarda le finalità e gli usi linguistici dei segni.

Nella definizione di un linguaggio formale intervengono solo i primi due aspetti essendo la pragmatica o a monte (le motivazioni per cui il linguaggio è definito) o a valle (la sua utilizzazione una volta che è definito) della definizione in sè e per sè.

La sintassi è evidentemente l'aspetto più "esterno" e questo spiega il perchè la storia della specifica formale dei linguaggi programmativi comincia con i metodi rigorosi di descrizione della sintassi (cfr. [8])derivati a sua volta dagli studi di formalizzazione della sintassi dei linguaggi naturali.(cfr.[13])

Dopo una serie di lavori pionieristici(Dijkstra, McCarthy, Landin, Scott-Strachey, Floyd, Hoare[15,28,24,32, 17, 20]) si sono delineati i principali metodi di specifica formale della semantica (cfr. [10]), al fine di

determinare strumenti matematici di valutazione e pro=gettazione della programmazione. I due temi fondamentali attorno a cui si è sviluppata tale ricerca sono la cor=rettezza (cfr. [11]) e l'astrazione (cfr. [9]).

1.- Il problema della correttezza nasce con questioni del tipo:

a) Un dato programma calcola effettivamente la funzione che si intende calcolare tramite esso, o anche, veri=fica certe proprietà desiderate ?

E più in generale

b) Quali sono i criteri che in qualche modo garantiscono una programmazione coerente con i suoi obiettivi ?

L'astrazione consiste nella tendenza sempre più diffusa di sganciare la programmazione da un prefissato modello implementativo, per esempio:

a') Specificare un tipo astratto di dati significa de=scrivere certi oggetti solo in quanto manipolabili con operazioni che verificano certe condizioni desi=derate, prescindendo da qualsiasi loro rappresenta=zione concreta.

b') Specificare astrattamente un programma significa definire (in un certo linguaggio di specifica) la funzione che si intende calcolare senza bisogno di esibire una qualche particolare procedura di calco=lo.

La specifica astratta è, per sua natura, utiliz=zabile come strumento per costruzione di programmazione corretta,in quanto permette una strutturazione per mo=duli con una organizzazione a livelli successivi di astrazione nei confronti del sistema fisico di calcolo (cfr. [1]) .

Da un punto di vista logico una specifica astratta

definisce una teoria, formale o informale, la cui classe di modelli individua possibili realizzazioni concrete della specifica. In linea di principio anche un intero linguaggio di programmazione potrebbe essere descritto tramite un linguaggio di specifica che non appena implementato in modo efficiente determinerebbe(la cosa è ancora esistente solo a livello sperimentale) un salto di qualità nella programmazione, ovvero l'uso di linguaggi non più ad alto, ma ad "altissimo" livello, per una "programmazione automatica avanzata".

In effetti tale salto di qualità è in linea con la tendenza per la quale in passato si è passati dalla macchina calcolatrice su cui i calcoli venivano eseguiti direttamente (su tastiera o simili) dal programmatore, alla macchina in cui il calcolo è descritto con un programma e fornito insieme ai dati.

2.- Nei metodi formali di specifica per la semantica di linguaggi programmativi si distinguono due tipi fondamentali di approcci:

- operativo, in base al quale la semantica di un linguaggio è data tramite un qualche sistema formale (i.e. di manipolazione simbolica)
- interpretativo, in base al quale la semantica di un linguaggio è data tramite una qualche struttura matematica.

Tale dicotomia potrebbe essere espressa con altri termini antitetici quali nominalista/realista o formalista/contenutista. Tali opposizioni sono infatti legate a problematiche ricorrenti nelle dispute filosofico-scientifiche sul linguaggio. Sarebbe interessante,ma al di fuori dell'ambito di questo lavoro,rintracciare le analogie tra la problematica sulla specifica formale in programmazione e la questione dei fondamenti della matematica (in cui era praticamente in discussione la semantica del linguag=

gio matematico),o il dibattito degli scolastici sugli universali, relativo alla semantica del linguaggio naturale.

Nella bipartizione di sopra si inquadra la usuale classificazione dei principali indirizzi attuali: operazionale, denotazionale e assiomatico.

- La semantica operazionale (cfr. [34])associa al linguaggio considerato una macchina astratta, facilmente descrivibile tramite un sistema formale, che rappresenti il comportamento ingressi/uscite del linguaggio. Tale semantica è stata la prima ad essere formulata e applicata permettendo di risolvere il problema di base della semantica formale: la descrizione rigorosa di un linguaggio programmativo. Tramite essa attraverso una serie di macchine astratte sempre più vicine al sistema, si dispone di un metodo rigoroso per la implementazione di linguaggi.

- La semantica assiomatica (cfr. [7])è anch'essa di tipo operativo e associa al dato linguaggio di programmazione :
  i) un linguaggio logico di "asserzioni" che rappresentano proprietà o relazioni tra programmi;
  ii)delle regole di inferenza mediante cui da certe asserzioni si deducono altre asserzioni.
  È evidente che tale tipo di approccio è orientato alle dimostrazioni di correttezza, ma è pur vero che individua principi generali di programmazione nella misura in cui certe metodologie di costruzione di programmi consentono una applicazione naturale dei metodi dimostrativi.

- La semantica denotazionale (cfr.[10][18]) è di tipo interpretativo nel senso che associa ai costrutti linguistici del dato linguaggio tramite funzioni di interpretazione, dei significati astratti, ovvero elementi di opportune strutture matematiche. In tal modo è pos=

sibile una analisi matematica della struttura del linguaggio in sè e per sè; il problema della correttezza è così affrontato con criterio opposto a quello della semantica assiomatica. Contemporaneamente la semantica denotazionale di un linguaggio costituisce naturalmente una specifica astratta del suo funzionamento.

3.- Oltre ai tre approcci sopra indicati vi sono varie soluzioni miste, più o meno collocabili in un ambito preciso.

L'indirizzo algebrico è però quello che sintetizza naturalmente i due aspetti , interpretativo e operativo .

Esso si è sviluppato di recente sia come diramazione dell'approccio denotazionale (cfr. [1] [5]) sia come metodo operativo basato sui sistemi di manipolazione di termini o alberi (cfr. [14] [23] [30]), ma attualmente riunendo i due aspetti si configura come metodo di specifica formale di grande generalità (cfr. [12] [16] [33]).

Secondo la semantica algebrica, ad un dato linguaggio è associata una teoria (algebrica) che descrive astrattamente il significato dei costrutti linguistici. Ogni modello di tale teoria fornisce una particolare formalizzazione matematica del linguaggio. Inoltre essendo la teoria algebrica espressa in un linguaggio formale con una naturale nozione di deducibilità (calcolo equazionale) la semantica di un costrutto può essere ottenuta operativamente. Tale schema è facilmente generalizzabile utilizzando teorie non più algebriche, ma logiche di tipo del tutto generale, dotate di nozioni di deducibilità e soddisfacibilità (logica dinamica, temporale).

Per tale motivo la semantica algebrica risulta

i) un nucleo di aggregazione dei metodi tradizionali di semantica formale,

ii) un paradigma di sviluppo per ricerche future rivolte ad aspetti programmativi in fase di formalizzazione (parallelismo, concorrenza).

Inoltre i metodi algebrici consentono di stabilire collegamenti e criteri unificanti tra lo studio di linguaggi programmativi e quello di linguaggi logici (cfr. [25]) ed il che è di interesse sia teorico (logica astratta, logiche non classiche) che applicativo (intelligenza artificiale, basi di dati).

Gli strumenti di base della semantica algebrica sono quelli dell'algebra universale (cfr. [19]).

Nel seguito daremo nella I^{a} parte una presentazione sistematica delle nozioni fondamentali di algebra universale, e nella II^{a} parte esempi di specifica algebrica di nozioni e linguaggi di programmazione.

Il punto di arrivo sarà l'applicazione di un nuovo metodo, completamente algebrico , per la specifica formale di linguaggi programmativi .
Tale risultato si fonda su un'estensione del classico teorema di Birkhoff di esistenza di algebra iniziale per varietà (cfr. [27]) e consente di esprimere la semantica per integrazione piuttosto che per interpretazione . Infatti la sintassi di un linguaggio risulta un'algebra che riceve significato integrandosi con un'altra algebra (base) supposta nota e generalmente di diverso tipo di similarità algebrica . L'integrazione è definita da una teoria algebrica che esprime tramite "operatori interpretativi" come il nuovo si riduce al noto . Sotto certe ipotesi del tutto generali su tali teorie le semantiche così ottenute risultano corrette .

I

ALGEBRE ETEROGENEE E SEGNATURE

0.- DEFINIZIONE

Dicesi algebra (eterogenea) una coppia $(\mathfrak{J}, O)$ ove $\mathfrak{J}$ è una famiglia di insiemi e O una famiglia di operazioni (totali) i cui domini sono prodotti cartesiani tra insiemi di $\mathfrak{J}$, e i codomini sono insiemi di $\mathfrak{J}$.

Se A è un insieme di $\mathfrak{J}$, una costante di A è un elemento $a \in A$ che può essere completamente individuato con una operazione

$$f_a: \{\mathfrak{D}\} \longrightarrow A$$

(del tutto definita dal suo valore in $\mathfrak{D}$) ponendo $f_a(\mathfrak{D}) = a$.

Poichè $\{\mathfrak{D}\} = \{f \mid f: \mathfrak{D} \longrightarrow A\} = A^{\circ}$ si può concludere che in un'algebra le costanti sono identificabili con operazioni nullarie.

Gli insiemi della famiglia $\mathfrak{J}$ diconsi domini o sostegni dell'algebra.

1.- ESEMPIO

$$\underline{\underline{L}} = (L, N, B, \vee, ||, +, =, \lambda, 0, 1, T, F)$$

$L \equiv \{a,b\}^*$ i,e l'insieme di stringhe su a,b
$N \equiv$ l'insieme dei naturali $0,1,2,\ldots$
$B \equiv$ l'insieme dei valori di verità T,F

$\vee : L \times L \longrightarrow L$	ove	$\alpha \vee \beta$ è la concatenazione delle stringhe α e β
$\|\| : L \longrightarrow N$	ove	$\|\alpha\|$ è la lunghezza di α
$+ : N \times N \longrightarrow N$	"	$x+y$ è la somma di x e y
$= : L \times L \longrightarrow B$	"	$(\alpha = \beta) = T \iff \alpha$ è uguale a β

(segue)

$\lambda \in L$ (stringa vuota) i.e. $\lambda : L^{o} \longrightarrow L$

$0,1 \in N$ i.e. $0,1: N^{o} \longrightarrow N$

$T,F \in B$ i.e. $T,F: B^{o} \longrightarrow B$

2.- DEFINIZIONE

Una algebra $\underline{\underline{L}}'$ dicesi simile ad $\underline{\underline{L}}$ se i suoi domini sono degli insiemi L', N', B', in corrispondenza a L, N, B, rispettivamente e le sue operazioni $\vee'$, $|\,|'$, $+'$, $='$, λ', $0'$, $1'$, T', F', in corrispondenza a quelle di $\underline{\underline{L}}$ sono tali che operazioni corrispondenti agiscono tra domini corrispondenti, cioè

$$\vee' : L' \times L' \longrightarrow L'$$

$$|\,|' : L' \longrightarrow N'$$

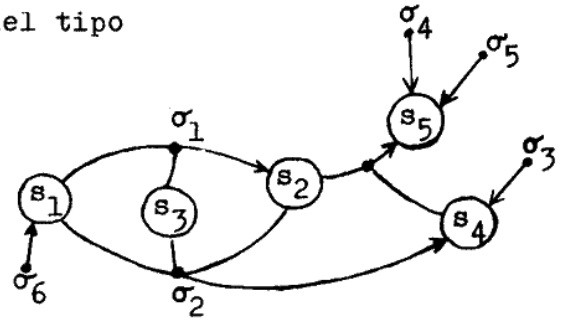
$$+' : N' \times N' \longrightarrow N'$$

$$=' : L' \times L' \longrightarrow B'$$

$$\lambda' \in L' \qquad 0',1' \in N' \qquad T',F' \in B'$$

3.- DEFINIZIONE

Un metodo per trattare comodamente la similarità tra algebre è quello di individuare tipi di similarità con multi= grafi dette <u>segnature</u> , ovvero insiemi di nodi collegati da multifrecce aventi più nodi di partenza e un nodo di arrivo, del tipo



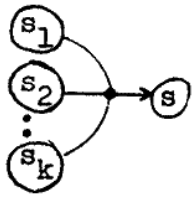
s_1, s_2, s_3, s_4, s_5, diconsi <u>sorte</u> o tipi

$\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6$, diconsi <u>operatori</u>

4. DEFINIZIONE

Una algebra $\underline{\underline{A}}$ dicesi di segnatura Σ ,o Σ-algebra se in $\underline{\underline{A}}$ vi sono tanti domini A_{s_1}, A_{s_2}, ... quante sono le sorte s_1, s_2, ... della segnatura Σ e tante operazioni σ_1^A, σ_2^A, ... quante sono le multifrecce (operatori) di Σ in modo tale che se σ è una multifreccia di rango

$$s_1\ s_2\ \ldots\ s_k \longrightarrow s \qquad \text{cioè}$$



(ovvero arietà $s_1 s_2 \ldots s_k$ e sorta s)

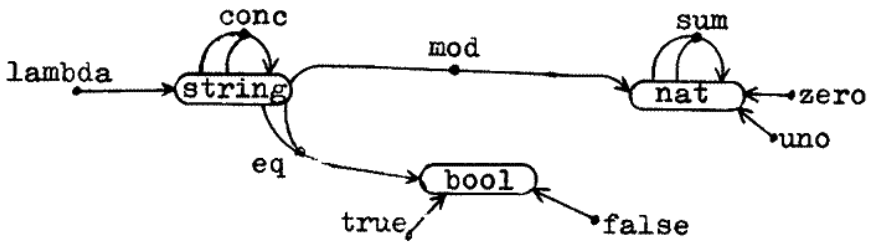
allora in $\underline{\underline{A}}$ si avrà

$$\sigma^A : A_{s_1} \times A_{s_2} \times \ldots \times A_{s_k} \longrightarrow A_s$$

(Le multifrecce senza sorte di partenza individuano operazioni nullarie, cioè costanti)

5. ESEMPIO

L'algebra $\underline{\underline{L}}$ dell'esempio 1 è una Σ-algebra per la seguente segnatura Σ



ove :

$L_{string} = L$

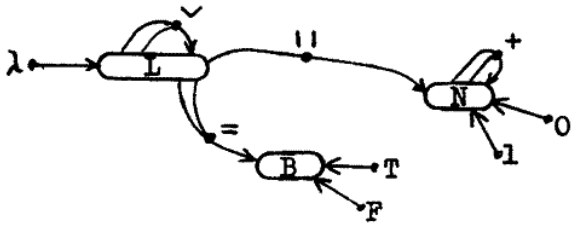
$L_{nat} = N$

$L_{bool} = B$

$conc^L = \vee$	$eq^L = =$	$uno^L = 1$
$mod^L = \|\|$	$lambda^L = \lambda$	$true^L = T$
$sum^L = +$	$zero^L = 0$	$false^L = F$

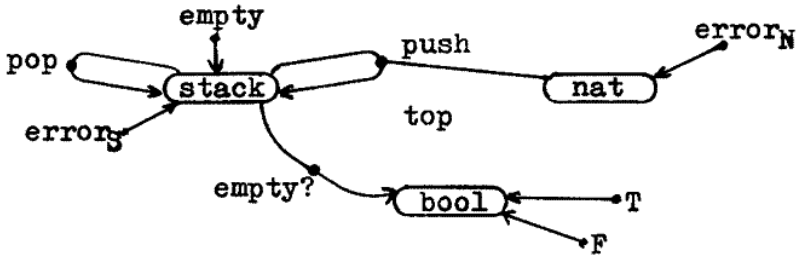
ovvero Σ fornisce un sistema di notazioni per domini e operazioni di $\underline{\underline{L}}$.

Possiamo a questo punto indicare la Σ-algebra $\underline{\underline{L}}$ con un grafo analogo a quello di sopra ove ai nodi sono associati insiemi e alle multifrecce sono associate ope= razioni:



6. ESEMPIO

Sia Σ la seguente segnatura



$\underline{\underline{P}}$ è una Σ-algebra, ove

$$\begin{cases} P_{stack} = N^* \cup \{error_S\} \\ P_{nat} = N \cup \{error_N\} \\ P_{bool} = B = \{T,F\} \end{cases}$$

$push^P : P_{nat} \times P_{stack} \longrightarrow P_{stack}$

$pop^P : P_{stack} \longrightarrow P_{stack}$

$top^P : P_{stack} \longrightarrow P_{nat}$

$empty?^P : P_{stack} \longrightarrow P_{bool}$

$error_S^P = error_S \quad ; \quad error_N^P = error_N$

$empty^P = \lambda$ (stringa nulla)

$T^P = T \quad ; \quad F^P = F$

$push^P(n, n_1 n_2 \ldots n_k) = (n n_1 n_2 \ldots n_k)$

$pop^P(n n_1 n_2 \ldots n_k) = (n_1 n_2 \ldots n_k)$

$top^P(n n_1 n_2 \ldots n_k) = n$

$push^P(error_N, n_1 n_2 \ldots n_k) = push^P(n, error_S) = error_S$

$pop^P(error_S) = pop^P(empty) = error_S$

$top^P(error_S) = top^P(empty) = error_N$

$empty?^P(S) = T \iff S = empty^P$

7.- CONVENZIONI NOTAZIONALI

1) Nel seguito Σ indica una generica segnatura di sorte $\mathcal{S}$, inoltre u indicherà un generico elemento di $\mathcal{S}^*$, ovvero

$$u = s_1 s_2 \ldots s_k \qquad (k = 0 \Rightarrow u = \lambda)$$

con $s_1, s_2, \ldots s_k \in \mathcal{S}$ (insieme di sorte)

2) Sia $(A_s \mid s \in \mathcal{S})$ una famiglia di insiemi $\mathcal{S}$-indicizzata e $(f_s \mid s \in \mathcal{S})$ una famiglia di funzioni $\mathcal{S}$-indicizzata

A_u sta per $A_{s_1} \times A_{s_2} \times \ldots A_{s_k}$ $\quad (A_\lambda = \{\emptyset\})$

f_u " " $f_{s_1} \times f_{s_2} \times \ldots f_{s_k}$ $\quad (f_\lambda : A_\lambda \rightarrow \quad)$

ove

$$x \in A_u \iff x = (x_1, x_2, \ldots x_k) \text{ e } x_1 \in A_{s_1} \ldots x_k \in A_{s_k}$$

$$f_u x = (f_{s_1} x_1, f_{s_2} x_2, \ldots f_{s_k} x_k)$$

3) Si abbrevia con $\sigma \in \Sigma$ la scrittura più precisa

$$\sigma \in \bigcup_{\substack{s \in \mathcal{S} \\ u \in \mathcal{S}^*}} \Sigma_{u,s}$$

spesso generalizzaremo tale abbreviazione ad una famiglia qualsiasi $\mathcal{F} = (\mathcal{F}_i \mid i \in I)$, scrivendo $f \in \mathcal{F}$ per $f \in \bigcup_{i \in I} \mathcal{F}_i$

8.- OSSERVAZIONE

Una segnatura può essere equivalentemente definita come una famiglia di insiemi disgiunti

$$\Sigma = (\Sigma_{u,s} \mid u \in \mathcal{S}^* , s \in \mathcal{S})$$

($\mathcal{S}$ insieme delle sorte e $\Sigma_{u,v}$ degli operatori di rango $u \rightarrow s$).

In tal modo una Σ-algebra è individuata da una coppia

$$(A, \alpha)$$

ove

$$A=(A_s \mid s \in \mathcal{S}) \qquad e \qquad \alpha =(\sigma^A \mid \sigma \in \Sigma)$$

($\Sigma_{\lambda,s}$ o più semplicemente Σ_s indica l'insieme degli operatori nullari di Σ).

È evidente che algebre di uguale segnatura sono simili e che viceversa date delle algebre simili può trovarsi una segnatura Σ per cui tutte risultino Σ-algebre.

9.- ESERCIZIO

Definire delle segnature e algebre di quelle segnature.

10.- CONVENZIONE

Data una segnatura Σ , Σ-alg indica la classe di tutte le Σ-algebre.

II

SOTTOALGEBRE E ALGEBRE PRODOTTO

Sia $\underline{\underline{L}}$ l'algebra dell'Es.1 privata della costante 1, siano

$$L' \subset L \qquad e \qquad N' \subset N$$

ove $\quad N' = \{x \mid x \in N \quad e \quad x \text{ pari}\}$

$$L' = \{\alpha \mid \alpha \in L \quad e \quad |\alpha| \in N'\}$$

è evidente che

$$\alpha, \beta \in L' \Rightarrow \alpha \vee \beta \in L'$$

$$\alpha \in L' \Rightarrow |\alpha| \in N'$$

$$x, y \in N' \Rightarrow x + y \in N'$$

cioè L', N' sono chiusi rispetto a $\vee, |\ |, +$.

In tal caso possiamo definire una algebra $\underline{\underline{L}}'$

$$\underline{\underline{L}}' = (L', N', B, \vee', |\ |', +', =', \lambda, 0, 1, T, F)$$

ove $\vee', |\ |', +', ='$ sono le restrizioni di $\vee, |\ |, +, =$, ai relativi insiemi con apice.

$\underline{\underline{L}}'$ dicesi quindi sottoalgebra di $\underline{\underline{L}}$. In generale si ha la seguente

11.- DEFINIZIONE

Sia A una Σ-algebra e sia $B = (B_s \mid s \in \mathcal{S})$

Su B è definibile una Σ-algebra $\underline{\underline{B}}$ che dicesi Σ-sottoalgebra di $\underline{\underline{A}}$ scrivendo $\underline{\underline{B}} \subseteq \underline{\underline{A}}$ se valgono le seguenti condizioni:

$$\text{i)} \quad B_s \subseteq A_s \quad \forall\, s \in \mathcal{S} \qquad \begin{cases} \forall\, u,s \in \mathcal{S}^* \\ \forall\, x \in B_u \\ \forall\, \sigma \in \Sigma_{u,s} \end{cases}$$

$$\text{ii)} \quad x \in B_u \Rightarrow \sigma^A x \in B_s$$

$$\text{iii)} \quad \sigma^B = \sigma^A \Big|_{B_u}$$

($\sigma^A \Big|_{B_u}$ è la restrizione di σ^A a B_u)

Dalla condizione ii) segue $\Sigma_s \subseteq B_s \quad \forall\, s \in \mathcal{S}$.

12.- DEFINIZIONE

Siano $\underline{\underline{A}}$, $\underline{\underline{B}}$ due Σ-algebre, dicesi <u>Σ-algebra prodotto</u> di $\underline{\underline{A}}$ e $\underline{\underline{B}}$ la seguente algebra $\underline{\underline{A}} \times \underline{\underline{B}}$

$$\underline{\underline{A}} \times \underline{\underline{B}} = \left((A_s \times B_s \mid s \in \mathcal{S}) \, , \, (\sigma^A \times \sigma^B \mid \sigma \in \Sigma) \right)$$

avente come domini i prodotti dei domini di $\underline{\underline{A}}$ con i corrispondenti di $\underline{\underline{B}}$ e come operazioni i prodotti tra operazioni corrispondenti di $\underline{\underline{A}}$ e $\underline{\underline{B}}$

Tale definizione si generalizza in modo del tutto ovvio a prodotti qualsiasi di famiglie di algebre.

III

MORFISMI E CONGRUENZE

Consideriamo le due algebre simili

$$\underline{\underline{A}} = (N^*, \cup)$$

$$\underline{\underline{B}} = (B^*, \vee)$$

ove $\cup$ indica la concatenazione tra stringhe di naturali e $\vee$ la concatenazione tra stringhe di booleani (T,F) .

Sia $h: N \longrightarrow B^*$

e $h^*: N^* \longrightarrow B^*$

tale che

$$h^*(n_1 n_2 \ldots n_k) = h(n_1) \vee h(n_2) \vee \ldots h(n_k)$$

se $\alpha , \beta \in N^*$ h^* verifica la seguente uguaglianza

$$h^*(\alpha \cup \beta) = h^*(\alpha) \vee h^*(\beta)$$

Possiamo esprimere tale proprietà tramite il diagramma:

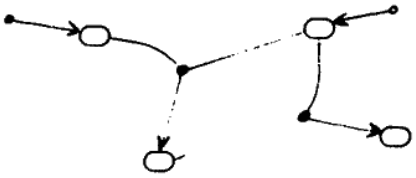
$$\begin{array}{ccc} N^* \times N^* & \xrightarrow{\cup} & N^* \\ \downarrow{\scriptstyle h^* \times h^*} & & \downarrow{\scriptstyle h^*} \\ B^* \times B^* & \xrightarrow{\vee} & B^* \end{array}$$

dicendo che esso è commutativo nel senso che partendo da un elemento $(\alpha , \beta) \in N^* \times N^*$ operando secondo il cammino $\cup , h$ si ottiene lo stesso risultato che si ottiene secondo il cammino $h^* \times h^* , \vee$

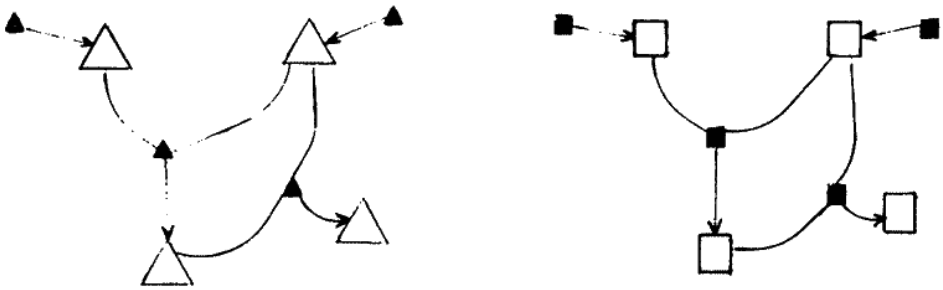
Delle funzioni, tra domini corrispondenti di Σ-algebre, che commutano per ogni operatore il relativo diagramma costituiscono un morfismo

Esprimiamo tale nozione mediante grafi, sia data una

segnatura quale

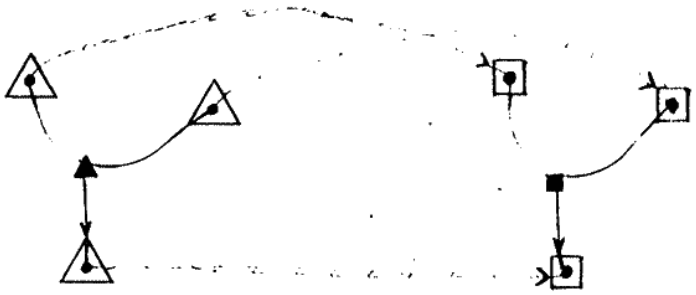


e due algebre di tale segnatura, una "triangolare" e l'altra "quadrata" :



un morfismo tra l'algebra triangolare e quella quadrata è una corrispondenza ⟿ che manda ogni triangolo nel corrispondente quadrato e che per esempio riferita allo operatore  verifica la seguente corrispondenza

di evidente lettura



(i punti interni alle figure rappresentano elementi)

13.- DEFINIZIONE

Sia Σ una segnatura di sorte $\mathcal{S}$ e $\underline{\underline{A}}$, $\underline{\underline{B}}$ due Σ-algebre, dicesi $\underline{\Sigma\text{-morfismo}}$ da $\underline{\underline{A}}$ in $\underline{\underline{B}}$

$$h: \underline{\underline{A}} \longrightarrow \underline{\underline{B}}$$

una famiglia di funzioni

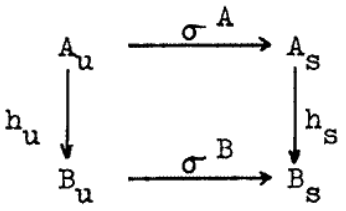
$$h = (h_s \mid s \in \mathcal{S})$$

tale che

i) $h_s : A_s \longrightarrow B_s$

ii) $h_s(\sigma^A x) = \sigma^B(h_u x) \qquad \forall\, \sigma \in \Sigma_{u,s}\ ,\ x \in A_u$

ovvero $\forall\, \sigma \in \Sigma$ h commuta il diagramma



Osserviamo che la condizione ii) impone in particolare che $h_s(\sigma^A) = \sigma^B \quad \forall\, \sigma \in \Sigma_s$

Se $\forall$ s h_s è

1)	iniettiva	h	dicesi	monomorfismo
2)	surgettiva	h	"	epimorfismo
3)	biunivoca	h	"	isomorfismo

(Omomorfismo è una variante terminologica di morfismo)

14.- ESERCIZIO

Verificare che la composizione di Σ-morfismi è un Σ-morfismo.

Sia $\underline{A}$ una Σ-algebra (di sorta $\mathcal{S}$) ed

$$R = (R_s \mid s \in \mathcal{S})$$

una famiglia di relazioni tali che R_s è una relazione di

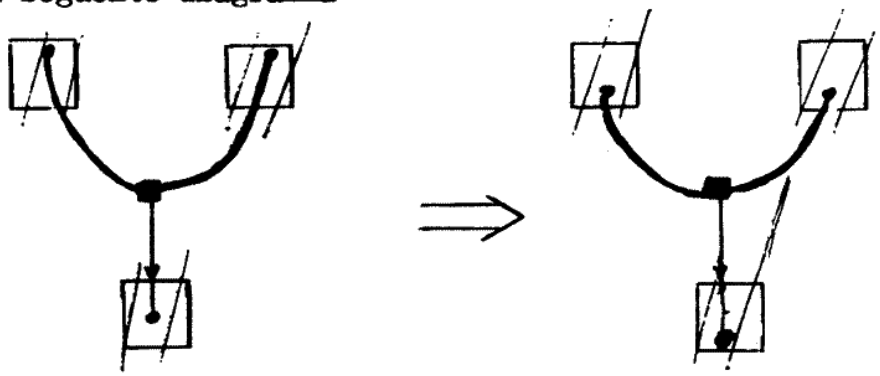
equivalenza su A_s per ogni $s \in \mathcal{S}$.

Se $x,y \in A_u$ intendiamo con

$$x \; R_u \; y$$

$$x_1 R_{s_1} y_1 \quad e \quad x_2 R_{s_2} y_2 \quad \cdots\cdots \quad x_k R_{s_k} y_k$$

Se tutte le operazioni di $\underline{\underline{A}}$ applicate ad elementi equivalenti producono elementi equivalenti, come esemplificato dal seguente diagramma



allora R dicesi una congruenza su $\underline{\underline{A}}$ come precisato dalla seguente

15.- DEFINIZIONE

$R = (R_s \mid s \in \mathcal{S})$ è una $\underline{\Sigma\text{-congruenza}}$ su A se

se $\forall \sigma \in \Sigma_{u,s}$

$$x \; R_u \; y \implies \sigma^A x \; R_s \; \sigma^A y$$

16.- ESEMPIO

Nell'algebra $\underline{\underline{L}}$ dell'esempio 1, la seguente relazione R è una congruenza

a) $\alpha \; R_{string} \; \beta \iff \alpha$ è permutazione di β

b) $x \; R_{nat} \; y \iff$ x e y dividono gli stessi numeri primi

c) $\forall p,q \; L_{bool} \iff p = q$

Dato un insieme e una relazione di equivalenza su di esso l'insieme quoziente modulo tale relazione è ottenuto considerando come elementi le classi di equivalenza della relazione.

Una congruenza permette di estendere la stessa costruzione anche per le algebre come specificato dalla seguente

17.- DEFINIZIONE

Data una Σ-congruenza R su una Σ-algebra $\underline{\underline{A}}$ dicesi algebra quoziente di $\underline{\underline{A}}$ modulo R l'algebra seguente

$$=/R = \left((A_s/R_s \mid s \in \mathcal{S}), (\sigma^{A/R} \mid \sigma \in \Sigma) \right)$$

i cui domini sono gli insiemi quozienti A_s/R_s e indicando con $[x]_R$ la classe di equivalenza di x rispetto R

$$\sigma^{A/R}([x]_R) = [\sigma^A x]_R$$

Tale definizione è ben posta poichè essendo R una congruenza su $\underline{\underline{A}}$

$$x,y \in A_u \Rightarrow x \; R_u \; y \Rightarrow \sigma^A x \; R_s \; \sigma^A y \Rightarrow [\sigma^A x]_R = [\sigma^A y]_R$$

cioè la definizione delle operazioni di $\underline{\underline{A}}/R$ non dipende dai particolari rappresentanti delle classi di equivalenza

La nozione di morfismo e congruenza sono intimamente connesse come specificato dal seguente

18.- TEOREMA (fondamentale del morfismo)

Sia $h: \underline{\underline{A}} \longrightarrow \underline{\underline{B}}$ un morfismo

i) Se $\hat{h}_s(A_s) = \{h_s x \in B_s \mid x \in A_s\}$

allora sulla famiglia

$$\hat{h}(A) = (\hat{h}_s(A_s) \mid s \in \mathcal{S})$$

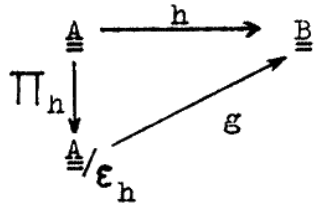
è definibile una Σ-sottoalgebra di $\underline{\underline{B}}$

ii) Se $x \; \varepsilon_h \; y \iff h_s x = h_s y \quad \forall s \in \mathcal{S} \quad x,y \in A_s$

allora ε_h è una congruenza su $\underline{\underline{A}}$

iii) Se $\Pi_h: \underline{\underline{A}} \to \underline{\underline{A}}/\varepsilon_h$ e $\Pi_h x = [x]_{\varepsilon_h}$

allora esiste un unico monomorfismo g che rende commutativo il seguente diagramma



Prova

i) Bisogna provare che la famiglia $\hat{h}(A)$ è chiusa rispetto alle operazioni di $\underline{\underline{B}}$. Osserviamo che

$$\hat{h}(A)_s = \hat{h}(A_s)$$

e che $\hat{h}(A)_u = \hat{h}_u(A_u) = \hat{h}_{s_1}(A_{s_1}) \times \ldots\ldots \hat{h}_{s_k}(A_{s_k})$

rimane quindi da provare che

$$y \in \hat{h}_u(A_u) \Rightarrow \sigma^B y \in \hat{h}_s(A_s) \quad \forall\, \sigma \in \Sigma_{u,s}\,,\ y \in B_u$$

ma $y \in \hat{h}_u(A_u) \Rightarrow y = h_u x$ e $x \in A_u$

quindi $\sigma^B y = \sigma^B(h_u x) = h_s(\sigma^A x)$

ovvero $\sigma^B y \in \hat{h}_s(A_s)$

ii) $x \;\varepsilon_h\; y \Rightarrow hx = hy \Rightarrow \sigma^B(hx) = \sigma^B(hy)$

(per def.) $\Downarrow$ (h morfismo)

$h(\sigma^A x) = h(\sigma^A y)$

$\Downarrow$ (per def.)

$\sigma^A x \;\varepsilon_h\; \sigma^A y$

Per semplicità sono stati omessi indici ad h e ad ε_h

iii) Basta definire

$$\#) \qquad g\,[x]_{\varepsilon_h} = hx$$

g è un morfismo infatti:

$$g(\sigma^{A/\varepsilon_h}\,[x]_{\varepsilon_h}) = g[\sigma^A x]_{\varepsilon_h} = h(\sigma^A x) = \sigma^B(hx) =$$

$$= \sigma^B(g\,[x]_{\varepsilon_h})$$

g è unico poichè ogni altro morfismo che commuta il diagramma deve soddisfare #)

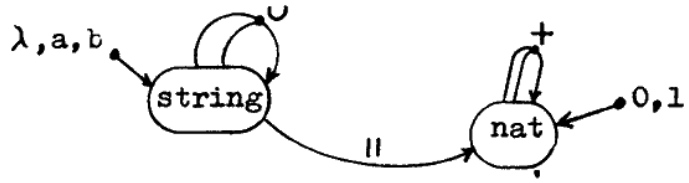
g è iniettivo poichè

$$g[x]_{\varepsilon_h} = g[y]_{\varepsilon_h} \Rightarrow hx = hy \Rightarrow x\;\varepsilon_h\;y \Rightarrow [x]_{\varepsilon_h} = [y]_{\varepsilon_h}$$

IV

TERMINI ASSEGNAMENTI VALUTAZIONI

Data una segnatura Σ quale



i termini di tale segnatura sono tutte le espressioni costruibili utilizzando gli operatori della segnatura per denotare elementi di una Σ-algebra, per esempio:

$$a,\ b,\ a \cup b,\ (a \cup b) \cup \lambda, \ldots .$$

$$0,\ |a|,\ |a| + 1, \ldots .$$

Se oltre agli elementi della segnatura si usano variabili di tante sorte quante sono quelle della segnatura, per esempio

$x, y \ldots$ di tipo nat

$\alpha, \beta \ldots$ di tipo string

abbiamo altre espressioni quali

$$\alpha,\ \alpha \cup a,\ (\alpha \cup \beta) \cup b, \ldots .$$

$$x,\ |\alpha \cup a| + x, \ldots .$$

Una definizione formale della nozione può darsi per induzione.

Richiamiamo che A^n indica l'insieme delle n-uple su A e che A^* è l'insieme di tutte le possibili n-uple con $n \in N$, cioè $A^* = \bigcup_{n \in N} A^n$ $\quad (A^0 = \lambda)$

Definiamo quindi per induzione l'insieme $L(A)$ delle liste su A

$$L_0(A) = A$$

$$L_{i+1}(A) = (L_i(A))^* \cup L_i(A)$$

$$L(A) = \bigcup_{i \in N} L_i(A)$$

per esempio se $a,b \in A$ allora

$$a\ ,\ (a,b)\ ,\ (\ (a,b),b\ ),\ (\ (a,b,b),\ ((a,b),b)\ )$$

sono elementi di $L(A)$ (spesso per abbreviare ometteremo virgole e/o parentesi esterne)

19.- DEFINIZIONE

Sia $V = (V_s \mid s \in \mathcal{S})$

una famiglia $\mathcal{S}$-indicizzata di insiemi di elementi detti variabili.

La famiglia $T_\Sigma(V)$ di <u>termini</u> è costituita da insiemi di liste su costanti di Σ e variabili di V :

$$T_\Sigma(V) = (\ (T_\Sigma(V))_s \mid s \in \mathcal{S}\ )$$

definita come segue per induzione

i) $\Sigma_s \subset (T_\Sigma(V))_s$ e $V_s \subset (T_\Sigma(V))_s$

ii) $\sigma \in \Sigma_{u,s}$, $t \in (T_\Sigma(V))_u \Rightarrow (\sigma,t) \in (T_\Sigma(V))_s$

Utilizzando tale definizione, il termine informalmente indicato con $|\alpha \cup a| + x$ è completamente tradotto, secondo la condizione ii), nel termine formale

$$(+,(\ (||,(\cup,(\alpha,a))\ ),\ x)\ )$$

V

GENERATORI, MINIMALITÀ, INIZIALITÀ

Data una Σ-algebra e dei sottoinsiemi dei suoi domini, la generazione di tali sottoinsiemi nell'algebra data è costituita da tutti gli elementi ottenibili applicando in tutti i modi possibili le operazioni dell'algebra a partire dagli elementi dei sottoinsiemi.

Tale idea può precisarsi utilizzando i termini.

21.- DEFINIZIONE

Sia $\underline{\underline{A}} \in \Sigma\text{-alg}$ e $B = (B_s \mid s \in \mathcal{S})$ una famiglia di insiemi tali che

$$B_s \subseteq A_s \qquad \forall\, s \in \mathcal{S}$$

Sia inoltre $W = (W_s \mid s \in \mathcal{S})$ una famiglia di variabili tale che

$$|B_s| = |W_s| \qquad \forall\, s \in \mathcal{S}$$

e $j = (j_s \mid s \in \mathcal{S})$ con $j_s : W_s \to B_s$, j_s biunivoca $\forall s \in \mathcal{S}$

diciamo allora <u>chiusura</u> di B in $\underline{\underline{A}}$ la famiglia

$$(B)_{\underline{\underline{A}}} = \left(\left((B)_{\underline{\underline{A}}} \right)_s \mid s \in \mathcal{S} \right)$$

ove

$$\left((B)_{\underline{\underline{A}}} \right)_s = \left\{ \coprod t \coprod_{sj} \mid t \in T_\Sigma(W) \right\}$$

(cfr. convenzione 7.3 e definizione 17)

22.- TEOREMA

i) Su $(B)_{\underline{\underline{A}}}$ è definibile una sottoalgebra di $\underline{\underline{A}}$, $(\underline{\underline{B}})_{\underline{\underline{A}}}$ è detta <u>sottoalgebra</u> di A <u>generata</u> da B.

ii) $(\underline{\underline{B}})_{\underline{\underline{A}}}$ è la minima (rispetto all'inclusione) sottoalgebra di $\underline{\underline{A}}$ che include B

<u>Prova</u>

Basta verificare che la definizione sopra data di $(B)_{\underline{\underline{A}}}$ è equivalente alla richiesta delle seguenti condizioni $\forall\, s \in \mathcal{S}$:

o) $B_s \subset ((B)_{\underline{\underline{A}}})_s$

i) $x \in ((B)_{\underline{\underline{A}}})_u \;,\; \sigma \in \Sigma_{u,s} \Rightarrow \sigma^A x \in ((B)_{\underline{\underline{A}}})_s$

23.- DEFINIZIONE

B dicesi un <u>sistema di generatori</u> per $\underline{\underline{A}}$ sse.

$$(\underline{\underline{B}})_{\underline{\underline{A}}} = \underline{\underline{A}}$$

24.- ESEMPIO

L'algebra $(N,+,0,1)$ ha $\{0,1\}$ come sistema di generatori

25.- DEFINIZIONE

Data una segnatura Σ e delle variabili V definiamo <u>l'algebra dei termini</u> $\underline{\underline{T}}_{\Sigma}(V)$ ponendo

$$\underline{\underline{T}}_{\Sigma}(V) = \left(((T_{\Sigma}(V))_s \mid s \in \mathcal{S}), (\sigma^T \mid \sigma \in \Sigma) \right)$$

ove

i) $\sigma^T(t) = (\sigma\ t)$

$\forall\ u,s \in \mathcal{S}^*, \forall \sigma \in \Sigma_{u,s}$, $\forall\ t \in (T_{\Sigma}(V))_u$

ii) $\sigma^T = \sigma$ $\qquad \forall\ s \in \mathcal{S}$, $\forall\ \sigma \in \Sigma_s$

ovvero i domini di $\underline{\underline{T}}_{\Sigma}(V)$ sono i termini su Σ e le variabili di V, mentre le operazioni di $\underline{\underline{T}}_{\Sigma}(V)$ si identificano con le operazioni di costruzione di liste con cui si ottengono i termini .

Osserviamo che le condizioni i), ii), iii) della def. 20, equivalgono ad affermare che g: $g(t) = [\![t]\!]_{\rho}$ è un morfismo da $\underline{\underline{T}}_{\Sigma}(V)$ in $\underline{\underline{A}}$ che estende ρ su tutti i termini. Inoltre g è unico poichè ogni altra g che verifica i), ii), iii), come si verifica semplicemente per induzione, coincide con g.

26.- DEFINIZIONE

Una algebra dicesi <u>minimale</u> se non contiene sottoalgebre proprie (diverse da se stessa) ovvero per il teorema precedente se é generata dalle costanti.

27.- COROLLARIO

Indichiamo con $\underline{\underline{T}}_{\Sigma}$ l'algebra di termini su Σ e su una famiglia vuota di variabili i.e. $V_s = \emptyset \ \forall \ s$, allora $\underline{\underline{T}}_{\Sigma}$ è minimale

Prova

E' evidente che $\underline{\underline{T}}_{\Sigma}$ è generata dalle costanti, quindi è minimale.

28.- OSSERVAZIONE

Se $\Sigma_s = \emptyset \quad \forall \ s \in \mathcal{S}$ allora $\underline{\underline{T}}_{\Sigma}$ ha tutti i domini vuoti. Poichè ammettere alcuni domini vuoti porterebbe noiose eccezioni nei teoremi che seguono noi supporremo che le segnature considerate d'ora innanzi siano tali che $\forall \ s \in \mathcal{S} \quad (T_{\Sigma})_s \neq \emptyset$

29.- TEOREMA

Se $\underline{\underline{A}} \in \Sigma\text{-alg}$ allora $\underline{\underline{A}}$ è <u>immagine epimorfica</u> di $\underline{\underline{T}}_{\Sigma}(V)$ per un opportuno sistema V di variabili; ovvero per il teorema fondamentale del morfismo

$$\underline{\underline{A}} \cong T_{\Sigma}(V)/_{\theta}$$

per una opportuna congruenza θ ($\cong$ indica isomorfismo)

Prova

Sia $B = (B_s | s \in \mathcal{S})$ un sistema di generatori per $\underline{\underline{A}}$ tale sistema esiste sempre, al limite $B = A$, si ha quindi che

$$\underline{\underline{A}} = (\underline{\underline{B}})_{\underline{\underline{A}}}$$

Consideriamo quindi una famiglia di variabili $V = (V_s | s \in \mathcal{S})$ tale che

$$|V_s| = |B_s| \qquad \forall\, s \in \mathcal{S}$$

Sia j un assegnamento biunivoco tra V_s e B_s

$$g_s(t) = [\![t]\!]_{sj} \qquad \forall\, s \in \mathcal{S} \quad , \; t \in (T_{\Sigma}(V))_s$$

in base alla definizione di valutazione $[\![\ ]\!]$ si verifica subito che

$$g = (g_s | s \in \mathcal{S})$$

è un morfismo da $\underline{\underline{T}}_{\Sigma}(V)$ in $\underline{\underline{A}}$, inoltre per definizione di chiusura g è anche surgettivo su $(B)_{\underline{\underline{A}}}$ e quindi su $\underline{\underline{A}}$, allora la congruenza θ dell'enunciato, per il teorema del morfismo è data dalla condizione:

$$t \,\theta\, t' \iff [\![t]\!]_j = [\![t']\!]_j$$

30.- DEFINIZIONE

Sia $\mathcal{C} \subseteq \Sigma\text{-alg}$, una algebra $\underline{\underline{I}}$ dicesi <u>iniziale</u> in $\mathcal{C}$ sse. $\underline{\underline{I}} \in \mathcal{C}$ e $\forall\, \underline{\underline{A}} \in \mathcal{C}$ esiste un unico Σ-morfismo da $\underline{\underline{I}}$ in $\underline{\underline{A}}$.

31.- TEOREMA

Se $\underline{\underline{I}}$ è iniziale in $\mathcal{C}$ allora $\underline{\underline{I}}$ è unica a meno di isomorfismi.

Prova

Supponiamo vi sia un'algebra $\underline{\underline{I}}'$ iniziale per $\mathcal{C}$ allora per la inizialità di I avremmo un unico morfismo

$$f : \underline{\underline{I}} \longrightarrow \underline{\underline{I}}'$$

e per quella di $\underline{\underline{I}}'$ un unico morfismo

$$g : \underline{\underline{I}}' \longrightarrow I$$

ma $g \circ f$ essendo composizione di morfismi è ancora un morfismo, allora $g \circ f : \underline{\underline{I}} \longrightarrow \underline{\underline{I}}$ deve coincidere con l'identità su I , essendo l'identità un morfismo ed essendoci per inizialità un tale morfismo tra $\underline{\underline{I}}$ e se stessa, quindi

$$g \circ f = i_A$$

cioè g ed f sono biunivoco, ovvero isomorfismi cioè

$\underline{\underline{I}}$ ed $\underline{\underline{I}}'$ sono algebre isomorfe.

32.- DEFINIZIONE

Se $\mathcal{C} \subseteq \Sigma$-alg , una algebra $\underline{\underline{F}} \in \mathcal{C}$ dicesi finale in $\mathcal{C}$ sse

$\forall\, \underline{\underline{A}} \in \mathcal{C}$ esiste un unico morfismo $f: \underline{\underline{A}} \longrightarrow \underline{\underline{F}}$

In modo analogo al teorema 31 si dimostra che un'algebra finale è unica a meno di isomorfismi

33.- TEOREMA

$\underline{\underline{T}}_{\Sigma}$ è iniziale in Σ-alg

Prova

Sia $\underline{\underline{A}} \in \Sigma$-alg , allora ogni morfismo da $\underline{\underline{T}}_{\Sigma}$ in $\underline{\underline{A}}$ deve verificare le condizioni

$$f(\sigma^{T}) = \sigma^{A} \qquad \forall\, \sigma \in \Sigma_{s}$$

$$f(\sigma^{T}x) = \sigma^{A}(fx) \qquad \forall\, \sigma \in \Sigma_{u,s} \ , \ x \in (T_{\Sigma})_{u}$$

ma tali condizioni definiscono per induzione una funzione da T_{Σ} in A , ovvero vi è un unico morfismo da $\underline{\underline{T}}_{\Sigma}$ in $\underline{\underline{A}}$.

34.- TEOREMA

Se $\underline{\underline{A}}$ è minimale allora $\underline{\underline{A}} \cong \underline{\underline{T}}_{\Sigma}/_{\theta}$ per una opportuna congruenza θ

Prova

Infatti dato un morfismo f da $\underline{\underline{T}}_{\Sigma}$ in $\underline{\underline{A}}$ (che esiste ed è unico per la inizialità di $\underline{\underline{T}}_{\Sigma}$ in Σ-alg) per la minimalità di $\underline{\underline{A}}$ f deve essere surgettivo (altrimenti $\underline{\underline{A}}$ avrebbe sottoalgebre proprie) quindi per il teorema del morfismo

$$\underline{\underline{A}} \cong \underline{\underline{T}}_{\Sigma}/_{\varepsilon_{f}}$$

Per denotare tramite un termine un elemento di una Σ-algebra bisogna dare valore alle variabili e interpretare gli operatori del termine con operazioni dell'algebra.

Poichè i termini sono formalmente definiti per induzione, possiamo definire formalmente per induzione la valutazione di termini.

20.- DEFINIZIONE

Sia A una famiglia $\mathcal{S}$-indicizzata di domini di un'algebra, un <u>assegnamento</u> su A è una famiglia ρ di funzioni

$$\rho = (\rho_s \mid s \in \mathcal{S})$$

tali che $\rho_s : V_s \longrightarrow A_s \qquad \forall\, s \in \mathcal{S}$

Indichiamo con $Ass(V,A)$ l'insieme dagli assegnamenti di V in A.

Diciamo <u>valutazione</u> di $T_\Sigma(V)$ in $\underline{A} \in \Sigma\text{-alg}$ la famiglia di funzioni

$$[\![\ ]\!]_s : (T_\Sigma(V))_s \times Ass(V,A) \longrightarrow A_s \qquad \forall\, s \in \mathcal{S}$$

definita per induzione :

i) $[\![v]\!]_{s\rho} = \rho v \qquad \forall\, v \in V_s$

ii) $[\![\sigma]\!]_{s\rho} = \sigma^A \qquad \forall\, \sigma \in \Sigma_s$

iii) $[\![\sigma\, t]\!]_{s\rho} = \sigma^A([\![t]\!]_{u\rho}) \qquad \forall\, \sigma \in \Sigma_{u,s}$

35.- OSSERVAZIONE

L'inizialità è uno strumento essenziale per la definizione di concetti algebrici.

Infatti un requisito fondamentale per essi è quello di essere caratterizzati a meno di isomorfismi.

Non appena ci si assicura che essiste ed è iniziale l'algebra verificante certe condizioni, allora si può assumere che dette condizioni costituiscono una buona definizione di algebra.

L'esempio più immediato al riguardo è l'algebra T_Σ che è univocamente individuata dalla segnatura Σ in quanto algebra iniziale di Σ-alg .

VI

TEORIE ALGEBRICHE

36.- DEFINIZIONE

Diciamo $\sum^V$-equazione una coppia di termini $t_1, t_2 \in T_\sum(V)$ che scriveremo

$$t_1 = t_2$$

$\sum^V$-eq individua l'insieme di tutte le $\sum^V$-equazioni.

37.- DEFINIZIONE

Diciamo che l'equazione $t_1 = t_2$ vale in $\underline{\underline{A}}$ e scriveremo

$$\underline{\underline{A}} \models t_1 = t_2 \qquad (\underline{\underline{A}} \text{ soddisfa } t_1 = t_2)$$

se $\forall\ \rho \in Ass(V,A) \quad [\![t_1]\!]_\rho = [\![t_2]\!]_\rho$

ovvero se t_1 e t_2 denotano lo stesso elemento di A per ogni valutazione delle variabili

38.- DEFINIZIONE

Dati $\mathcal{E} \subseteq \sum^V\text{-eq}$; $\mathcal{C} \subseteq \sum\text{-alg}$; $\underline{\underline{A}} \in \sum\text{ alg}$

poniamo :

i) $\underline{\underline{A}} \models \mathcal{E} \iff \underline{\underline{A}} \models e \quad \forall\ e \in \mathcal{E}$

ii) $\mathcal{C} \models e \iff \underline{\underline{A}} \models e \quad \forall\ \underline{\underline{A}} \in \mathcal{C}$

iii) $\mathcal{C} \models \mathcal{E} \iff \mathcal{C} \models e \quad \forall\ e \in \mathcal{E} \iff \underline{\underline{A}} \models \mathcal{E} \quad \forall \underline{\underline{A}} \in \mathcal{C}$

39.- DEFINIZIONE

Dati $\mathcal{E} \subseteq \Sigma^{V}\text{-eq}$; $\mathcal{C} \subseteq \Sigma\text{- alg}$ poniamo

i) $\Sigma\text{-alg}(\mathcal{E}) = \{\underline{\underline{A}} \in \Sigma alg \mid \underline{\underline{A}} \models \mathcal{E}\}$

$\Sigma\text{-alg}(\mathcal{E})$ è detta <u>varietà</u> di equazioni $\mathcal{E}$

ii) $\Sigma^{V}\text{-eq}(\mathcal{C}) = \{e \in \Sigma^{V}eq \mid \mathcal{C} \models e\}$

iii) $Th(\mathcal{E}) = \Sigma^{V}\text{-eq}(\Sigma\text{-alg}(\mathcal{E}))$

$Th(\mathcal{E})$ è detta <u>teoria equazionale</u> di presentazione $(\Sigma, \mathcal{E})$

40.- DEFINIZIONE

Diciamo <u>equazioni condizionali</u> una $n+1$-upla di equazioni che scriveremo nella forma

$$e_1 \quad e_2 \quad \ldots\ldots \quad e_n \longrightarrow e_{n+1}$$

Diciamo che tale equazione condizionale vale in $\underline{\underline{A}} \in$ alg quando

$$\underline{\underline{A}} \models e_1 \quad e \quad \underline{\underline{A}} \models e_2 \quad \ldots\ldots \quad \underline{\underline{A}} \models e_n \Rightarrow \underline{\underline{A}} \models e_{n+1}$$

In modo del tutto naturale si può parlare di teorie di equazioni condizionali generalizzando in modo ovvio le definizioni di sopra. Diciamo <u>quasi-varietà</u> le classi di algebre verificanti teorie di equazioni condizionali. Gli assiomi di una teoria algebrica in senso lato possono avere forme logiche più generali di quelle viste, ciò che è peculiare è l'uso dell'uguaglianza, variabili e operatori come unici strumenti espressivi oltre a connettivi e quantificatori.

VII

VARIETÀ

41.- DEFINIZIONE (Calcolo equazionale)

Sia $\mathfrak{E} \subseteq \Sigma^{V}\text{-eq}$ diciamo <u>chiusura deduttiva</u> (equazionale) di $\mathfrak{E}$ il seguente insieme

$$\mathfrak{E}^* \subseteq \Sigma^{V}\text{-eq}$$

definito induttivamente

o) $\mathfrak{E} \subset \mathfrak{E}^*$, $v=ve \mathfrak{E}^* \quad \forall\, ve V$

1) $t_1 = t_2 \in \mathfrak{E}^* \implies t_2 = t_1 \in \mathfrak{E}^*$

2) $t_1 = t_2 \ , \ t_2 = t_3 \in \mathfrak{E}^* \implies t_1 = t_3 \in \mathfrak{E}^*$

3) $$\begin{cases} (t_1 \dots t_k), (t_1' \dots t_k') \in (T_\Sigma(V))_u \ , \ \sigma \in \Sigma_{u,s}, \ t_1 = t_1' \dots t_k = t_k' \in \mathfrak{E} \\ \Downarrow \\ \sigma(t_1 \dots t_k) = \sigma(t_1' \dots t_k') \in \mathfrak{E}^* \end{cases}$$

4) $$\begin{cases} \tau \in \mathrm{Ass}(V, T_\Sigma(V)) \ , \ t_1 = t_2 \in \mathfrak{E}^* \\ [\![t_1]\!]_\tau = [\![t_2]\!]_\tau \in \mathfrak{E}^* \end{cases}$$

La definizione precedente può essere data in modo equivalente ponendo una relazione $\vdash$ di deducibilità tale che

$$e \in \mathfrak{E}^* \iff \mathfrak{E} \vdash e$$

la cui descrizione in forma $\frac{\text{premesse}}{\text{conclusioni}}$ è la seguente

0) $\mathcal{E} \vdash e \quad \forall\, e \in \mathcal{E}$. $\forall c \in \mathcal{E}$, $\mathcal{E} \vdash v = v \quad \forall\, v \in V$

1) $$\frac{\mathcal{E} \vdash t_1 = t_2}{\mathcal{E} \vdash t_2 = t_1}$$ (simmetria)

2) $$\frac{\mathcal{E} \vdash t_1 = t_2 \qquad \mathcal{E} \vdash t_2 = t_3}{\mathcal{E} \vdash t_1 = t_3}$$ (transitività)

3) $$\frac{\mathcal{E} \vdash t = t'}{\mathcal{E} \vdash \sigma t = \sigma t'} \qquad \forall\, t \in (T_\Sigma)_u\,,\ \sigma \in \Sigma_{u,s}$$ (congruità)

4) $$\frac{\mathcal{E} \vdash t = t'}{\mathcal{E} \vdash [\![t]\!]_\tau = [\![t']\!]_\tau} \qquad \forall\, \tau \in \mathrm{Ass}\,(V, T_\Sigma(V))$$ (sostitutività)

42.- TEOREMA

$\mathcal{E}^*$ è la minima congruenza che include $\mathcal{E}$ e chiusa per endomorfismi di $\underline{\underline{T}}_\Sigma(V)$. (un endomorfismo di $\underline{\underline{A}} \in \Sigma\text{-alg}$ è un morfismo di $\underline{\underline{A}}$ in $\underline{\underline{A}}$ e una congruenza θ è chiusa per endomorfismo f se $x\,\theta\,y \iff fx\,\theta\,fy$) ($R \trianglelefteq T \iff (xRy \implies xTy)$)

Prova

Per definizione $\mathcal{E}^*$ è una congruenza (cfr. def. 41) ed è inoltre chiusa per endomorfismi di $\underline{\underline{T}}_\Sigma(V)$ per la regola 4) della def. 41 essendo un endomorfismo di $\underline{\underline{T}}_\Sigma(V)$ completamente individuato da un $\tau \in \mathrm{Ass}\,(V, T_\Sigma(V))$ cfr. teor. 33).

Inoltre ogni congruenza chiusa per endomorfismi deve verificare 0),1),2),3),4) della def. 41) quindi $\mathcal{E}^*$ è la minima tra tutte le congruenze siffatte.

43.- TEOREMA

Se $\underline{\underline{A}}$ è minimale allora $\underline{\underline{A}}$ è iniziale in Σ-alg(Σ^V-eq(A))

(e quindi in ogni $\mathcal{C} \subseteq \Sigma$-alg(Σ^V-eq(A))

Prova

Se $\underline{\underline{A}}$ è minimale $\underline{\underline{A}}$ è generata dalle costanti quindi (omettiamo gli indici)

$$*) \qquad A = \left\{ [\![t]\!]_{\rho_o} \mid t \in T_\Sigma \right\}$$

con ρ_o prefissato , $\rho_o \in Ass(V,A)$

Sia Ψ_o prefissato , $\Psi_o \in Ass(V,B)$ con

$$\underline{\underline{B}} \in \Sigma\text{-alg}(\Sigma^V\text{-eq}(\underline{\underline{A}}))$$

e poniamo $g([\![t]\!]_{\rho_o}) = [\![t]\!]_{\Psi_o} \quad \forall\, t \in T_\Sigma$

g verifica la condizione seguente $\forall\ t_1, t_2 \in T_\Sigma$

$$**) \quad [\![t_1]\!]_{\rho_o} = [\![t_2]\!]_{\rho_o} \Longrightarrow g([\![t_1]\!]_{\rho_o}) = g([\![t_2]\!]_{\rho_o})$$

ovvero $[\![t_1]\!]_{\rho_o} = [\![t_2]\!]_{\rho_o} \Longrightarrow [\![t_1]\!]_{\Psi_o} = [\![t_2]\!]_{\Psi_o}$

Infatti poichè t_1, t_2 non hanno variabili

$$[\![t_1]\!]_{\rho_o} = [\![t_2]\!]_{\rho_o} \Longrightarrow [\![t_1]\!]_{\rho} = [\![t_2]\!]_{\rho} \quad \forall\, \rho \in Ass(V,A)$$

quindi $\underline{\underline{A}} \models t_1 = t_2$

cioè essendo $\underline{\underline{B}} \in \Sigma$-alg(Σ^V-eq($\underline{\underline{A}}$)) $\quad \underline{\underline{B}} \models t_1 = t_2$

quindi $[\![t_1]\!]_{\Psi_o} = [\![t_2]\!]_{\Psi_o}$

vale in definitiva $**$)

Da *) e **) segue allora che g è una funzione ben definita da $\underline{\underline{A}}$ in $\underline{\underline{B}}$. Si verifica facilmente che g è un morfismo.

Inoltre g è unico perchè se vi fossero due morfismi g, g' da $\underline{\underline{A}}$ in $\underline{\underline{B}}$ essendo $\underline{\underline{A}}$ minimale $\underline{\underline{A}} \cong T_{\Sigma/\theta}$ per una θ opportuna allora $\lambda t.g([t]_\theta)$ e $\lambda t.g'([t]_\theta)$ sarebbero due morfismi da $\underline{\underline{T}}_\Sigma$ in $\underline{\underline{B}}$ contro l'inizialità di $\underline{\underline{T}}_\Sigma$ in Σ-alg.

44.- TEOREMA

Una varietà ammette sempre algebra iniziale (ed essa è anche minimale).

Prova

Siano $\mathcal{E}$ le equazioni che definiscono la data varietà e sia $\underline{\underline{I}} = \underline{\underline{T}}_\Sigma / \overline{\mathcal{E}^*}$ ove $\forall\, t_1, t_2 \in T_\Sigma$ si ponga

$$t_1 \,\overline{\mathcal{E}^*}\, t_2 \iff \mathcal{E} \vdash t_1 = t_2 \quad \text{(cfr. def. 41)}$$

i) Mostriamo prima che $\underline{\underline{I}} \in \Sigma\text{-alg}(\mathcal{E})$, cioè

$$t_1 = t_2 \in \mathcal{E} \implies T_\Sigma/\overline{\mathcal{E}^*} \models t_1 = t_2$$

Infatti per definizione di $\overline{\mathcal{E}^*}$ si ha

$$[\![t_1]\!]_\tau = [\![t_2]\!]_\tau \in \mathcal{E}^* \qquad \forall\, \tau \in \mathrm{Ass}(V, T_\Sigma)$$

cioè $\left[\, [\![t_1]\!]_\tau \,\right]_{\overline{\mathcal{E}^*}} = \left[\, [\![t_2]\!]_\tau \,\right]_{\overline{\mathcal{E}^*}}$

ma questo implica (per induzione sui termini)

$$[\![t_1]\!]_{\sigma'} = [\![t_2]\!]_{\sigma'} \qquad \forall\ \sigma' \in Ass(V, T_{\Sigma/\overline{\xi^*}})$$

Ovvero $\quad \underline{\underline{T}}_{\Sigma/\overline{\xi^*}} \models t_1 = t_2$

ii) Mostriamo ora che esiste un morfismo $g: \underline{\underline{I}} \to \underline{\underline{A}}$ $\forall\ A \in \Sigma\text{-alg}(\xi)$.

Poniamo $g\,[t]_{\xi^*} = [\![t]\!]_{\rho_o}$ ove ρ_o è un qualsiasi, ma fissato $\rho_o \in Ass(V,A)$, g è ben definita poichè

$$[t]_\xi = [t']_\xi \Rightarrow \xi \vdash t = t' \Rightarrow \underline{\underline{A}} \models t = t' \Rightarrow$$

$$\Rightarrow [\![t]\!]_{\rho_o} = [\![t']\!]_{\rho_o} \qquad \text{(cfr. più avanti Lemma 46)}$$

E' semplice verificare che g è un morfismo infatti:

$$g\sigma^T = g[\sigma]_{\xi^*} = [\![\sigma]\!]_{\rho_o} = \sigma^A$$

$$g(\sigma^T(t)) = g[\sigma\, t]_{\xi^*} = [\![\sigma\, t]\!]_{\rho_o} = \sigma^A [\![t]\!]_{\rho_o} = \sigma^A(gt)$$

iii) Mostriamo infine che da $\underline{\underline{I}}$ vi è al più un unico morfismo in ogni altra algebra $\underline{\underline{A}}$ di $\Sigma\text{-alg}(\xi)$

Infatti $\underline{\underline{T}}_\Sigma$ è iniziale in Σ-alg, quindi data $\underline{\underline{A}} \in \Sigma\text{-alg}(\xi)$ esiste un unico morfismo f da $\underline{\underline{T}}_\Sigma$ in $\underline{\underline{A}}$, allora non possono esistere due morfismi diversi h e h' da $\underline{\underline{I}}$ in $\underline{\underline{A}}$ poichè altrimenti vi sarebbero due diversi morfismi $h \circ \Pi$ e $h' \circ \Pi$ da $\underline{\underline{T}}_\Sigma$ in $\underline{\underline{A}}$ contro l'unicità di f .

Diamo ora un teorema che stabilisce la completezza del calcolo $\vdash$. Premettiamo alcuni lemmi.

Siano : $\mathcal{E} \subseteq \Sigma^{V}\text{-eq}$, $e \in \Sigma^{V}\text{-eq}$

45.- LEMMA

$$\underline{\underline{T}}_{\Sigma}(V)/_{\mathcal{E}^*} \models t_1 = t_2 \implies \mathcal{E} \vdash t_1 = t_2$$

Prova

$$\underline{\underline{T}}_{\Sigma}(V)/_{\mathcal{E}^*} \models t_1 = t_2$$

$$\Downarrow$$

$$[\![t_1]\!]_{\tau'} = [\![t_2]\!]_{\tau'} \qquad \forall \quad \tau' \in Ass(V, T_{\Sigma}(V)/_{\mathcal{E}^*})$$

$$\Downarrow$$

$$\left[[\![t_1]\!]_{\tau}\right]_{\mathcal{E}^*} = \left[[\![t_2]\!]_{\tau}\right]_{\mathcal{E}^*} \quad \forall\, \tau \in Ass(V, T_{\Sigma}(V))$$

$$\Downarrow$$

$$\left[[\![t_1]\!]_{j}\right]_{\mathcal{E}} = \left[[\![t_2]\!]_{j}\right]_{\mathcal{E}^*} \text{ ove } jv = v \quad \forall\, v \in V$$

$$\Downarrow$$

$$[t_1]_{\mathcal{E}^*} = [t_2]_{\mathcal{E}^*} \implies \mathcal{E} \vdash t_1 = t_2$$

46.- LEMMA

Sia $A \in \Sigma\text{-alg}$ allora :

$$\underline{\underline{A}} \models \mathcal{E} \quad , \quad \mathcal{E} \vdash e \implies \underline{\underline{A}} \models e$$

Prova

Segue facilmente dalla definizione di $\vdash$

47.- LEMMA

$$\mathcal{E} \vdash t_1 = t_2 \implies T_\Sigma(V)/_{\mathcal{E}^*} \models t_1 = t_2$$

Prova

$$\mathcal{E} \vdash t_1 = t_2$$

$$\Downarrow$$

$$\mathcal{E} \vdash [\![t_1]\!]_\tau = [\![t_2]\!]_\tau \qquad \forall\, \tau \in Ass(V, T_\Sigma(V))$$

$$\Downarrow$$

$$\left[[\![t_1]\!]_\tau \right]_{\mathcal{E}^*} = \left[[\![t_2]\!]_\tau \right]_{\mathcal{E}^*}$$

$$\Downarrow$$

$$[\![t_1]\!]_{\tau'} = [\![t_2]\!]_{\tau'} \qquad \forall \quad \tau' \in Ass(V, T_\Sigma(V)/_{\mathcal{E}^*})$$

$$\Downarrow$$

$$T_\Sigma(V)/_{\mathcal{E}^*} \models t_1 = t_2$$

48.- LEMMA

$$T_\Sigma(V)/_{\mathcal{E}^*} \models t_1 = t_2 \iff \Sigma\text{-alg}(\mathcal{E}) \models t_1 = t_2$$

Prova

$\Longleftarrow$ segue dal lemma precedente in base al quale

$$T_\Sigma(V)/_{\mathcal{E}^*} \in \Sigma\text{-alg}(\mathcal{E})$$

$\Longrightarrow$ segue dal fatto che per il lemma 45 se

$$T_\Sigma(V)/_{\mathcal{E}^*} \models t_1 = t_2 \quad \text{allora} \quad \mathcal{E} \vdash t_1 = t_2$$

e poichè per ogni $\underline{\underline{A}} \in \Sigma\text{-alg}(\mathcal{E})$ si ha $\underline{\underline{A}} \models \mathcal{E}$ dal lemma 46 si ottiene

$$A \models t_1 = t_2$$

49.- TEOREMA (BIRKHOFF / di completezza)

$$\Sigma\text{-alg}(\mathfrak{E}) \models e \iff \mathfrak{E} \vdash e$$

Prova

$\Longrightarrow$ dal 4º e 1º Lemma

$\Longleftarrow$ dal 3º e 4º Lemma

L'algebra $T_\Sigma(V)/_{\mathfrak{E}^*}$ dicesi algebra libera di generatori V della varietà Σ-alg($\mathfrak{E}$) nel senso della seguente definizione

50.- DEFINIZIONE

Una Σ-algebra $\underline{\underline{L}}$ dicesi <u>libera di generatori</u> $G = (G_s \mid s \in \mathcal{S})$ nella classe $\mathcal{C} \subseteq \Sigma$-alg sse $\underline{\underline{L}} \in \mathcal{C}$ e $\forall\ \underline{\underline{A}} \in \mathcal{C}$ data una famiglia $f = (f_s \mid s \in \mathcal{S})$ tale che

$$f_s : G_s \longrightarrow A_s$$

esiste un unico morfismo $\bar{f} : \underline{\underline{L}} \longrightarrow \underline{\underline{A}}$ che estende f.
Tale definizione è una generalizzazione della proprietà prima stabilita per $\underline{\underline{T}}_\Sigma(V)$ cfr. Def. 25.

51.- ESERCIZIO

Siano $\underline{\underline{L}}$ ed $\underline{\underline{L}}'$ due algebre libere per $\mathcal{C}$ di geberatori G e G' ove $|G_s| = |G'_s|$ $\forall\ s \in \mathcal{S}$ allora $\underline{\underline{L}}$ ed L' sono isomorfe

52.- ESERCIZIO

Verificare che $T_\Sigma(V)/_{\mathfrak{E}^*}$ soddisfa la definizione precedente

53.- OSSERVAZIONE

$T_{\Sigma}(V)/_{\mathcal{E}^*}$ è un'algebra generica nella classe Σ-alg($\mathcal{E}$) nel senso specificato dal 4º Lemma prima stabilito; cioè in essa valgono tutte le equazioni di Σ^V-eq(Σ-alg($\mathcal{E}$)).

$T_{\Sigma}(V)/_{\mathcal{E}^*}$ è però intrinsecamente generica nel senso che le equazioni della varietà sono tutte e sole quelle valide sulla famiglia dei suoi generatori:

$$t_1 = t_2 \iff [\![t_1]\!]_{j'} = [\![t_2]\!]_{j'} \quad \text{con } j'v = [v]_{\mathcal{E}^*} \ \forall v \in V$$

Un verso dell'equivalenza è evidente, per l'altro si osservi che in base al teorema di completezza

$$\mathcal{E} \models t_1 = t_2 \iff \mathcal{E} \vdash t_1 = t_2$$

ma $\mathcal{E} \vdash t_1 = t_2 \iff [t_1]_{\mathcal{E}^*} = [t_2]_{\mathcal{E}^*}$

$$\Updownarrow$$

$$\left[[\![t_1]\!]_j \right]_{\mathcal{E}^*} = \left[[\![t_2]\!]_j \right]_{\mathcal{E}^*}$$

$$\Updownarrow$$

$$[\![t_1]\!]_{j'} = [\![t_2]\!]_{j'}$$

ove $jv = v \quad \forall v \in V$.

54.- ESERCIZIO

Verificare che dalla def. di algebra libera segue che

$$\underline{\underline{L}} \text{ libera} \iff \theta_{\gamma} = \theta_{\varepsilon}$$

($\underline{\underline{L}}$ è un'algebra con G famiglia di generatori e θ_{γ}, $\theta_{\mathcal{C}}$ congruenze su $\underline{\underline{T}}_{\Sigma}(V)$ definite come segue:

$$t_1 \,\theta_{\gamma}\, t_2 \iff [\![t_1]\!]_{\gamma} = [\![t_2]\!]_{\gamma} \quad , \gamma \in Ass(V,G)$$

$$t_1 \,\theta_{\mathcal{C}}\, t_2 \iff [\![t_1]\!]_{\rho} = [\![t_2]\!]_{\rho} \quad \forall\, \underline{\underline{A}} \in \mathcal{C} \;,\; \rho \in Ass(V,A) \quad) \,.$$

SINTASSI ASTRATTA

54.-Descrizione algebrica di grammatiche BNF

Consideriamo il semplice linguaggio programmativo descritto dalla seguente grammatica in BNF

<Comando> =:: (<Identificatore> ← <Cifra>) |
(If <Booleano> then(<Comando>)else(<Comando>))

<Booleano>=:: (<Identificatore> = <Identificatore>) |
True | False

<Identificatore> =:: I | J

<Cifra> =:: Ø | 1 | .... | 9

Definiamo l'algebra $\underline{L}$

$\underline{L} = (L_{cmd}, L_{bool}, L_{id}, L_{cfr}, True, False, I, J, Ø, \ldots, 9, assign, equal, cond)$

$L_{id} = \{I, J\}$

$L_{cfr} = \{Ø, 1, \ldots, 9\}$

$L_{bool} = \{True, False\} \cup \{H = K \mid H, K \in L_{id}\}$

$L^0_{cmd} = \{(H \leftarrow n) \mid H \in L_{id},\ n \in L_{cfr}\}$

$L^{i+1}_{cmd} = \{(If\ B\ then(C_1)else(C_2)) \mid B \in L_{bool},\ C_1, C_2 \in L^i_{cmd}\}$

$L_{cmd} = \bigcup_{i \in N} L^i_{cmd}$

$assign : L_{id} \times L_{cfr} \rightarrow L_{cmd}$

$equal : L_{id} \times L_{id} \rightarrow L_{bool}$

$cond : L_{bool} \times L_{cmd} \times L_{cmd} \rightarrow L_{cmd}$

ove

$assign(H, n) = (H \leftarrow n)\ \forall\ H \in L_{id},\ n \in L_{cfr}$

$equal(H, K) = (H = K)\ \forall\ H, K \in L_{id}$

$$cond(B,C_1,C_2) = (\underline{If}\ B\ \underline{then}(C_1)\underline{else}(C_2))$$

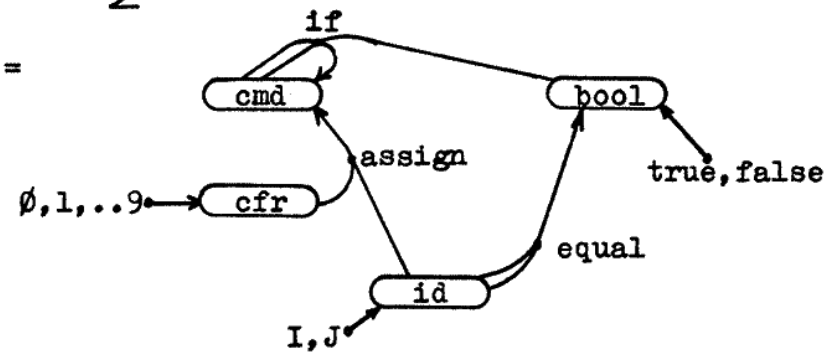
$$\forall\ B \in L_{bool},\ C_1, C_2 \in L_{cmd}$$

È evidente che $\underline{\underline{L}}$ rappresenta completamente la grammatica sopra data , infatti per ogni simbolo di categoria sintattica , in $\underline{\underline{L}}$ vi è l'insieme dei costrutti del linguaggio relativi alla categoria sintattica e ogni produzione è rappresentata da una operazione dell'algebra.

Se vogliamo esprimere la peculiarità della sintassi di tale linguaggio a prescindere da tutte le possibili varianti concrete (per esempio l'uso di parole chiave diverse per l'If o per l'uguaglianza) possiamo utilizzare in modo naturale la rappresentazione algebrica ottenuta affermando che la sintassi astratta di L è la classe di tutte le algebra isomorfe di $\underline{\underline{L}}$ che in quanto tali identificano varianti concrete della stessa sintassi astratta. Tale procedimento può applicarsi in modo ovvio a qualsiasi grammatica in BNF.

Inoltre poichè una segnatura Σ individua univocamente l'algebra $\underline{\underline{T}}_{\Sigma}$ una sintassi astratta può essere rappresentata tramite una segnatura. Se consideriamo per esempio il linguaggio di sopra è facile verificare che $\underline{\underline{L}}$ è isomorfa a $\underline{\underline{T}}_{\Sigma}$ ove

$\Sigma =$



Analogamente qualsiasi grammatica BNF (non ambigua) può essere identificata con una segnatura e in tal senso essere vista come sintassi astratta.

In generale per determinare una sintassi astratta di un qualsiasi linguaggio formale basta descriverne una sintassi come **algebra**.

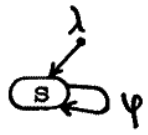
55.- Esempi di sintassi algebrica di linguaggi formali

Consideriamo per esempio una sintassi astratta per

$$L_1 = \{a^m b^n \mid n \in N\}$$

$$\underline{\underline{L}}_1 = (L_1, \varphi_1, \lambda) \quad \text{ove} \quad \begin{cases} \varphi_1 : L_1 \to L_1 \\ \varphi_1 x = axb \quad \forall x \in L_1 \end{cases}$$

È evidente che $\underline{\underline{L}}_1$ è isomorfo a $\underline{\underline{T}}_{\Sigma_1}$ ove

$\Sigma_1 =$ 

basta definire per induzione h $\begin{cases} h(\lambda) = \lambda \\ h(axb) = h(\varphi_1 x) = \varphi x \end{cases}$

per avere un isomorfismo da L_1 in T_{Σ_1}

$\underline{\underline{T}}_{\Sigma_1}$ individua anche la sintassi astratta del lin= guaggio $L_2 = \{a^n b^n c^n \mid n \in N\}$ basta a tal fine definire

$$\underline{\underline{L}}_2 = (L_2, \varphi_2, \lambda) \quad \text{ove} \quad \varphi_2(a^n b^n c^n) = a^{n+1} b^{n+1} c^{n+1} \ \forall n \in N$$

Come sopra si verifica facilmente l'isomorfismo tra $\underline{\underline{L}}_2$ e T_{Σ_1}

Ricordiamo che L_1 è un linguaggio context- free mentre L_2 è contex-sensitive, questo ci fa notare che la sintassi astratta di un linguaggio esprime proprietà strutturali più profonde della classificazione secondo Chomsky(cfr. [21]).

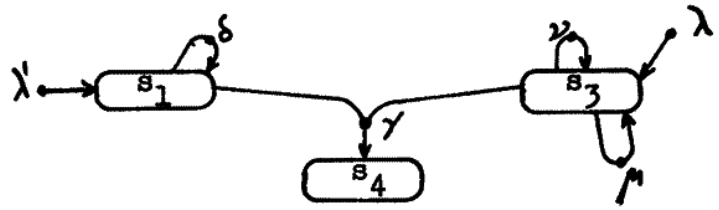
La sintassi astratta è in sostanza una sorta di ponte tra sintassi concreta e semantica, in quanto evidenzia unicamente le categorie sintattiche dei costrutti del linguaggio e per ogni costruzione sintattica le categorie sintattiche degli argomenti e quelle del risultato.

Come altro esempio definiamo una sintassi astratta di

$$L_4 = \{\alpha\beta \mid \alpha \in L_1 , \beta \in L_3\}$$

ove L_1 è come sopra e $L_3 = \{a^n b^m \mid n,m \in N\}$

Basta definire la segnatura Σ_4



e considerare

$$\underline{L}_4 = (L_1, L_3, L_4, \lambda_4; \lambda'_4, \gamma_4, \delta_4, \nu_4, \mu_4)$$

ove $\lambda_4 = \lambda'_4 = \lambda$

$\gamma_4 : L_1 \times L_3 \to L_4$

$\delta_4 : L_1 \to L_1$

$\nu_4, \mu_4 : L_3 \to L_3$

$\gamma_4(x,y) = xy$ (concatenazione)

$\delta_4 x = axb$

$\nu_4 x = ax$

$\mu_4 x = bx$

T_{Σ_4} risulta evidentemente una sintassi astratta per L_4 .

56.- Sintassi algebrica di linguaggi a struttura di frase

In generale dato un linguaggio L a struttura di frase generato dalla grammatica G=(V,T,S,P) (dove V:alfabeto, T:terminali, S:simbolo iniziale, $P=\pi_1 \ldots \pi_n$ produzioni) la sua rappresentazione algebrica è data da

$$\underline{L} = (L, V^*, N, \varphi_0, \varphi_i, S)$$

ove $\forall\, \pi_i \in P^*$ $\qquad \varphi_i : V^* \times N \to V^*$

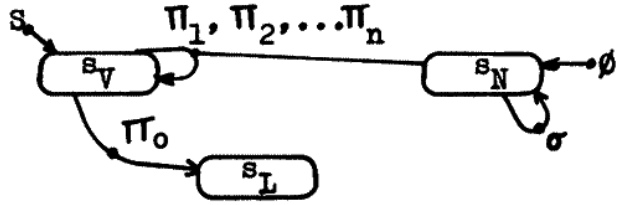
$$\varphi_i(n,x) = \begin{cases} y & \text{se } \pi_i : \alpha \to \beta \text{ e } y \text{ è ottenuta da } x \text{ sostituendo l'ennesima occorrenza di } \alpha \text{ con } \beta \\ x & \text{altrimenti} \end{cases}$$

$$\varphi_0(x) = \begin{cases} x & \text{se } x \in T^* \\ x_0 & \text{altrimenti} \end{cases}$$

(x_0 elemento fissato di L che supponiamo essere non vuoto).

Un'algebra che rappresenta la sintassi astratta di L è quindi un'algebra quoziente di T_Σ ove :

$\Sigma =$



(i naturali sono generati dallo zero tramite successore)

57.- ESERCIZI

Descrivere algebricamente sintassi astratte per i linguaggi

$$\{a^n b^m c^n \mid n,m \in N\}$$
$$\{a^n b^n b^n a^n \mid n \in N\}$$
$$\{a\alpha\, b \mid \alpha \in \{a,b\}^*\}$$

IX

TIPI DI DATO

58.- Tipo di dato astratto

Un tipo di dato è individuato da certi oggetti e da certe operazioni su di essi. È evidente che tale nozione intuitiva è completamente specificabile in modo preciso tramite quella di algebra.

Abbiamo per esempio già visto tra i primi esempi di algebra eterogenea il tipo di dato "pila su naturali". La stessa algebra $(N,+,\cdot,\emptyset,1)$ è un tipo di dato (il più antico).

L'approccio algebrico consente però in modo naturale la formalizzazione di un concetto di estremo interesse programmativo: quello di tipo di dato astratto. L'aggettivo astratto indica che gli oggetti su cui si vuole operare non sono definiti poichè ciò che interessa sono certe condizioni sulle operazioni che su di essi si intendono effettuare. In termini algebrici questo significa che un tipo di dato astratto non è un'algebra bensì una teoria algebrica i cui modelli risultano possibili realizzazioni concrete del tipo astratto.

Specificare algebricamente un tipo di dato astratto significa quindi tradurre in teoria algebrica una qualche teoria formale o informale che descriva il tipo considerato.

Consideriamo i seguenti esempi:

I) Pile su elementi di un prefissato insieme

Le pile sono oggetti costituiti a partire da una pila vuota empty e una operazione di "push" tale che

$$P \text{ è pila, } a \text{ è elemento} \Longrightarrow \text{push}(a,P) \text{ è pila}$$

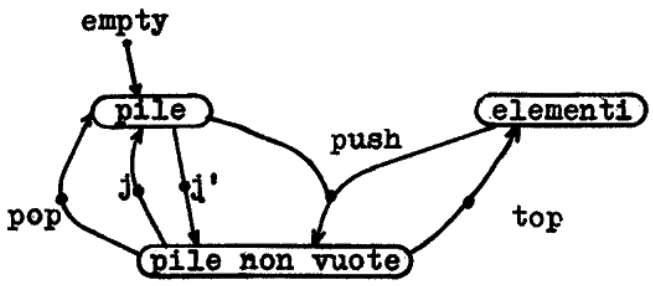
$$P' = \text{push}(a,P) \Longrightarrow \text{top}(P') = a$$

$$P' = \text{push}(a,P) \Longrightarrow \text{pop}(P') = P$$

(top e pop sono operazioni che agiscono su pile non vuote.)

Una specifica algebrica è la seguente

Segnatura



P : variabile di pile
Q : " " pile non vuote
e : " " elementi

Equazioni

j'(jP)= P Assioma che impone l'iniettività dell'immersione j .

pop (push (e,P)) = P

top (push (e,P)) = e

II) Liste su un prefissato insieme A

Le liste sono oggetti costituiti a partire dagli elementi di A detti atomi, utilizzando una operazione di costruzione **cons** secondo le regole seguenti :

o) Liste e atomi costituiscono insieme le espressioni

i) a è atomo, l è lista ⟹ cons (a,l) è lista

l è lista, l' è lista ⟹ cons (l,l') è lista

ii) Vi è un solo oggetto che è sia atomo che lista indicato con **Nil** e altre due operazioni **car** e **cdr** definite su tutte le liste tranne Nil come segue

l'= cons(e,l) ⟹ car (l') = e

l'= cons(e,l) ⟹ cdr (l') = l

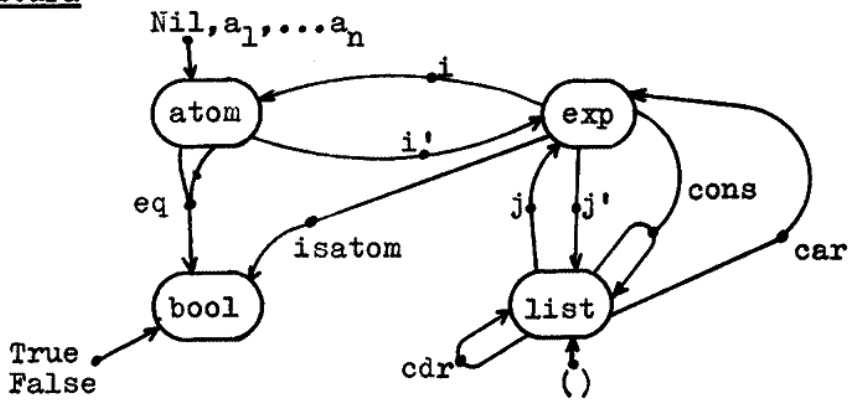
iii) Vi sono due operazioni booleane

eq binaria su atomi:
eq(a,a')= True $\iff$ a = a'

isatom unaria su espressioni:
isatom(a)= True $\iff$ a è atomo

Specifica algebrica:

Segnatura



$a_1, a_2, \ldots$ costanti, una per ogni elemento di A

η - variabile di exp
l - " " list
a - " " atom

Abbreviazioni:

$$\text{cons}(\eta, l) = (\eta\, l)$$
$$(\eta_1(\eta_2\eta_3)) = (\eta_1\eta_2\eta_3)$$

Equazioni:

$$\text{car}((\eta\, l)) = \eta$$
$$\text{cdr}((\eta\, l)) = l$$
$$\text{car}(\,()\,) = j()$$
$$\text{cdr}(\,()\,) = ()$$
$$i\ \text{Nil} = j()$$

$$\begin{cases} j'(jl) = l \\ i'(ia) = a \end{cases}$$ iniettività delle immersioni di liste e atomi in espressioni

$isatom(ia) = True$

$isatom((\eta l)) = False$

$eq(a,a) = True$

$eq(a_i, Nil) = False \quad i = 1,2,\ldots$

$eq(a_i, a_j) = False \quad i,j = 1,2,\ldots \; , \; i \neq j$

59.- Specifiche gerarchiche

Supponiamo ora di volere specificare il tipo di dato astratto Pila su naturali ove i naturali sono l'usuale algebra

$$\underline{\underline{N}} = (N, B, +, \cdot, =, \emptyset, 1)$$

e poi a partire da tali pile definire altre strutture astratte di dati. Tale modo di procedere è tipico delle cosidette specifiche gerarchiche. Un modo semplice di sviluppare tale procedimento per via algebrica si ottiene tramite la nozione di **algebra basata** che è una variante del concetto di algebra di dati (cfr. [35]).

DEFINIZIONE

Sia $\underline{\underline{B}}$ una algebra di segnatura Σ e sia Σ' una segnatura che estende Σ , cioè avente tra le sorte quelle di Σ e tra gli operatori quelli di Σ

— Una Σ'-algebra $\underline{\underline{A}}$ dicesi **basata** su $\underline{\underline{B}}$ se

$$\underline{\underline{A}}|_{\Sigma} \cong \underline{\underline{B}}$$

cioè se il sistema di domini e operazioni di $\underline{\underline{A}}$ relativi a Σ è una Σ-algebra isomorfa a $\underline{\underline{B}}$.

$\underline{\underline{A}}$ dicesi **sovrabasata** su $\underline{\underline{B}}$ se $\underline{\underline{B}}$ è isomorfa ad una sottoalgebra di $\underline{\underline{A}}|_{\Sigma}$

— Sia Σ_B la segnatura in cui tutti gli elementi dei domini di $\underline{\underline{B}}$ sono aggiunti come costanti (ciascuno nella sorta opportuna) dicesi diagramma di $\underline{\underline{B}}$ il seguente insieme di equazioni δ_B

$$\delta_B = \{ t_1 = t_2 \mid t_1, t_2 \in T_{\Sigma_B} , \underline{\underline{B}} \models t_1 = t_2 \}$$

Una teoria algebrica che includa δ_B dicesi B-basata.

È evidente che una specifica gerarchica è descrivibile algebricamente tramite teorie algebriche basate. Quindi se $\mathcal{E}$ è la teoria delle pile su elementi generici e δ_N è il diagramma di $\underline{\underline{N}}$ (ove ad N è assegnata la sorta elementi) $\mathcal{E} \cup \delta_N$ sarà la teoria delle pile su N.

Vale inoltre per le teorie basate il seguente teorema fondamentale

TEOREMA (sulle teorie basate; cfr. per la prova [27])

Dato un insieme $\mathcal{E}$ di Σ'-equazioni $\mathcal{E}$ dicesi

— B-consistente sse $\mathcal{E} \cup \delta_B \not\vdash a = a'$
con a, a' elementi distinti di $\underline{\underline{B}}$

— B-completo sse $\forall\, t \in \underline{\underline{T}}_{\Sigma_B} \mid_{\Sigma}$ $\mathcal{E} \cup \delta_B \vdash t = a$
per qualche $a \in \underline{\underline{B}}$

Con tali notazioni valgono i fatti seguenti:

i) Se $\mathcal{E}$ è B-consistente e B-completa allora la classe delle Σ'_B-algebre B-basate che verificano $\mathcal{E}$ ha $T_{\Sigma'_B / \overline{(\mathcal{E} \cup \delta_B)^*}}$ come algebra iniziale.

ii) Se $\mathcal{E}$ è B-consistente la classe delle algebre B-sovrabasate che verificano $\mathcal{E}$ ha $T_{\Sigma'_B / \overline{(\mathcal{E} \cup \delta_B)^*}}$ come algebra iniziale.

iii) Se $\mathcal{E}$ è B-consistente e B-completa la classe

delle algebre minimali B-basate che verificano
$\mathcal{E}$ ha algebra finale.

60.- Specifiche per inizialità

Un altro caso di specifica astratta è il seguente.

Definiamo matematicamente una Σ_o algebra che rappresenta il tipo di dato Lista ove Σ_o è la segnatura prima considerata a proposito delle liste, con l'esclusione di i' e j' :

$$\underline{\underline{L}} = (A, B, E, L, cons^L, car^L, cdr^L, isatom^L, eq^L, a_1^L, \ldots, j^L, i^L, ()^L, Nil^L)$$

ove

$A = \{a_1, \ldots\} \cup \{Nil\}$ (Atomi)

$B = \{True, False\}$ (Booleani)

$L_o = \{()\}$

$L_{i+1} = L_i \times L_i \cup A \times L_i$

$L = \bigcup_{i \in N} L_i$ (liste)

$E = A \cup L$ (espressioni)

$Nil^L = Nil$, $()^L = ()$, $a_i^L = a_i$ per $i = 1, \ldots$

$cons^L$= operazione di accoppiamento cartesiano

car^L = prima proiezione di coppia cartesiana

cdr^L = seconda proiezione di coppia cartesiana

$isatom^L$, eq^L : usuali test booleani

i, j : immersioni di atomi e liste in espressioni

Possiamo affermare che ogni algebra isomorfa ad $\underline{\underline{L}}$ rappresenta bene il tipo di dato Liste e in tal senso la classe di isomorfismo di $\underline{\underline{L}}$ rappresenta un tipo di

dato astratto.

Notiamo subito la differenza tra la nozione prima vista di tipo di dato astratto e questa qui considerata. La prima si identifica con una teoria e quindi con la classe di tutte le algebre che la soddisfano, mentre la seconda si riferisce alla classe di isomorfismo di un algebra. Per rimarcare tale differenza nel secondo caso parleremo di tipo di dato astratto categorico.

Se $\underline{\underline{I}}$ è l'algebra iniziale relativa alla teoria prima considerata sulle liste, si può verificare facilmente che $\underline{\underline{L}}$ è isomorfa a $\underline{\underline{I}}|_{\Sigma_0}$ ove Σ_0 è la segnatura di $\underline{\underline{I}}$ privata dagli operatori i' e j'.

In tal senso $\underline{\underline{I}}|_{\Sigma_0}$ dicesi specifica via inizialità dell'algebra $\underline{\underline{L}}$ e quindi del tipo di dato astratto individuato da $\underline{\underline{L}}$.

Le dimostrazioni di isomorfismo tra $\underline{\underline{I}}|_{\Sigma_0}$ e $\underline{\underline{L}}$ costituiscono le dimostrazioni di correttezza della specifica.

È evidente che la fondamentale differenza tra la definizione prima data di $\underline{\underline{L}}$ e quelle di $\underline{\underline{I}}$ è che la prima fa uso di nozioni matematiche usuali, mentre la seconda è data tramite una teoria algebrica ovvero in un linguaggio formale in cui si utilizzano solo le variabili, gli operatori e l'uguaglianza e si dispone di una ben precisa nozione di derivabilità (il calcolo equazionale) e di soddisfacibilità.

Notiamo che per specificare tramite inizialità una data Σ-algebra può essere utile introdurre degli operatori non presenti in Σ che agevolano la caratterizzazione assiomatica. Per esempio se si vuole specificare per inizialità la Σ-algebra $\underline{\underline{N}}$ dei naturali con la somma $\underline{\underline{N}} = (N, +)$ ($\Sigma = \{+\}$) è del tutto usuale considerare i seguenti assiomi nella segnatura $\Sigma' \supset \Sigma$ in cui

oltre la somma + , sono indicati lo zero e il succes= sore ' :

$$\mathcal{E} = \begin{cases} x + 0 = x \\ x + y' = (x + y)' \end{cases}$$

È evidente che

$$T_{\Sigma'}/\mathcal{E}^*|_{\Sigma} = \underline{\underline{N}}$$

Tali operatori aggiunti nella specifica e poi non con= siderati a costruzione avvenuta diconsi **operatori na= scosti.**

Nel caso della specifica per inizialità delle liste, i' e j' erano operatori nascosti, ma in tal ca= so la loro presenza nella specifica per inizialità po= teva essere evitata infatti se $\mathcal{E}_0$ sono gli assiomi delle liste senza quelli relativi a i'e j' si può verificare facilmente che

$$\underline{\underline{T}}_{\Sigma_0}/\mathcal{E}^* = \underline{\underline{L}}$$

(Σ_0 segnatura di $\mathcal{E}_0$) .

61.- Estensioni di tipi di dato

La tecnica di specifica tramite inizialità può essere ovviamente estesa utilizzando le teorie basate secondo la proposizione i) del teorema fondamentale delle teorie basate(cfr. 59).

Inoltre tramite teorie basate si può precisare il concetto di "estensione" di tipo di dato:

consideriamo per esempio il tipo di dato astratto "vettori sui naturali" esso può essere identificato con la classe di isomorfismo dell'algebra

$$\underline{V} = (V, N, -(-), -[-/-], \underline{0}, \emptyset)$$

ove N sono i "naturali"

$V = \{ f \mid f: N \rightarrow N \}$

$-(-) = V \times N \rightarrow N$ $\quad$ v(i) individua il numero di posizione i nel vettore v

$-[-/-] : V \times N \times N \rightarrow V$ $\quad (v[n/i])(j) = \begin{cases} v(j) & \text{se } i \neq j \\ n & \text{se } i = j \end{cases}$

$\underline{Q}(i) = \emptyset \quad \forall\, i \in N$

Se volessimo specificare assiomaticamente tale algebra sarebbe naturale imporre i seguenti assiomi (la segnatura Σ' è evidente)

$$\underline{Q}(i) = \emptyset \qquad \forall\, i \in N$$

$$(v[n/i])(i) = n \qquad \forall\, i \in N$$

$$(v[n/i])(j) = v(j) \qquad \forall\, i,j \in N\ ;\ i \neq j$$

Se ora consideriamo l'algebra iniziale $\underline{\underline{I}}$ di tale teoria basata ci si convince però che $\underline{\underline{I}}$ non è isomorfa a $\underline{\underline{V}}$ poichè dagli assiomi di sopra non deriva

$$(\underline{Q}\,[n/n])[m/m] = (\underline{Q}\,[m/m])[n/n]$$

e quindi i due membri di tale equazione designano elementi diversi di $\underline{\underline{I}}$.

In effetti l'algebra $\underline{\underline{V}}$ è isomorfa all'algebra $\underline{\underline{T}}_{\Sigma'/\theta}$ ove θ è la congruenza definita come segue.

Sia Σ la segnatura relativa alla base di $\underline{\underline{I}}$ contenente solo la sorta dei naturali e l'operatore nullario per lo zero ($\Sigma \subset \Sigma'$).

Diciamo **contesto basato** un termine W(x) di sorta in Σ con una variabile libera, poniamo quindi in $\underline{\underline{T}}_{\Sigma'}$

$$t_1 \theta t_2 \iff [W(t_1)]_\tau = [W(t_2)]_\tau$$

per ogni contesto basato $W(x)$ ove $W(x)$ è ottenuta sostituendo t al posto di x in $W(x)$, e τ è un assegnamento qualsiasi nell'algebra $\underline{I}$ (qualsiasi poichè in $W(t_1)$ e $W(t_2)$ non vi sono variabili) .

Ovvero due termini sono considerati equivalenti se sono "osservabilmente" equivalenti sulla base, cioè se in $\underline{I}$ (e quindi in tutte le algebre basate che verificano gli assiomi dati) il loro comportamento sulla base è identico (dal punto di vista algebrico).

Si dimostra che tale algebra $\underline{T}_{\Sigma/\theta}$ è in effetti l'algebra finale nella classe di tutte le algebre minimali basate sui naturali. La proposizione iii) del teorema sulle teorie basate fornisce quindi un criterio per la specifica algebrica, via finalità delle estensioni astratte di tipo di dato.

62.- Implementazioni di tipi di dato

Un altro fenomeno interessante sui tipi di dato, esprimibile per via algebrica, è l'implementazione.

Data un Algebra $\underline{A}$ diciamo che una algebra $\underline{D}$ è un algebra derivata da $\underline{A}$ se i domini di $\underline{D}$ sono sottoinsiemi dei domini di $\underline{A}$ e le operazioni di $\underline{D}$ sono del tipo

$$\lambda x_1 \ldots x_n \; t(x_1 \ldots x_n)$$

ove $t(x_1 \ldots x_n)$ è un termine sull'algebra $\underline{A}$ di variabili $x_1 \ldots x_n$.

Una implementazione di un tipo di dato astratto $\underline{C}$ su $\underline{A}$ è un algebra $\underline{D}$ derivata da $\underline{A}$ tale che $\underline{C} \cong \underline{D}$. La dimostrazione dell'isomorfismo fornisce la correttezza dell'implementazione.

ESEMPIO di implementazione

Sia $\underline{\underline{\mathcal{S}}}$ il seguente tipo di dati

$\underline{\underline{\mathcal{S}}} = (\mathcal{S}, B, delete, insert, set, has, eq, empty)$

ove

$B = \{True, False\}$

empty= l'insieme vuoto

$\mathcal{S}_0 = \{empty\}$

$\mathcal{S}_{i+1} = \mathcal{P}(\mathcal{S}_i) \cup \mathcal{S}_i$

$\mathcal{S} = \bigcup_{i \in N} \mathcal{S}_i$

$\begin{cases} delete : \mathcal{S} \times \mathcal{S} \to \mathcal{S} \\ delete\ (x,y) = y - \{x\} \end{cases}$

$\begin{cases} insert : \mathcal{S} \times \mathcal{S} \to \mathcal{S} \\ insert\ (x,y) = y \cup \{x\} \end{cases}$

$\begin{cases} eq : \mathcal{S} \times \mathcal{S} \to B \\ eq(x,y) = T \iff x = y \end{cases}$

$\begin{cases} has : \mathcal{S} \times \mathcal{S} \to B \\ has(x,y) \iff x \in y \end{cases}$

$\begin{cases} set : \mathcal{S} \to \mathcal{S} \\ set(x) = \{x\} \end{cases}$

Implementiamo in $\underline{\underline{\mathcal{S}}}$ il tipo di dati naturali, zero, successore (N, O, S)

Basta definire $\underline{\underline{D}} = (D_N, O^D, S^D)$ ponendo:

1) o) $empty \in D_N$

   i) $x \in D_N \Rightarrow x \cup \{x\} \in D_N$

2) $O^D = empty$

3) $S^D = \lambda x.\ insert(x,x)$

È evidente che $\underline{D}$ è isomorfa a $(N,0,S)$ infatti $\underline{D}$ non è altro che il classico modello insiemistico di Von Neumann dei numeri naturali.

Una implementazione di un tipo di dato astratto $\mathfrak{E}$ su un tipo di dato $\underline{A}$ è un algebra $\underline{D}$ derivata da $\underline{A}$ che verifica $\mathfrak{E}$. La dimostrazione che $\underline{D} \models \mathfrak{E}$ costituisce la dimostrazione di correttezza dell'implementazione.

63.- Problemi di specifica formale

La problematica di specifica formale dei tipi di dato ha in sè tutti gli aspetti tipici presenti in generale nella specifica formale del software :

i) definizione formale di strutture e teorie che descrivano astrattamente nozioni programmative;
ii) traduzione di specifiche date, da un certo linguaggio in
iii) verifica di correttezza di una certa specifica nei confronti di un altra.

Diamo alcune veloci esemplificazioni di tale problematica.

Sia data una specifica (in linguaggio del 1º O) del seguente tipo di dato astratto coda

predicati: coda(-), elem(-), top(-,-), pop(-,-), push(-,-,-)

costanti: empty, a, b,....

assiomi:

$$\forall x\ (coda(x) \vee elem(x))$$

$$elem(a)$$
$$elem(b)$$
$$\vdots$$
$$coda(empty)$$

$\forall x\ y\ z\ (push(x,y,z) \longleftrightarrow coda(x) \wedge elem(y) \wedge coda(z))$
$\forall\ x\ y\ \ (pop(x,y) \longleftrightarrow coda(x) \wedge \sim x=empty \wedge coda(y)\ )$
$\forall\ x\ y\ \ (top(x,y) \longleftrightarrow coda(x) \wedge \sim x=empty \wedge elem(y)\ )$
$\forall\ x\ ((coda(x) \wedge \sim x = empty) \longrightarrow (\exists!y(top(x,y) \wedge \exists!z(pop(x,z)))$
$\forall\ x\ y\ (push(empty,x,y) \longleftrightarrow top(y,x) \wedge pop(y,empty))$
$\forall\ x\ y\ z\ u\ v\ (pop(x,y) \wedge push(x,z,u) \wedge push(y,z,v) \longrightarrow pop(u,v))$
$\forall\ x\ y\ z\ u\ (top(x,y)\ \ push(x,z,u) \longrightarrow top(u,p))$
$\forall\ x\ y\ (coda(x) \wedge coda(y) \rightarrow (x=y) \longleftrightarrow (x=empty \wedge y=empty) \vee \exists uv(top(x,y) \wedge top(y;u) \wedge pop(x,v) \wedge pop(y;v))$

PROBLEMI

I) Specificare algebricamente tale teoria e dimostrare la correttezza della specifica fornita, ovvero dimostrare una corrispondenza biunivoca tra i modelli delle due teorie.

II) Definire un modello della teoria algebrica data in i) verificando la correttezza di tali definizioni cioé che i modelli verificano la teoria.

III) Specificare algebricamente per inizialità alge= bre date in ii) e verificare la correttezza delle specifiche.

IV) Implementare le algebre definite in ii) sull'alge= bra delle liste e verificare la correttezza delle implimentazioni.

V) Definire per inizialità il tipo di dati naturali

$$(N, B, +, \cdot, \emptyset, 1, \equiv, True, False)$$

($\equiv$ predicato di uguaglianza tra numeri)

VI) Definire per inizialità il tipo di dati stringhe su $A = \{a, b\}$

$$\mathcal{S} = (A^*, \{True, False\}, \lambda, \equiv, \cup, a, b)$$

($\equiv$: uguaglianza tra stringhe)

VII) 1) Specificare algebricamente l'usuale tipo di

dato file (su certi elementi A) con le operazioni di get, put, reset , read , endoffile.

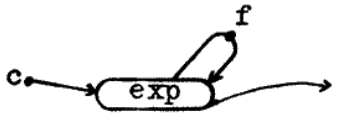
2) Specificare algebricamente un tipo di dato matrici booleane 2x3 con l'operazione di somma (componente per componente) estrazione di un elemento da matrice, estrazione di riga ed estrazione di colonna.

X

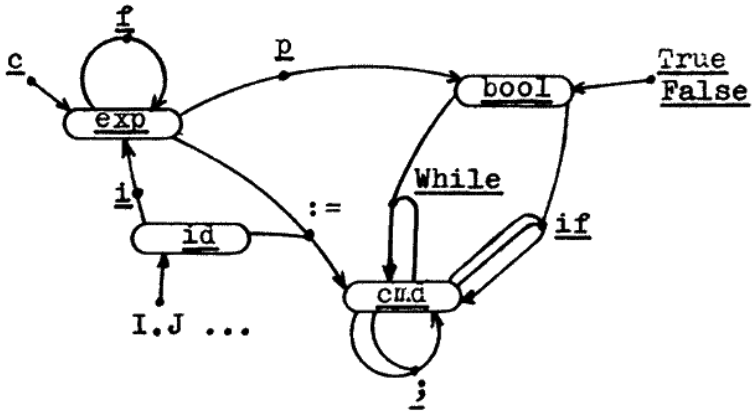
SEMANTICA DI LINGUAGGI

64.- Semantica di un semplice linguaggio imperativo

Consideriamo un tipo di dati $\underline{A} = (A, f^A, c^A)$ con la seguente segnatura



Possiamo costruire su $\underline{A}$ il linguaggio $\underline{A}$-While avente sintassi astratta espresa dalla seguente segnatura di evidente lettura



($\underline{i}$ rappresenta l'immersione di identificatori in espressioni che per brevità nel seguito non indicheremo)

Indichiamo con **EXP , ID , CMD** gli insiemi di termini di tale segnatura di sorte exp, id, cmd rispettivamente e definiamo il seguente insieme

$$\mathbf{STA} = \{\rho \mid \rho : \mathbf{ID} \longrightarrow \mathbf{A}\}$$

consideriamo le seguenti funzioni:

— aggiornamento $-[-/-] : \mathbf{STA} \times \mathbf{A} \times \mathbf{ID} \longrightarrow \mathbf{STA}$

ove $(\rho[a/I])I = a$

e $(\rho[a/I])J = \rho J \qquad \forall\, J \neq I$

— condizionale $-\supset-,-$: $\{True, False\} \times STA \times STA \rightarrow STA$

ove $True \supset \rho, \rho' = \rho$

$False \supset \rho, \rho' = \rho'$

È del tutto naturale postulare delle funzioni di interpretazione $\mathcal{E}$, $\mathcal{G}$, $\mathcal{C}$ tali che:

$\mathcal{E}$: EXP x STA $\rightarrow$ A $\cup$ {undefined}

$\mathcal{G}$: BOOL x STA $\rightarrow$ {True, False, undefined}

$\mathcal{C}$: CMD x STA $\rightarrow$ STA $\cup$ {undefined}

per cui valgano $\forall \rho \in STA$ le seguenti condizioni ove I, J, E, H, C, C_1, C_2 sono elementi generici di ID , EXP, BOOL, CMD rispettivamente ($\mathcal{C}$(while True do C)$_\rho$ =undefined)

$\mathcal{E}(\underline{c})\rho = c^A$

$\mathcal{E}(I)\rho = \rho I$

$\mathcal{E}(\underline{f}E)\rho = f^A(\mathcal{E}(E)\rho)$

$\mathcal{G}(\underline{True})\rho = True$

$\mathcal{G}(\underline{False})\rho = False$

$\mathcal{G}(\underline{p}E)_\rho = p^A(\mathcal{E}(E)\rho)$

$\mathcal{C}(C_1;C_2)\rho = \mathcal{C}(C_2)(\mathcal{C}(C_1)\rho)$

$\mathcal{C}(I:= E)\rho = \rho[\mathcal{E}(E)\rho/I]$

$\mathcal{C}(\underline{if}\ H\ \underline{then}\ C_1\ \underline{else}\ C_2)\rho = \mathcal{G}(H)\rho \supset \mathcal{C}(C_1)\rho, \mathcal{C}(C_2)\rho$

$\mathcal{C}(\underline{While}\ H\ \underline{do}\ C)\rho = \mathcal{G}(H)\rho \supset \mathcal{C}(\underline{While}\ H\ \underline{do}\ C)(\mathcal{C}(C)\rho), \rho$

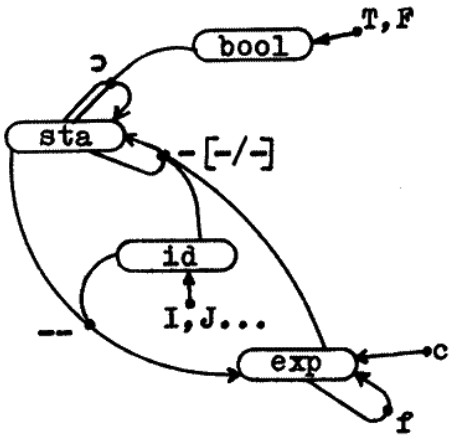
Anche se il significato di tali equazioni è del tutto intuitivo esse non possono essere considerate una definizione di $\mathcal{E}$ e $\mathcal{C}$ e questo per il fatto che

tali condizioni sono ricorsive in senso stretto.

Basta considerare equazioni del tipo $fx = fx$ oppure $fx = fx + 1$ ($x \in N$, $f: N \to N$) per capire come una equazione ricorsiva può avere infinite o nessuna soluzione e in quanto tale non è assumibile come condizione definitoria. Un modo per superare tale difficoltà è quella di leggere tali condizioni in opportune strutture entro cui è possibile dare loro un carattere definitorio. È questa la soluzione adottata in **semantica denotazionale** tramite la **teoria del punto fisso** (cfr.[10]).

Qui presenteremo una soluzione del problema completamente algebrica.

Consideriamo la seguente segnatura Σ

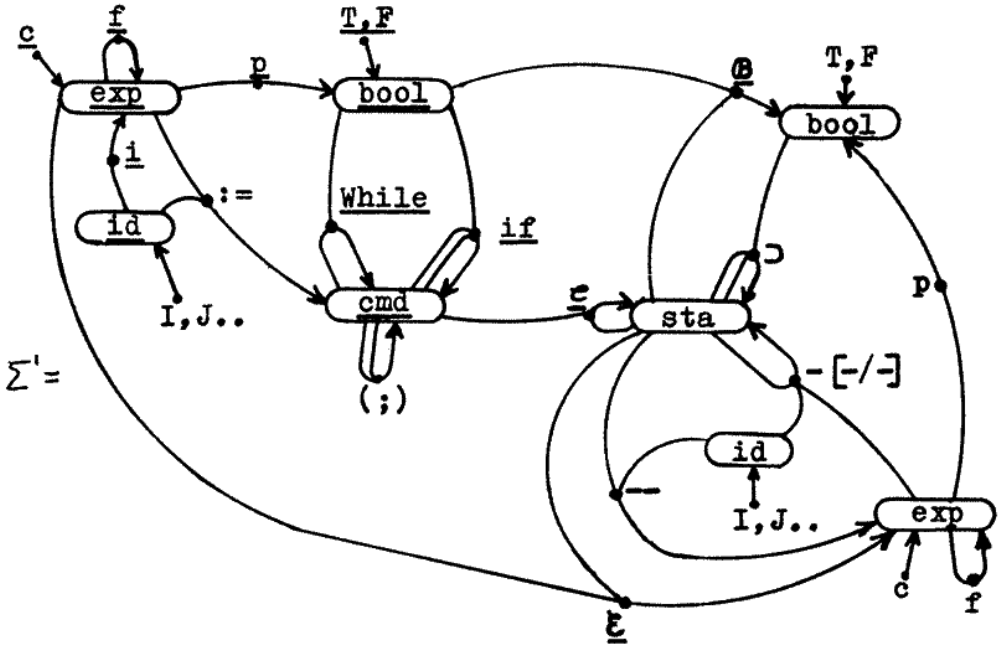


Sia $\underline{B}$ la Σ-algebra ove

$$B_{id} = ID \qquad B_{exp} = A \qquad B_{sta} = STA$$

e ove gli operatori $-[-/-]$, $-ɔ-,-$, $--$, individuano le operazioni di aggiornamento e condizionale come sopra definite e quella di applicazione funzionale.

Sia quindi Σ' la segnatura ottenuta unendo a Σ la segnatura della sintassi prima definita :



ove $\underline{\mathfrak{E}}$, $\underline{\mathfrak{B}}$, $\underline{\mathfrak{C}}$ sono operatori associati alle funzioni di interpretazione $\mathfrak{E}$, $\mathfrak{B}$, $\mathfrak{C}$ sopra considerate.

Consideriamo le equazioni semantiche di prima e riscriviamole come schemi di assiomi **Ax** nella segnatura Σ', ovvero sottolineando i simboli $\mathfrak{E}$, $\mathfrak{B}$, $\mathfrak{C}$ e sostituendo c^A ed f^A con c ed f rispettivamente. Se si verifica che tali equazioni così riscritte risultano B-consistenti, allora per il teorema fondamentale sulle teorie basate (punto ii)) la classe delle algebre B-sovrabasate che verifica **Ax** ha un'algebra iniziale $\underline{I}$.

Evidentemente $\mathfrak{E}^I$, $\mathfrak{B}^I$, $\mathfrak{C}^I$ dove:

$$\mathfrak{E}^I : I_{\underline{exp}} \times I_{sta} \longrightarrow I_{exp}$$

$$\mathfrak{B}^I : I_{\underline{bool}} \times I_{sta} \longrightarrow I_{bool}$$

$$\mathfrak{C}^I : I_{\underline{cmd}} \times I_{sta} \longrightarrow I_{sta}$$

verificano le equazioni semantiche date all'inizio, ma

i domini e codomini di $\underline{\underline{\mathcal{E}}}^I$, $\underline{\underline{\mathcal{B}}}^I$, $\underline{\underline{\mathcal{C}}}^I$ includeranno quelli specificati per le funzioni di interpretazione poichè $\underline{\underline{I}}$ è sovrabasata su $\underline{\underline{B}}$.

In effetti poichè in **Ax** non vi sono assiomi in cui si eguagliano termini di sorte **exp** o **cmd** avremo che $I_{exp} = EXP$ e $I_{cmd} = CMD$; però $I_{sta} \supset STA$ e l'inclusione (a meno di biunivocità, ovvero $I_{sta} \supset STA'$ e STA', STA biunivoci) è stretta in quanto, per esempio $\forall \rho$ non esiste un $\rho' \in STA$ tale che

$$\rho' = \underline{\underline{\mathcal{C}}}^I(\underline{\text{While}}\ I = I\ \text{do}\ I := J)\rho$$

e quindi anche $I_{exp} \supset A$ (a meno di biunivocità).

Possiamo però definire facilmente le funzioni se= mantiche postulate all'inizio ponendo

$$\mathcal{E}(E)\rho = \underline{\underline{\mathcal{E}}}^I(E)\rho \in A \supset \underline{\underline{\mathcal{E}}}^I(E)\rho\ ,\ \text{undefined}$$

$$\mathcal{C}(c)\rho = \underline{\underline{\mathcal{C}}}^I(c)\rho \in STA \supset \underline{\underline{\mathcal{C}}}^I(c)\rho\ ,\ \text{undefined}$$

Lo schema adottato per l'$\underline{\underline{A}}$-While può generaliz= zarsi a qualsiasi linguaggio:

1) Si definisce la sintassi astratta;
2) Si individua una base $\underline{\underline{B}}$ e si scrivono le equazioni semantiche desiderate per delle funzioni di inter= pretazione;
3) Si riscrivono le equazioni semantiche come teoria algebrica la cui segnatura include quelle della sin= tassi astratta e quella della base e degli operatori interpretativi relativi alle funzioni interpretative che connettono sintassi con base;
4) Si dimostra la B-consistenza della teoria algebrica ottenuta, da cui discende l'esistenza di un'algebra $\underline{\underline{I}}$

in cui valgono le equazioni di partenza per funzioni più definite di quelle semantiche desiderate (che in genere sono parziali);

5) Si restringono le funzioni relative agli operatori interpretativi di $\underline{\underline{I}}$ sui domini della base $\underline{\underline{B}}$ considerandole non definite se forniscono come risultati elementi non appartenenti alla base .

Il problema fondamentale per la applicabilità di tale metodo è la dimostrazione di B-consistenza della teoria algebrica che specifica la semantica del linguaggio. Esistono in proposito dei metodi che stabiliscono tale consistenza utilizzando tecniche induttive o risultati sui sistemi di riscrittura (cfr.[26] [27]) e addirittura per la maggior parte dei linguaggi programmativi tali metodi sono di applicazione semplicissima nel senso che si riducono a semplici controlli sintattici sulla specifica algebrica (cfr.[27])

Notiamo che il metodo presentato sviluppa una semantica che potremmo dire integrativa piuttosto che strettamente interpretativa, ovvero le funzioni semantiche non sono un morfismo tra algebre simili, ma connettono un' algebra sintattica a cui dare significato con un'algebra semantica (base) di oggetti assunti come già conosciuti.

Inoltre il calcolo equazionale permette in modo operativo di calcolare la semantica di un costrutto poichè se questo è un valore della base allora il calcolo lo ottiene in un numero finito di passi (altrimenti il calcolo diventa infinito) (cfr.[27]).

Consideriamo un comando in $\underline{\underline{N}}$-While ove

$$\underline{\underline{N}} = (N, +^N, <^N, 0^N, 1^N)$$

e calcoliamone la semantica utilizzando la sua specifica

algebrica su uno stato ρ_0 tale che $\rho_0 J = 0 \quad \forall J \in ID$
(abbrevieremo 0^N e 1^N con $0,1$)

$\mathcal{C}$(While I < 1+1 do I:= I+1)ρ_0 =

$\mathcal{B}(I < 1+1)\rho_0 = \mathcal{E}(I)\rho_0 <^N \mathcal{E}(1+1)\rho_0 =$

$= \rho_0 I <^N \mathcal{E}(1)\rho_0 +^N \mathcal{E}(1)\rho_0 =$

$= 0 <^N 1 +^N 1 = 0 <^N 2 =$ True

$= \mathcal{C}$(While I < 1+1 do I:= I+1)($\mathcal{C}$(I:= I+1)ρ_0)=

$\mathcal{C}(I := I+1)\rho_0 = \rho_0[\mathcal{E}(I+1)\rho_0/I] =$

$= \rho_0[\mathcal{E}(I)\rho_0 +^N \mathcal{E}(1)\rho_0/I] =$

$= \rho_0[\rho_0 I +^N 1/I] = \rho_0[0 +^N 1/I] =$

$= \rho_0[1/I] = \rho_1$

$= \mathcal{C}$(While I < 1+1 do I:= I+1)ρ_1 =

$\mathcal{B}(I < I+1)\rho_1 = \mathcal{E}(I)\rho_1 <^N \mathcal{E}(1+1)\rho_1 =$

$= \rho_1 I <^N \mathcal{E}(1)\rho_1 +^N \mathcal{E}(1)\rho_1 =$

$= 1 <^N 1 +^N 1 = 1 <^N 2 =$ True

$= \mathcal{C}$(While I < 1+1 do I:= I+1) ($\mathcal{C}$(I:= I+1)ρ_1) =

$\mathcal{C}(I := I+1)\rho_1 = \rho_1[\mathcal{E}(I+1)\rho_1/I] =$

$= \rho_1[\mathcal{E}(I)\rho_1 +^N \mathcal{E}(1)\rho_1/I] =$

$= \rho_1[\rho_1 I +^N 1/I] = \rho_1[1 +^N 1/I] =$

$= \rho_1[2/I] = \rho_2$

$= \mathcal{C}$(While I < 1+1 do I:= I+1)ρ_2 =

$$\Big|\ B(I < 1+1)\rho_2 = \rho_2 I <^N 2 = 2 <^N 2 = \text{False}$$

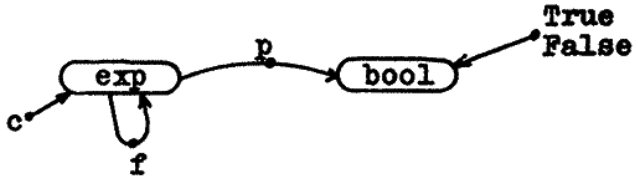
$$= \rho_2$$

65.- Semantica algebrica di un semplice linguaggio funzionale .

Consideriamo ancora un algebra $\mathbb{A}$

$$\mathbb{A} = (A, \{\text{True}, \text{False}\}, f^A, c^A, p^A)$$

di segnatura



Definiamo a partire da $\mathbb{A}$ il linguaggio $\mathbb{A}$-McCarthy

La sintassi astratta di tale linguaggio in BNF è la seguente

<Program> ::= eval (<Declaration> , <Expression>)

<Expression> ::= <Identifier> | f <Expression> | _

call <Function name> (<Expression>)

if <Boolean> then <Expression>

else <Expression>

<Declaration> ::= define <Function name> (<Identifier>) =

= <Expression> |

<Declaration> ; <Declaration>

<Boolean> ::= p <Expression> | True | False

<Identifier> ::= I | I_1 | I_2 ....

<Function name> ::= F | F_1 | F_2 ....

Indichiamo con PRG, EXP, DEC, BOOL, ID, FUN gli insiemi di termini della segnatura associata alla grammatica di sopra le cui sorte indicheremo con prg, exp dec, bool, id, fun rispettivamente.

Definiamo quindi un algebra base $\underline{B}$ di domini

$$B_{val} = A \cup \{error\}$$

$$B_{bool} = \{True, False, error\}$$

$$B_{exp} = EXP$$

$$B_{id} = ID$$

$$B_{fun} = FUN$$

$$B_{def} = \{\delta \mid \delta : FUN \rightarrow EXP \times ID \cup \{unbound\}\}$$

$$B_{sta} = \{\rho \mid \rho : ID \rightarrow A \cup \{unbound\}\}$$

Su tali domini assumiamo le funzioni definite dalle seguenti condizioni:

— Applicazioni funzionali

$$\rho \in B_{sta},\ I \in B_{id} \Rightarrow \rho I \in B_{exp} \text{ (applic.di } \rho \text{ a } I)$$

$$\delta \in B_{def},\ F \in B_{fun},\ \delta F = (E,I) \Rightarrow \begin{cases} \delta F \downarrow EXP = E \\ \delta F \downarrow ID = I \end{cases}$$

— Condizionali

$$a,b,a_1,a_2 \in B_{val},\ a_1 \neq a_2 \Rightarrow \begin{cases} True \supset a,b = a \\ False \supset a,b = b \\ a_1 = a_2 \supset a,b = b \\ a_1 = a_1 \supset a,b = a \end{cases}$$

— Aggiornamenti

$$(\rho[a/I])J = \begin{cases} a & \text{se } I = J \\ \rho J & \text{se } I \neq J \end{cases}$$

$$(\delta[(E,I)\ /\ F])G = \begin{cases} (E,I) & \text{se } F = G \\ \delta F & \text{se } F \neq G \end{cases}$$

— Test su espressioni ($E \in B_{exp}$)

$$free(E) = \begin{cases} True & \text{se } E \text{ contiene identificatori} \\ False & \text{altrimenti} \end{cases}$$

$$free(I,E) = \begin{cases} True & \text{se } E \text{ contiene identificatori diversi da } I \\ False & \text{altrimenti} \end{cases}$$

Nel seguito indicheremo gli operatori relativi a tali funzioni con la stessa simbologia adottata per le funzioni .

Possiamo quindi esprimere la semantica dell'$\underline{A}$-McCarthy tramite una segnatura composta a partire dalla sintassi e dalla segnatura di $\underline{B}$ aggiungendo i seguenti operatori (di interpretazione)

$\mathfrak{E}$: exp def sta → val

$\mathfrak{B}$: bool def sta → bool

$\mathfrak{D}$: dec def → def

$\mathfrak{P}$: prg → val

e dando le seguenti equazioni $\forall\ \varphi \in B_{sta}$, $\delta \in B_{def}$

con I elemento generico di sorta id e id$\cdot$ (B_{id}= ID)

E " " " " exp e exp (B_{exp}=EXP)

H " " " " bool

D " " " " dec

$\mathfrak{E}\ (c)\delta\varphi = c$

$\mathfrak{E}\ (I)\delta\varphi = \varphi I$

$\mathcal{E}(\underline{f}E)\delta\rho = f(\mathcal{E}(E)\delta\rho)$

$\mathcal{E}(\underline{\text{if}}\ H\ \underline{\text{then}}\ E_1\underline{\text{else}}\ E_2)\delta\rho = \mathcal{B}(H)\delta\rho = \text{error} \supset \text{error},$

$\mathcal{B}(H)\delta\rho = \text{True} \supset \mathcal{E}(E_1)\delta\rho, \mathcal{E}(E_2)\delta\rho$

$\mathcal{E}(\underline{\text{call}}\ F(E))\delta\rho = \mathcal{E}(E)\delta\rho = \text{error} \supset \text{error}, (\delta F = \text{unbound} \supset$

$\supset \text{error},\ \mathcal{E}(\delta F\downarrow EXP)\delta\rho[\mathcal{E}(E)\delta\rho | \delta F\downarrow ID])$

$\mathcal{B}(\underline{\text{True}}) = \text{True}$

$\mathcal{B}(\underline{\text{False}}) = \text{False}$

$\mathcal{B}(\underline{p}E)\delta\rho = p(\mathcal{E}(E)\delta\rho)$

$\mathcal{D}(\underline{\text{define}}\ F(I)=E)\delta = \delta F = \text{unbound} \supset (\text{free}(I,E) \supset \text{error},$

$\delta[(E,I)/F]),\ \text{error}$

$\mathcal{P}(\underline{\text{eval}}\ E\ \underline{\text{in}}\ D) = \mathcal{D}(D)\delta_0 = \text{error} \supset \text{error},\ \text{free}(E) \supset \text{error},$

$\mathcal{E}(E)(\mathcal{D}(D)\delta_0)\rho_0$

$\mathcal{D}(D_1;\ D_2)\delta = \mathcal{D}(D_2)(\mathcal{D}(D_1)\delta)$

$\rho_0 I = \text{unbound}$

$\delta_0 I = \text{unbound}$

E' del tutto naturale estendere la semantica sopra data nel caso in cui si ammettono chiamate di funzione con più di un argomento.

66- Descrizione algebrica del LISP

Un linguaggio di tipo LISP è costituito da un nucleo la cui sintassi astratta e la cui semantica sono riconducibili a quelle del Liste-McCarthy ove Liste è il tipo di dati di liste su stringhe alfanumeriche definibile formalmente come si è visto precedentemente.

Esprimendo la sintassi astratta in BNF avremo :

<Program> ::= eval (<Declaration> , <Expression>)

Ψ: <Expression> ::= <Identifier> | <List> | <Atom> |
car(<Expression>) | cdr(<Expression>) |
cons(<Expression> , <Expression>) |
<Function name> (<Argument>) |
if <Boolean> then <Expression>
else <Expression>

Π: <Argument> ::= <Expression> | <Argument> , <Expression> |

<Atom> ::= <Alfanum> <Atom> | <Alfanum>

<Alfanum> ::= A | B | ... | Z | Ø | 1 | ... | 9

Λ: <List> ::= l (l è lista del tipo di dati "liste su atomi")

Δ: <Declaration> ::= define <Function name> (<Identifier>)=
= <Expression> |
<Declaration> ; <Declaration>

χ: <Identifier> ::= I <Atom>

Φ: <Function name> ::= F <Atom>

Γ: <Boolean> ::= eq(<Atom>, <Atom>) | isatom(<Expression>) |
True | False

(Se Ψ è un'espressione scorretta del tipo car(nil) $\xi(\Psi)_{\delta\varrho}$ =error)

Ciò che però è peculiare del LISP è il fatto di avere una sintassi concreta espressa entro il tipo di dato liste; ovvero vi è una funzione τ di realizzazione sintattica che traduce i domini EXP, BOOL, ... in liste.

Avviene quindi che una certa espressione del Liste-McCarthy che individua una funzione su liste è in LISP rappresentata tramite una lista. Cosi in modo del tutto naturale si possono rappresentare in LISP programmi che manipolano programmi, ovvero algoritmi a vari livelli di applicazione funzionale.

Tale aspetto rende il LISP estremamente indicato in vari contesti programmativi e sopratutto in intelligenza artificiale.

Definiamo come esempio una realizzazione sintattica τ :

$\tau(\chi) = \chi$

$\tau(\blacksquare) = \blacksquare$

$\tau(\Psi,\Gamma)) = (\tau(\Psi), \tau(\Gamma))$

$\tau(\underline{\text{True}}) = \text{TRUE}$

$\tau(\underline{\text{False}}) = \text{FALSE}$

$\tau(\text{nil}) = \text{NIL}$

$\tau(\Lambda) = (\text{QUOTE}\ \Lambda) \qquad \forall\ \Lambda \neq \text{nil}$

$\tau(\ \underline{\text{eq}}(\Psi_1, \Psi_2)\ ) = (\text{EQ}\ \ \tau(\Psi_1)\ \tau(\Psi_2)\ )$

$\tau(\underline{\text{isatom}}(\Psi)\ ) = (\text{ISATOM}\ \ \tau(\Psi)\ )$

$\tau(\underline{cons}(\Psi_1,\Psi_2)) = (\text{CONS } \tau(\Psi_1)\ \tau(\Psi_2))$

$\tau(\underline{car}(\Psi)) = (\text{CAR } \tau(\Psi))$

$\tau(\underline{cdr}(\Psi)) = (\text{CDR } \tau(\Psi))$

$\tau(\underline{if}\ \Gamma\ \underline{then}\ \Psi_1\ \underline{else}\ \Psi_2) = (\text{COND } \tau(\Gamma)\ \tau(\Psi_1)\ \tau(\Psi_2))$

$\tau(\underline{define}(\Phi(\chi) = \Psi)) = (\text{DEFINE } \Phi(\chi)\ \tau(\Psi))$

$\tau(\Delta_1; \Delta_2) = (\tau(\Delta_1)\ \tau(\Delta_2))$

$\tau(F(\Pi)) = (F\ \tau(\Pi))$

$\tau(\underline{eval}(\Delta, \Psi)) = (\text{EVAL } \tau(\Delta)\ \tau(\Psi))$

Osserviamo che l'atomo **QUOTE** serve a distinguere le liste che sono nomi di costrutti da quelle che sono liste vere e proprie, ovvero elementi del tipo di dato. L'uso di QUOTE è analogo alle virgolette con cui in italiano distinguiamo Roma come nome di un ente extralinguistico e "Roma" come nome della parola.

Esempio di traduzione dal Liste-McCarthy al LISP.

P= eval(define FF(II)= if eq(II,nil) then FINE
else cons(CIAO,FF(cdr(II))) , FF((X X X)))

τ(P) = (EVAL (DEFINE FF(II) (COND (EQ II NIL) (QUOTE
FINE) (CONS (QUOTE CIAO) (FF (CDR II)))))
FF((QUOTE X X X)))

Esercizio : descrivere τ come morfismo

XI

SEMANTICA ALGEBRICA DELLO SMALL

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

Sia $\underline{A}=(A, f^A, c^A)$ 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::= $I_1 \mid I_2 \mid \ldots$

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) | 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 sequenze : 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 → Loc , funzione tale che new x fornisce la prima (in un prefissato ordine) locazione che nello Store x vale unused, error se tale locazione non c'è .

J : CMD → Env , funzione di legame per la etichetta go to come vedremo avanti

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

Osserviamo che la semantica del go to sarà trattata 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 sequenza (Jump o Continue) che informa se nella semantica del comando sono utilizzate equazioni semantiche di go to .

— Vi è una funzione J: CMD → Env che lega ogni etichetta 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 (While ...)

ii) C;While (...) altrimenti

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

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

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

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

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

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

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

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

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

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 x Env x Sta $\longrightarrow$ (Ev x Sta) + {error}

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

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

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

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

Equazioni semantiche

Nel seguito abbrevieremo (x,(y...z)) con (x,y...z)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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* :

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

Si può comunque definire $\mathbb{E}$ in altri modi ad esempio **per testo** o **per nome** ponendo rispettivamente $\mathbb{E}(E)\delta\varsigma = E$ o $\mathbb{E}(E)\delta\varsigma = (E,\delta)$, 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 $\underline{\mathbb{E}}(I)\delta\varsigma$ (e conseguentemente i domini semantici)ponendo:

per testo

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

per nome

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

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

Analogamente data una definizione di $\mathcal{D}$ si può definire la semantica con altri tipi di passaggi, per esempio **per valore** o **valore/risultato** ponendo:

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

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

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

ii) **per valore/risultato**

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

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

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 procedura con binding dinamico piuttosto che statico.

Il discorso è analogo nel caso della chiamata di funzione laddove $\mathcal{C}$ è sostituito da $\mathcal{E}$ 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 $\underline{I}$ iniziale nella classe di tutte le Σ'_β-algebre che soddisfano le date equazioni semantiche.

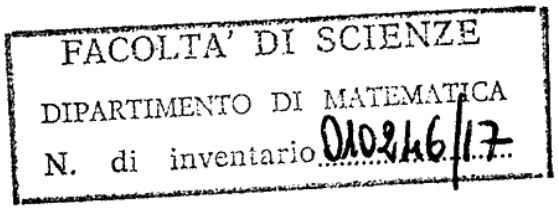
A partire da $\underline{\mathcal{E}}^I$, $\underline{\mathcal{R}}^I$, $\underline{\mathcal{C}}^I$, $\underline{\mathcal{D}}^I$, $\underline{\mathcal{P}}^I$ si possono quindi definire le funzioni di interpretazione $\mathcal{E}$, $\mathcal{R}$, $\mathcal{C}$, $\mathcal{D}$, $\mathcal{P}$, secondo quanto si è detto precedentemente (cfr. $\underline{A}$-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

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

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^*).



BIBLIOGRAFIA

[1] ADJ: R.M. Burstall; J.A. Goguen; An informal introduction to specifications using clear, in R.S.Boyer, J.S.Moore (cfr. [11])

[2] ADJ: J.A. Goguen, Some ideas in algebric semantics, Proceedings of 3rd IBM Symp. on Math. Foundations of Comp. Sc., Kobe, Japan, 1978

[3] ADJ: J.A. Goguen; J.W.Thatcher; E.G. Wagner; An initial algebra approach to the specification, implementation and correnteness of abstract dat types, in current trends in software methodology IV : Data structuring, R.Yeh (ed), Prentice Hall International 1978

[4] ADJ: J.A. Goguen; J.W.Thatcher; E.G. Wagner; J.B. Wright; Initial algebra semantics and continous algebras, J. ACM 24, 1977

[5] ADJ: E.G. Wagner; J.W. Thatcher; J.B. Wright; Programming languages as mathematical objects, in 7th MFCS, Lect. Not. in Comp. Sc. 64 , Springer, 1978

[6] H. Andreka; I. Nemeti; Some universal algebraic and model theoretic results in Computer Science, in Fundamentals in Computation theory, Lect. Not. in Comp. Sc. 117, Geeseg (ed), Springer, 1981

[7] K.R. Apt, Ten Years of Hoare's logic : A survey-= Part I, ACM Transations, Vol. 3. N.4 Oct.1981

[8] J.W. Backus, The syntax and semantics of the proposed International Algebraic Language of the Zürich ACM-GAMM Conference, ICIP Proceedings, Paris 1959, Butterworth's, London, 1960

[9] D. Bjørner (ed), Abstract software specification, Lect. Not. in Comp. Sc. 86, Springer, 1979

[10] D. Bjørner; C.B. Jones; Formal specification and software development, Prentice Hall International, 1982

[11] R.S. Boyer; J.S. Moore (eds), The correctness problem in Computer Science, Academic Press, 1981

[12] M. Broy; P. Pepper; M. Wirsing; On desing principles for programming languages: An algebraic approach, in algorithmic languages; IFIP Int. Symp. on Algorithmic Languages 1981, J.W. DeBakker, J.C. Van Vliet (ads.), North-Holland, 1981

[13] N. Chomsky, On certain formal formal properties of grammars, Information and Control, 2,2, 1959

[14] B. Courcelle; M. Nivat; The algebraic semantics of recursive program schemes, Lect. Not. in Comp. Sc. 64, Winkowski (ed), Springer, 1978

[15] E.W. Dijkstra; An attempt to unify the constituent concepts of serial program execution, Symbolic languages in data processing, Proc. ICC Symp., Roma 1962, Gordon & Breach, New York,1962

[16] G. Engels; U. Pletat; H.D. Ehrich; An operational semantics for specification of abstract data type with error handling, Acta Informatica, 19, 1983

[17] R.W. Floyd, Assigning meaning to programs, in J.T. Schwartz (ed.), Mathematical Aspects of Computer Science, Proc. of Symp. in Applied Math., Am. Math. Soc., R I, 1967

[18] M. Gordon, The denotational description of programming languages: An Introduction, Springer, 1979

[19] G. Grätzer, Universal Algebra, Springer, 1979

[20] C.A.R. Hoare, The axiomatic basis of computer programming, ACM comunications 12, Oct. 1969

[21] J. Hopcroft; J. Ullmann; Formal languages and their

relation to automata, Addison , Welsey, 1969.

[22] E. Horowitz; Programming Languages--Programming Languages: A grand tour, Springer, 1983 .

[23] G. Huet; Confluent reductions: abstract properties and applications to term rewriting systems, 18 th. IEEE Symposium on Foundations of Comp. Sc., 1977

[24] P.J. Landin; A corrispondence betwen Algol 60 and Church's lambda notation: Part I, ACM Comunication 2, 1965

[25] V. Manca; A. Salibra; First-order theories as many-sorted algebras, Notre Dame Journal of Formal Logic, Jan. 1984, Vol. 25,N° 1.

[26] V. Manca; A. Salibra; Inductively defined algebraic semantics, to appear

[27] V. Manca; A. Salibra; Semantics of programming languages throughbased algebras, to appear

[28] J. McCarthy : Toward a mathetatical science of computation, IFIP 62, C.M. Popplewell (ed.) 1963

[29] M.J. O'Donnel; Computing in systems described by equations, Lect. Not. in Comp. Sc., 58, Springer 1977

[30] C. Pair; Abstract data types and algebraic semantics of programming languages, Theoret. Comp. Sc., 18, 1982

[31] B.K. Rosen; Tree manipulating systems & Church Rosser theorems, ACM Journal 20, 1, 1973

[32] D.S. Scott; C. Strachey; Towards a mathematical *semantics for computer languages*, in Computers and Automata, J. Fox (ed.), John Wiley, New York, 1972

[33] M. Wand; Final algebra semantics and data type

extensions, Journal of Computer and System Sciences, 19, 1979

[34] P. Wegner; The Vienna definition language, ACM Comp. Survey, 4, 1, 1972

[35] S.N. Zilles; Introduction to data algebras; in Bjørner (ed.), Abstract Software Specification (cfr. [9]).

I N D I C E