Un linguaggio di programmazione ricorsivo - $\mathcal{L}_3$

- Un programma del linguaggio ricorsivo $\mathcal{L}_3$ è costituito da un unico termine che può contenere chiamate ricorsive a se stesso o a dei sotto-programmi
- $\mathcal{L}_3$ non è costruito sulla logica predicativa
- In $\mathcal{L}_3$ non viene fatta alcuna differenza fra termini e formule. Le formule sono considerate termini, mentre i connettivi simboli funzionali aggiuntivi
- Per poter distinguere i tipi diversi di termine è necessario dare ad ogni simbolo funzionale un 'tipo' che specifichi il dominio di appartenenza ($\boldsymbol{\mathcal{D}}$ oppure $\{\mathbf{true}, \mathbf{false}\}$) degli argomenti e dei valori delle funzioni

Simboli di $\mathcal{L}_3$

1. Simboli ausiliari $\{(,), , , \Leftarrow\}$
2. Un insieme ricorsivamente enumerabile di variabili Var
3. Un insieme ricorsivamente enumerabile W di variabili per funzione (ciascuno di questi simboli ha associata la propria arietà)

NOTA

I nomi delle variabili per funzione servono come nomi per il programma principale e per i sotto-programmi. Formalmente però sono delle variabili che variano su funzioni con uguale arietà.
Le variabili per funzione non vanno confuse con i simboli funzionali del linguaggio

Tipi di $\mathcal{L}_3$

Siano b e d due simboli che rievocano gli insiemi $\{\mathbf{true}, \mathbf{false}\}$ e $\boldsymbol{\mathcal{D}}$ rispettivamente

L'insieme dei tipi viene definito induttivamente come segue

- b e d sono tipi
- se $\beta_1, \ldots, \beta_s, \beta_{s+1} \in \{b, d\}$ per $s \geq 1$, allora $(\beta_1, \ldots, \beta_s \rightarrow \beta_{s+1})$ è un tipo

Base di un linguaggio di tipo $\mathcal{L}_3$

Una base di un linguaggio di programmi ricorsivi è semplicemente un insieme $\mathcal{F}$ di simboli funzionali

Ad ogni elemento di $\mathcal{F}$ viene assegnato un tipo

Un simbolo funzionale di tipo b o d viene detto 'costante'

Assumeremo che ogni base $\mathcal{F}$ ha almeno i seguenti simboli

- *true*, *false* di tipo b
- $\neg$ di tipo $(b \rightarrow b)$
- $\wedge, \vee, \supset, \equiv$ di tipo $(b, b \rightarrow b)$
- `if` $-$ `then` $-$ `else` di tipo $(b, d, d \rightarrow d)$
- $=$ di tipo $d, d \rightarrow b$

Sintassi dei termini di un linguaggio di tipo $\mathcal{L}_3$

L'insieme dei termini di tipo b e di tipo d sulla base $\mathcal{F}$ vengono definiti per induzione strutturale simultanea come segue

- Ogni costante da $\mathcal{F}$ di tipo b o d è un termine di tipo b o d rispettivamente

  Ogni variabile è un termine di tipo d
- Se $f \in \mathcal{F}$ è un simbolo funzionale di tipo $(\beta_1, \ldots, \beta_s \rightarrow \beta_{s+1})$ e se $t_1, \ldots, t_s$ sono termini di tipo $\beta_1, \ldots, \beta_s$ rispettivamente, allora $f(t_1, \ldots, t_s)$ è un termine di tipo β_{s+1}
- Se $F \in W$ è una variabile per funzione di arietà s e $t_1 \ldots, t_s$ sono termini di tipo d, allora $F(t_1, \ldots, t_s)$ è un termine di tipo d

Sintassi dei programmi di tipo $\mathcal{L}_3$

Un programma ricorsivo sulla base $\mathcal{F}$ viene definito come un insieme di n equazioni ricorsive, $n \geq 1$

$$F_1(x_{11}, \ldots, x_{1s_1}) \Leftarrow t_1$$
$$\vdots$$
$$F_n(x_{n1}, \ldots, x_{ns_n}) \Leftarrow t_n$$

insieme ad una variabile per funzione F_k, $1 \leq k \leq n$, detta la *variabile per funzione principale*

Sintassi dei programmi di tipo $\mathcal{L}_3$ (ctnd)

NOTA

- $F_1, \ldots, F_n$ sono distinte
- F_i è una variabile per funzione di arietà s_i
- $x_{i1}, \ldots, x_{is_i}$ sono variabili differenti (diverse equazioni possono condividere le stesse variabili. Le variabili vengono considerate locali)
- t_i è un termine di tipo d che può soltanto contenere $F_1, \ldots, F_n$ come variabili funzionali e $x_{i1}, \ldots, x_{is_i}$ come variabili

Semantica dei programmi di tipo $\mathcal{L}_3$

- Ogni equazione ricorsiva di un programma in $\mathcal{L}_3$ rappresenta la definizione di una procedura
- Una computazione di un programma ricorsivo inizia con una chiamata alla procedura che viene identificata con la variabile per funzione principale

Costruzione delle sequenze di computazione

- *Sostituzione*: chiamata di procedura
- *Semplificazione*: valutazione del corpo della procedura

Valori indefiniti

- Il linguaggio di programmazione $\mathcal{L}_3$ non contiene costrutti di programmazione oltre ai termini
- La computazione di un termine in $\mathcal{L}_3$ potrebbe non terminare
- In questo caso diciamo che il termine ha un valore *indefinito*
- Tale termine può apparire come argomento di una funzione o di un altra variabile per funzione

Bisogna definire esplicitamente il valore di una funzione per argomenti indefiniti

SOLUZIONE: I domini delle funzioni vengono estesi in modo da includere valori indefiniti

Valori indefiniti (ctnd)

$\boldsymbol{\mathcal{D}} \neq \emptyset$

Poniamo $\boldsymbol{\mathcal{D}}_\omega = \boldsymbol{\mathcal{D}} \cup \{\omega_d\}$ dove $\omega_d \notin \boldsymbol{\mathcal{D}}$, indica il valore indefinito di tipo d

Analogamente $Bool_\omega = Bool \cup \{\omega_b\}$

Sia $f : S_1 \times \ldots \times S_n \rightarrow S_{n+1}$ con $S_i = \boldsymbol{\mathcal{D}}$ o $S_i = Bool$, $1 \leq i \leq n+1$. Una estensione di f $g : S_{1\omega} \times \ldots \times S_{n\omega} \rightarrow S_{(n+1)\omega}$ viene detta una ω-*estensione* di f

Una ω-estensione viene detta *stretta* se prende valore ω tutte le volte che un argomento è ω

Ordini parziali piatti

Sia $S = \boldsymbol{\mathcal{D}}$ o $S = Bool$. Definiamo su S_ω la relazione $\sqsubseteq$ tale che dati $a, b \in S_\omega$, $a \sqsubseteq b$ se e solo se $a = \omega$ oppure $a = b$

Si legge: a *è meno definito di* b

$(S_\omega, \sqsubseteq)$ è un ordine parziale e viene chiamato *ordine parziale piatto* di S

Funzioni monotone

Siano $(P_1, \sqsubseteq)$ e $(P_2, \sqsubseteq)$ due ordini parziali. Una funzione totale $f : P_1 \to P_2$ viene detta *monotona* (rispetto a questi ordini parziali) se per ogni $a, b \in P_1$, $a \sqsubseteq b$ implica $g(a) \sqsubseteq g(b)$

Ci interessiamo alle funzioni monotone rispetto agli ordini parziali piatti

Notiamo che: Ogni funzione stretta è monotona

L'`if-then-else` sequenziale è una funzione monotona

Caratterizzazione delle funzioni monotone

Una funzione

$$g : S_{1\omega} \times \ldots \times S_{n\omega} \rightarrow S_{(n+1)\omega}$$

è monotona se e solo se per un qualunque i, $1 \leq i \leq n$, e qualunque $(n-1)$-tupla
$(s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n) \in$
$S_{1\omega} \times \ldots \times S_{(i-1)\omega} \times S_{(i+1)\omega} \times \ldots \times S_{n\omega}$
vale una delle seguenti due condizioni

- $g(s_1, \ldots, s_{i-1}, \omega, s_{i+1}, \ldots, s_n) = \omega$
- per ogni $s_i \in S_{i\omega}$:

  $g(s_1, \ldots, s_{i-1}, \omega, s_{i+1}, \ldots, s_n) =$
  $g(s_1, \ldots, s_{i-1}, s_i, s_{i+1}, \ldots, s_n)$

Interpretazione per linguaggi di tipo $\mathcal{L}_3$

Sia $\mathcal{F}$ una base per $\mathcal{L}_3$. Un'*interpretazione* di $\mathcal{F}$ è una coppia $\boldsymbol{\mathcal{I}} = (\boldsymbol{\mathcal{D}}, \boldsymbol{\mathcal{I}}_0)$ tale che $\boldsymbol{\mathcal{D}}$ è un insieme non vuoto detto *dominio*, e $\boldsymbol{\mathcal{I}}_0$ una funzione su $\mathcal{F}$ definita come segue

- Per ogni $c \in \mathcal{F}$, c di tipo b, $\boldsymbol{\mathcal{I}}_0(c) \in Bool$
- Per ogni $c \in \mathcal{F}$, c di tipo d, $\boldsymbol{\mathcal{I}}_0(c) \in \boldsymbol{\mathcal{D}}$
- Per ogni simbolo funzionale $f \in \mathcal{F}$ di tipo $(\beta_1, \ldots, \beta_n \rightarrow \beta_{n+1})$, dove $n \geq 1$ e $\beta_i \in \{b, d\}$, $\boldsymbol{\mathcal{I}}_0(f)$ è una funzione totale e monotona
  $\boldsymbol{\mathcal{I}}_0(f) : S_{1\omega} \times \ldots \times S_{n\omega} \rightarrow S_{(n+1)\omega}$
  dove $S_i = Bool$ se $\beta_i = b$ e $S_i = \boldsymbol{\mathcal{D}}$ se $\beta_i = d$, per $i = 1, \ldots, n+1$

Assegnamenti

Un assegnamento per una base $\mathcal{F}$ e un'interpretazione $\boldsymbol{\mathcal{I}} = (\boldsymbol{\mathcal{D}}, \boldsymbol{\mathcal{I}}_0)$ di $\mathcal{F}$ è una funzione che mappa ogni variabile $x \in V$ in un elemento di $\boldsymbol{\mathcal{D}}_\omega$ ed ogni variabile per funzione s-aria $F \in W$ in una funzione monotona di $(\boldsymbol{\mathcal{D}}_\omega^s \rightarrow_m \boldsymbol{\mathcal{D}}_\omega)$, $s \geq 1$

L'insieme di tutti gli assegnamenti viene indicato con $\Gamma_{\boldsymbol{\mathcal{I}}}$

Gli ordini parziali sul dominio $\boldsymbol{\mathcal{D}}_\omega$ e sulle funzioni monotone inducono un ordine parziale su Γ:

Per tutti i $\gamma, \gamma' \in \Gamma$, $\gamma \sqsubseteq \gamma'$ se e solo se $\gamma(x) \sqsubseteq \gamma'(x)$ per tutti gli $x \in V$, e $\gamma(F) \sqsubseteq \gamma'(F)$ per tutti gli $F \in W$

Semantica dei termini

Sia $\boldsymbol{\mathcal{I}} = (\boldsymbol{\mathcal{D}}, \boldsymbol{\mathcal{I}}_0)$ un'interpretazione della base $\mathcal{F}$. Il *funzionale semantico di questa interpretazione*, $\boldsymbol{\mathcal{I}}$, assegna ad ogni termine t di tipo b o di tipo d una funzione

- $\boldsymbol{\mathcal{I}}(t) : \Gamma \rightarrow Bool_\omega$
- $\boldsymbol{\mathcal{I}}(t) : \Gamma \rightarrow \boldsymbol{\mathcal{D}}_\omega$

rispettivamente.

Semantica dei termini (ctnd 1)

Queste funzioni vengono definite induttivamente

a. $\boldsymbol{\mathcal{I}}(c)(\gamma) = \boldsymbol{\mathcal{I}}_0(c)$, se $c \in \mathcal{F}$ è una costante
$\boldsymbol{\mathcal{I}}(x)(\gamma) = \gamma(x)$, se $x \in V$ è una variabile

b. $\boldsymbol{\mathcal{I}}(f(t_1, \ldots, t_s))(\gamma) = \boldsymbol{\mathcal{I}}_0(f)(\boldsymbol{\mathcal{I}}(t_1)(\gamma), \ldots, \boldsymbol{\mathcal{I}}(t_s)(\gamma))$, se
$f \in \mathcal{F}$ è un simbolo funzionale s-ario, $s \geq 1$
$\boldsymbol{\mathcal{I}}(F(t_1, \ldots, t_s))(\gamma) = \gamma(F)(\boldsymbol{\mathcal{I}}(t_1)(\gamma), \ldots, \boldsymbol{\mathcal{I}}(t_s)(\gamma))$, se
$f \in W$ è una variabile per funzione s-aria, $s \geq 1$

Semantica dei termini (ctnd 2)

Due termini t, t' vengono detti equivalenti nell'interpretazione $\boldsymbol{\mathcal{I}}$, se $\boldsymbol{\mathcal{I}}(t)(\gamma) = \boldsymbol{\mathcal{I}}(t')(\gamma)$ per tutti gli assegnamenti $\gamma \in \Gamma$

Teorema Sia t un termine su una base $\mathcal{F}$ ed $\boldsymbol{\mathcal{I}}$ una interpretazione di $\mathcal{F}$. Allora $\boldsymbol{\mathcal{I}}(t)$ è una funzione monotona

Semplificazione

Sostituzione di un termine con un altro termine equivalente più semplice

Le semplificazioni vengono eseguite con l'aiuto degli 'schemi di semplificazione'

Per poter definire gli schemi di semplificazione è necessario introdurre la nozione di *termini generalizzati* (viene aggiunto un insieme di variabili di tipo b)

Schemi e regole di semplificazione

Uno *schema di semplificazione* (per un base $\mathcal{F}$, ed un'interpretazione $\mathcal{I}$ di $\mathcal{F}$) è una coppia ordinata (t, t') di termini generalizzati tale che

- la lunghezza di t' è minore di quella di t
- t e t' sono equivalenti in $\mathcal{I}$

Una *regola di semplificazione* è un insieme di schemi di semplificazione

Schemi e regole di semplificazione (ctnd.)

L'*applicazione di uno schema di semplificazione* (t, u) su un termine v (non generalizzato) consiste nella sostituzione di un sottotermine $t^{t_1,\ldots,t_n}_{x_1,\ldots,x_n}$ di v con $u^{t_1,\ldots,t_n}_{x_1,\ldots,x_n}$ dove $x_1, \ldots, x_n$ sono le variabili che occorrono in t e $t_1, \ldots, t_n$ sono alcuni termini (non generalizzati) del tipo corrispondente

L'*applicazione di una regola di semplificazione* consiste nell'applicare uno dei suoi schemi. Scriviamo: $t \rightarrow_r u$

Con '$\rightarrow_r^*$' indichiamo la chiusura riflessiva e transitiva di '$\rightarrow_r$'

t è massimamente semplificato (rispetto ad r) se non c'è alcun termine u tale che $t \rightarrow_r u$

Regola di semplificazione standard

La *regola di semplificazione standard* è l'insieme di tutti gli schemi di semplificazione (per una base $\mathcal{F}$ ed un'interpretazione $\mathcal{I}$ di $\mathcal{F}$) della forma $(f(t_1, \ldots, t_n), c)$ dove

- f è un simbolo funzionale di arietà $n \geq 1$
- ogni t_i è una variabile o una costante (del tipo appropriato)
- nessuna variabile occorre in $f(t_1, \ldots, t_n)$ più di una volta
- c è una costante

Denotiamo la regola di semplificazione standard con $s_{\mathcal{I}}$ o semplicemente con s

Regola di semplificazione standard (ctnd.1)

Malgrado la semplicità della regola di semplificazione standard essa è potente abbastanza da poter esprimere la semantica del linguaggio di programmazione $\mathcal{L}_3$

Teorema
Sia $\mathcal{F}$ una base, $\mathcal{I}$ una interpretazione di $\mathcal{F}$, t un termine su $\mathcal{F}$, ed s la regola di semplificazione standard. Se t è equivalente ad una costante c (in $\mathcal{I}$), allora $t \rightarrow_s^* c$

Regola di semplificazione standard (ctnd.2)

Ingredienti per la dimostrazione:
Chiamiamo *assegnamento ω-dappertutto* (per una data base e una data interpretazione) l'elemento $\gamma_0 \in \Gamma$ definito

- $\gamma_0(x) = \omega$ per ogni $x \in V$ e
- $\gamma_0(F) =$ la funzione s-aria costante con valore ω, per tutte le variabili per funzione s-arie $F \in W$, $s \geq 1$

Lemma
Sia t un termine su una base $\mathcal{F}$ e γ_0 l'assegnamento ω-dappertutto per una interpretazione $\mathcal{I}$ di $\mathcal{F}$. Se t non è equivalente ad una costante, allora $\mathcal{I}(t)(\gamma_0) = \omega$

Sostituzione

La sostituzione (o unfolding) consiste nel rimpiazzare una funzione con la sua definizione. Siano

$$F_j(x_{j1}, \ldots, x_{js_j}) \Leftarrow t_j$$

per $1 \leq j \leq n$ le equazioni di un programma ricorsivo S su una base $\mathcal{F}$

La funzione di sostituzione per S, $\mathcal{S}$, è una funzione che ha come dominio l'insieme di tutti i termini su $\mathcal{F}$ ed è definita induttivamente

Sostituzione (ctnd. 1)

a. $\mathcal{S}(c) = c$ per tutte le costanti $c \in \mathcal{F}$

   $\mathcal{S}(x) = x$ per tutte le variabili $x \in V$

b. $\mathcal{S}(f(u_1, \ldots, u_s)) = f(\mathcal{S}(u_1), \ldots, \mathcal{S}(u_s))$, per tutti i simboli funzionali s-ari $f \in \mathcal{F}$ e termini $u_1, \ldots, u_s$

   $\mathcal{S}(F(u_1, \ldots, u_s)) = F(\mathcal{S}(u_1), \ldots, \mathcal{S}(u_s))$, per tutte le variabili per funzione $F \in W$, $F \neq F_1, \ldots, F_n$ e termini $u_1, \ldots, u_s$

   $\mathcal{S}(F_j(u_1, \ldots, u_{s_j})) = (t_j)^{\mathcal{S}(u_1),\ldots,\mathcal{S}(u_{s_j})}_{x_{j1},\ldots,x_{js_j}}$, per tutti i j, $1 \leq j \leq n$, e termini $u_1, \ldots, u_{s_j}$

Sostituzione (ctnd. 2)

Quando $t' = \mathcal{S}(t)$ si dice che il termine t' viene ottenuto dal termine t per sostituzione simultanea (rispetto al programma S)

Teorema di sostituzione
Siano $F_j(x_{j1}, \ldots, x_{js_j}) \Leftarrow t_j$, per $j = 1, \ldots, n$, le equazioni di un programma ricorsivo S sulla base $\mathcal{F}$, e sia $\mathcal{S}$ la funzione sostituzione per S. Inoltre, sia $\boldsymbol{\mathcal{I}} = (\boldsymbol{\mathcal{D}}, \boldsymbol{\mathcal{I}}_0)$ una interpretazione di $\mathcal{F}$, $\gamma \in \Gamma$ un assegnamento, e t un termine su $\mathcal{F}$. Allora

$$\boldsymbol{\mathcal{I}}(\mathcal{S}(t))(\gamma) = \boldsymbol{\mathcal{I}}(t)(\gamma[F_1/g_1], \ldots, [F_n/g_n])$$

dove per $j = 1, \ldots, n$, la funzione $g_j : \boldsymbol{\mathcal{D}}_\omega^{s_j} \to \boldsymbol{\mathcal{D}}_\omega$ viene definita da

$$g_j(d_1, \ldots, d_{s_j}) = \boldsymbol{\mathcal{I}}(t_j)(\gamma[x_{j1}/d_1], \ldots, [x_{js_j}/d_{s_j}])$$

Relazione di transizione

Sia $\mathcal{F}$ una base, $\boldsymbol{\mathcal{I}} = (\boldsymbol{\mathcal{D}}, \boldsymbol{\mathcal{I}}_0)$ una interpretazione di $\mathcal{F}$, r una regola di semplificazione almeno tanto potente quanto la regola di semplificazione standard ed S un programma ricorsivo su $\mathcal{F}$, inoltre sia $\mathcal{S}$ la funzione di sostituzione per S

Allora sull'insieme dei termini costruiti da $\mathcal{F}$ viene definita la relazione di transizione $\Rightarrow_r$:
Dati due termini t, t', diciamo che $t \Rightarrow_r t'$ se valgono le seguenti tre condizioni

1. t contiene almeno una variabile per funzione
2. $\mathcal{S}(t) \rightarrow_r^* t'$
3. t' è massimamente semplificato

Computazioni

Una *sequenza di computazione* per un programma ricorsivo S rispetto all'interpretazione $\mathcal{I}$ e alla regola di semplificazione r per il *vettore di input* $(d_1, \ldots, d_{s_k}) \in \mathcal{D}^{s_k}$ è una sequenza di termini (possibilmente infinita)

$$t_0, t_1, t_2, \ldots$$

tale che $t_0 = F_k(c_1, \ldots, c_{s_k})$ dove $\mathcal{I}_0(c_j) = d_j$ per $j = 1, \ldots, s_k$, e tale che per termini consecutivi nella sequenza, $t_i \Rightarrow_r t_{i+1}$

Una sequenza di computazione che è

- infinita, oppure
- finisce con un termine non contenente variabili per funzione

viene detta *computazione*

Semantica operazionale di $\mathcal{L}_3$

Sia S un programma ricorsivo su una base $\mathcal{F}$ con una variabile per funzione m-aria. Sia $\boldsymbol{\mathcal{I}} = (\boldsymbol{\mathcal{D}}, \boldsymbol{\mathcal{I}}_0)$ una interpretazione di $\mathcal{F}$. Il *significato* di S (in $\boldsymbol{\mathcal{I}}$) è la funzione $\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S) : \boldsymbol{\mathcal{D}}^m \rightsquigarrow \boldsymbol{\mathcal{D}}$ definita da

$$\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(d_1, \ldots, d_m) = \begin{cases} d & \text{se, su input } d_1, \ldots, d_m, \\ & S \text{ termina con output } d \\ \text{indefinita} & \text{altrimenti} \end{cases}$$

Definizione di insiemi per induzione simultanea

Sia $\mathcal{U}$ l'universo del discorso e
$B_1, \ldots, B_n \subseteq \mathcal{U}$, $n \geq 1$ siano gli insiemi base

Per ogni tipo $\tau = (\beta_1, \ldots, \beta_k \rightarrow \beta_{k+1})$,
dove i $\beta_i \in \{1, \ldots, n\}$, per $i = 1, \ldots, k+1$ e $k \geq 1$,
sia $\mathcal{K}_\tau$ un insieme di relazioni $r \subseteq \mathcal{U}^k \times \mathcal{U}$ chiamati i costruttori
di tipo τ

Definizione di insiemi per induzione simultanea(ctnd.)

Gli insiemi $A_1, \ldots, A_n$ vengono definiti per *induzione strutturale simultanea* mediante $B_1, \ldots, B_n$ e gli insiemi di costruttori $\mathcal{K}_\tau$, se

$(A_1, \ldots, A_n)$ è la più piccola fra le $(S_1, \ldots, S_n) \subseteq \mathcal{U}^n$ per cui valgono le seguenti due condizioni

- $(B_1, \ldots, B_n) \subseteq (S_1, \ldots, S_n)$
- $r(S_{\beta_1} \times \ldots \times S_{\beta_k}) \subseteq S_{\beta_{k+1}}$ per ogni $r \in \mathcal{K}_\tau$ con $\tau = (\beta_1, \ldots, \beta_k \rightarrow \beta_{k+1})$