Definizione 17 *Sia $s_1,\ldots,s_t$ una computazione finita eseguita da una macchina a registri elementare con programma π. Tale computazione puó essere codificata tramite un intero $c = 2^{s_1} \cdot 3^{s_2} \ldots \cdot p_t^{s_t}$.*

Esempio 18 *La computazione del programma della somma è codificata nel seguente modo:*

$2^{(2\cdot 3^{72})} \cdot 3^{(2^2\cdot 3^{72})} \cdot 5^{(2^3\cdot 3^{24})} \cdot \ldots \cdot 29^{(3^{32})}$

Lemma 19 *Dato un programma π, la funzione*

$$T_\pi(x_1,\ldots,x_n,c) = \begin{cases} 1 \text{ se } c = \langle s_1,\ldots,s_F\rangle \text{ e } s_1,\ldots,s_F \text{ è la computazione di } \pi \\ \text{con input } x_1,\ldots,x_n \\ 0 \text{ altrimenti.} \end{cases}$$

è ricorsiva primitiva.

Dimostrazione. Introduciamo le seguenti funzioni in {0,1}:
• $I_\pi(x_1,\ldots,x_n,s) = 1$ se e solo se $s = \langle 1, \langle x_1,\ldots,x_n\rangle\rangle$
verifica che la computazione c inizi in uno stato iniziale.
• $F_\pi(s) = 1$ se e solo se $(s)_1 = 0$
verifica che la computazione termini in uno stato finale

Definiamo $R_\pi(s_p, s_q) = 1$ con $s_p = \langle i_p, \langle r_1^p \ldots, r_m^p\rangle\rangle$ e $s_q = \langle i_q, \langle r_1^q \ldots, r_m^q\rangle\rangle$ se e solo se:

a) se i_p corrisponde all'istruzione $\mathrm{R}j = \mathrm{R}j + 1$

$$\text{allora} \quad i_q = i_p + 1 \quad \text{ed} \quad r_k^q = \begin{cases} r_k^p + 1 & \text{se } k = j \\ r_k^p & \text{altrimenti} \end{cases}$$

b) se i_p corrisponde all'istruzione $\mathrm{R}j = \mathrm{R}j \dot{-} 1$

$$\text{allora} \quad i_q = i_p + 1 \quad \text{ed} \quad r_k^q = \begin{cases} r_k^p \dot{-} 1 & \text{se } k = j \\ r_k^p & \text{altrimenti} \end{cases}$$

c) se i_s : IF $\mathrm{R}j = 0$ GOTO h

$$\text{allora} \quad i_q = \begin{cases} h & \text{se } r_j^p = 0 \\ i_p + 1 & \text{altrimenti} \end{cases} \qquad \text{ed} \quad r_k^q = r_k^p \quad \text{per ogni } k.$$

d) Se i_p è un numero maggiore del numero di istruzione del programma allora $i_q = i_p$ ed $r_k^q = r_k^p$ per ogni k.
(si noti che questo ultimo caso comporta l'esecuzione di un ciclo infinito).

Possiamo definire

$$T_\pi(x_1,\ldots,x_n,c) = I_\pi(x_1,\ldots,x_n,(c)_1)\cdot\prod_{p=1}^{M(c)-1} R_\pi((c)_p,(c)_p+1)\cdot F_\pi((c)_{M(c)})$$

cioè si controlla che tutti gli esponenti della decomposizione di c in primi costituiscano una sequenza di calcolo.

Essendo ricorsive primitive le funzioni $\prod$, M, P_n, K_i segue che T_π è ricorsiva primitiva.

Teorema 20 *Data una macchina a registri elementare con programma π, la funzione $\lambda x_1,\ldots\lambda x_n.f(x_1,\ldots x_n)$ calcolata da π è una funzione ricorsiva parziale.*

Dimostrazione. Possiamo definire la funzione f nel seguente modo:

$$f(x_1,\ldots,x_n) = U(\mu c[T_\pi(x_1,\ldots,x_n,c) = 1])$$

dove $U(c) = (((c)_{M(c)})_2)_1$ è la funzione che estrae $s_F = (c)_{M(c)}$ da c ed estrae x_1^F da s_F.

Funzioni ricorsive e linguaggi funzionali

La teoria delle funzioni ricorsive ha avuto un impatto notevole sulla definizione e lo sviluppo del paradigma di *programmazione funzionale*.

I linguaggi funzionali permettono di definire funzioni senza far riferimento al modello di macchina che le calcola (come per le funzioni ricorsive).

valutazione di una espressione, invece che *esecuzione di una computazione*.

Un semplice linguaggio funzionale (formalismo di McCarthy)

$X = \{x_1, x_2, \ldots\}$ insieme *di simboli di variabile*;

$B = \{b_1, b_2, \ldots\}$ insieme *di simboli di funzione base*;

$F = \{F_1, F_2, \ldots\}$ insieme *di simboli di variabile funzione*;

$a : \mathbb{N}^+ \longmapsto \mathbb{N}$ aritá delle funzioni in B;

$A : \mathbb{N}^+ \longmapsto \mathbb{N}^+$ aritá delle funzioni in F.

Definizione 21 *Un* termine *su* $< X, B, F >$ *è definito induttivamente come segue:*

1. se $x \in X$ *allora* x *è un termine;*

2. se $b_j \in B$*, con aritá* $a(j) = n \geq 0$*, e* $t_1, \ldots, t_n$ *sono termini, allora* $b(t_1, \ldots, t_n)$ *è un termine;*

3. se $F_i \in F$*, con aritá* $A(i) = n > 0$*, e* $t_1, \ldots, t_n$ *sono termini, allora* $F_i(t_1, \ldots, t_n)$ *è un termine.*

4. se la aritá di una funzione base $b_k()$ *è* 0*, allora la funzione è detta costante, e viene rappresentata come* b_k*.*

Definizione 22 *Dati tre insiemi* X, B, F*, uno* schema di programma *sulla terna* $< X, B, F >$ *è costituito da una sequenza finita di coppie di termini del tipo*

$$F_i(x_1, \ldots, x_n) \Longleftarrow t_j$$

dove $F_i \in F$ *e* t_j *è un generico termine su* $< X, B, F >$*, seguita da un temine del tipo*

$$F_h(b_1, \ldots, b_m)$$

dove $F_h \in F$ *e* $A(h) = m$ *e* $b_1, \ldots, b_m$ *sono costanti.*

L'interpretazione di uno schema di programma vede la sequenza di coppie come una sequenza di dichiarazioni di funzioni, e il termine contenente la costante b è la richiesta di determinare il valore di F_h sugli argomenti $b_1, \ldots, b_m$.

Esempio 23
$b_1 \rightarrow$ *arità 3*
$b_2 \rightarrow$ *arità 1*
$b_3 \rightarrow$ *arità 1*
$b_2 \rightarrow$ *arità 0*
$F_1 \rightarrow$ *arità 2*
$F_2 \rightarrow$ *arità 1*
Un esempio di schema di programma su $X = \{x_1, x_2\}$, $B = \{b_1, b_2, b_3, b_4\}$ *e* $F = \{F_1, F_2\}$ *è il seguente:*

$F_1(x_1, x_2) \Leftarrow b_1(x_1, F_2(b_2(x_2)), x_1)$
$F_2(x_1) \Leftarrow b_2(F_1(x_1, b_3(x_1)))$
$F_2(b_4)$

Se un insieme di funzioni base è messo in corrispondenza con i simboli di B, allora otteniamo un linguaggio di programmazione funzionale.
Il linguaggio SLF, per la definizione di funzioni intere è definito come segue:

$B = \mathcal{N} \cup \{S, P, if - then - else\}$.

$if - then - else(t_1, t_2, t_3) = t_2$ se il valore di t_1 è 0, t_3 altrimenti.

Esempio 24 *Un programma per il calcolo della sottrazione* $4-3$ *può essere formulato in SLF come segue:*

$SUB(x, y) \Leftarrow if - then - else(y, x, P(SUB(x, P(y))))$
$SUB(4,3)$

Teorema 25 *Il linguaggio SLF consente la definizione di tutte le funzioni ricorsive.*

Dimostrazione. Le funzioni base possono essere espresse in SLF. Il successore è dato. Le funzioni $0_i^{(n)}$ possono essere definite come:
$F(x_1,\ldots,x_n) \Leftarrow 0$
Le funzioni selettive $U_i^{(n)}$ possono essere definite così:
$F(x_1,\ldots,x_n) \Leftarrow x_i$

Se F è definita per composizione di h, $g_1,\ldots,g_m$, essa può essere definita in SLF così:
$F(x_1,\ldots,x_n) \Leftarrow h(g_1(x_1,\ldots,x_n),\ldots,g_m(x_1,\ldots,x_n))$

Se F è definita per recursione primitiva a partire da g e h , essa può essere definita in SLF così:
$F(x_1,\ldots,x_n,y) \Leftarrow$
$if-then-else(y,g(x_1,\ldots,x_n),h(x_1,\ldots,x_n,y,F(x_1,\ldots,x_n,P(y))))$

Se F è definita per minimalizzazione a partire da g, essa può essere definita in SLF così:
$F(x_1,\ldots,x_n) \Leftarrow G(x_1,\ldots,x_n,0)$
$G(x_1,\ldots,x_n,t) \Leftarrow if-then-else(g(x_1,\ldots,x_n,t),t,G(x_1,\ldots,x_n,S(t)))$