PREDICATO STP

Consideriamo il predicato $STP^{(n)}(x_1, \dots, x_n, y, t) \Leftrightarrow$ il programma numero y si ferma sulle variabili di ingresso $x_1, \dots, x_n$ dopo al più t passi.
Il teorema del programma universale ci permette di dimostrare che i predicati $STP^{(n)}(x_1, \dots, x_n, y, t)$ sono calcolabili. Basta infatti modificare il programma universale in modo che conti i passi di calcolo compiuti dal programma e ci informi se il programma si è fermato o meno al passo t.

Programma che calcola $Y = STP^{(n)}(x_1, \dots, x_n, x_{n+1}, x_{n+2})$:

1. $Z \leftarrow X_{n+1} + 1$
2. $S \leftarrow \Pi_{n \le i \le 1} (p_{2i})^{Xi}$
3. $K \leftarrow 1$
4. [C] $Q \leftarrow Q+1$
5. IF $Q > X_{n+2} + 1$ GOTO E
6. IF $K = Lt(Z)+1$ v $K = 0$ GOTO F
7. $U \leftarrow r((Z)_K)$
8. $P \leftarrow p_{r(U)+1}$
9. IF $l(U) = 0$ GOTO N
10. IF $l(U) = 1$ GOTO A
11. IF $\sim(P|S)$ GOTO N
12. IF $l(U) = 2$ GOTO M
13. $K \leftarrow \min_{i \le Lt(Z)} [l((Z)_i) = l(U)-2]$
14. GOTO C
15. [M] $S \leftarrow \llcorner S/P \lrcorner$
16. GOTO N
17. [A] $S \leftarrow S \cdot P$
18. [N] $K \leftarrow K+1$
19. GOTO C
20. [F] $Y \leftarrow 1$

È uguale al programma universale tranne per le istruzioni 4 e 5 e per l'istruzione finale. L'istruzione 4 incrementa un contatore Q, mentre l'istruzione 5 controlla se il valore di Q è maggiore di t che è memorizzato nella variabile X_{n+2}. Se il programma termina entro i t passi allora viene assegnato 1 a Y, altrimenti Y rimane 0 ed esce con l'istruzione 5.

INSIEMI RICORSIVAMENTE ENUMERABILI

Dire che un insieme B appartiene ad una qualche classe di funzioni è equivalente a dire che il predicato $P(x) \Leftrightarrow x \in B$ appartiene alla classe di funzioni detta.
La relazione tra B e P(x) è data da $B = \{ x \in N \mid P(x) \}$.
Dire che l'insieme B è calcolabile (o ricorsivo) equivale a dire che P(x) è una funzione calcolabile.
Dire che l'insieme B è ricorsivo primitivo equivale a dire che P(x) è una funzione ricorsiva primitiva.
DEFINIZIONE: Un insieme $B \subseteq N$ si dice **ricorsivamente enumerabile** se esiste una funzione parzialmente calcolabile g(x) tale che $B = \{ x \in N \mid g(x) \text{ è definita} \}$.
Se P è un programma che calcola la funzione g, allora possiamo dire che B è l'insieme di tutti i valori d'ingresso di P per i quali P prima o poi si ferma. Il programma P ci aiuta solo parzialmente per verificare se un certo elemento x appartiene o no a B: se x appartiene a B allora il programma, prima o poi, si ferma ; ma se x non appartiene a B il programma continuerà a girare per sempre, e noi non possiamo sapere se ha ancora bisogno di tempo per terminare il calcolo oppure se è entrato in un ciclo infinito e non si fermerà mai.

PROPOSIZIONE: Se l'insieme B è ricorsivo allora è ricorsivamente enumerabile.
DIM: Consideriamo il programma P: [A] IF $\sim(X \in B)$ GOTO A
P è calcolabile in S, perché essendo B ricorsivo per ipotesi, allora $X \in B$ è un predicato calcolabile in S.
Sia h la funzione calcolata da P; h è definita solo sugli elementi di B, quindi si ha che:
$B = \{ x \in N \mid h(x) \text{ è definita} \}$. Poiché questa è proprio la definizione di insieme ricorsivamente enumerabile essendo h parzialmente calcolabile, allora la proposizione è dimostrata.

TEOREMA (di Post): L'insieme B è ricorsivo se e solo se sia B sia B^C sono ricorsivamente enumerabili.
DIM ($\Rightarrow$): Se B è ricorsivo allora anche B^C lo è; applicando la proposizione precedente a entrambi gli insieme allora otteniamo che sia B^C sia B sono ricorsivamente enumerabili.
DIM ($\Leftarrow$): Sia B sia B^C sono ricorsivamente enumerabili, quindi possiamo scrivere che
$B = \{ x \in N \mid g(x) \text{ è definita} \}$ $B^C = \{ x \in N \mid h(x) \text{ è definita} \}$
con g ed h funzioni parzialmente calcolabili.
Sia g calcolata dal programma P, sia h calcolata dal programma Q e siano p = #(P) e q = #(Q).
Possiamo costruire il programma che calcola la funzione caratteristica di B, dimostrando così che l'insieme B è ricorsivo:

1. [A] IF $STP^{(1)}$ (X, p, T) GOTO C
2. IF $STP^{(1)}$ (X, q, T) GOTO E
3. T ← T+1
4. GOTO A
5. [C] Y ← 1

Questo programma usa il predicato STP (che è calcolabile) ed esegue, alternandoli, i passi dei programmi che calcolano g e h; prima o poi uno dei due programmi termina e quindi possiamo sapere se X appartiene a B (in questo caso Y = 1) oppure X non appartiene a B (Y = 0).

Sappiamo che per ogni $n > 0$ abbiamo che la funzione $\Phi^{(n)}(x_1, \dots, x_n, y) = \Psi_P^{(n)}(x_1, \dots, x_n)$, con $y = \#(P)$ è parzialmente calcolabile. Fissato n, consideriamo la successione $\Phi^{(n)}(x_1, \dots, x_n, 0)$, $\Phi^{(n)}(x_1, \dots, x_n, 1)$, … che può anche scriversi come $\Phi_0^{(n)}(x_1, \dots, x_n)$, $\Phi_1^{(n)}(x_1, \dots, x_n)$, … :
tale successione enumera tutte le funzioni parzialmente calcolabili di n variabili.
Consideriamo adesso il caso di una sola variabile: $\Phi_0(x)$, $\Phi_1(x)$, …
Sia $W_n = \{ x \in N \mid \Phi_n(x) \text{ è definita} \}$; W_n è un insieme ricorsivamente enumerabile.
Sia $K = \{ n \in N \mid n \in W_n \}$ allora si ha che $n \in W_n \Leftrightarrow \Phi_n(n)$ è definita $\Leftrightarrow$ HALT(n, n), cioè K è l'insieme di tutti i numeri n tali che il programma numero n si fermerà, prima o poi, sull'ingresso n.
TEOREMA: K è ricorsivamente enumerabile ma non ricorsivo.
DIM (per assurdo): $K = \{ n \in N \mid n \in W_n \} = \{ n \in N \mid \Phi_n(n) \text{ è definita} \}$. Poiché $\Phi_n(n)$ è parzialmente calcolabile allora K è ricorsivamente enumerabile. Se K è ricorsivo, allora anche K^C deve essere ricorsivamente enumerabile. Supponiamo che lo sia, allora deve esistere un indice i tale che $W_i = K^C$.
Abbiamo che $i \in K \Leftrightarrow i \in W_i$, ma poiché $W_i = K^C$ abbiamo che $i \in K \Leftrightarrow i \in K^C$, contraddizione.
Ne segue che K^C non può essere ricorsivamente enumerabile e quindi K non è ricorsivo.
DIM (più immediata): $K = \{ n \in N \mid n \in W_n \} = \{ n \in N \mid \Phi_n(n) \text{ è definita} \} = \{ n \in N \mid \text{HALT}(n, n) \}$. Dire che K è ricorsivo equivale a dire che il predicato di cui è l'estensione è ricorsivo. Ma il predicato HALT non è ricorsivo, quindi K non può essere ricorsivo.

TEOREMA (di ricorsione): Sia $g(z, x_1, \dots, x_m)$ una funzione parzialmente calcolabile di m+1 variabili. Allora esiste un numero e tale che: $\Phi_e^{(m)}(x_1, \dots, x_m) = g(e, x_1, \dots, x_m)$.
Detto in altri termini: data una funzione g parzialmente calcolabile, un valore di una delle variabili è il numero di Gödel di un programma che calcola g come funzione delle rimanenti variabili.

PROPOSIZIONE (programma che si autoriproduce): Esiste un numero e tale che $\Phi_e(x) = e$ per ogni x.
Qualunque sia la variabile di input x, il programma numero e usa questa informazione per ricalcolare se stesso, cioè il suo numero di codice.

TEOREMA (del punto fisso): Sia f(x) una funzione calcolabile, allora esiste un numero e tale che $\Phi_{f(e)}(x) = \Phi_e(x)$
Detto in altri termini: per qualsiasi funzione calcolabile esiste un numero e tale che sia e sia f(e) sono numeri di programmi che calcolano la stessa funzione Φ.

Sia T un insieme di funzioni parzialmente calcolabili di una sola variabile.
Associamo a T l'insieme di indici $R_T = \{ t \in N \mid \Phi_t \in T \}$.
TEOREMA (di Rice): Sia T un insieme di funzioni parzialmente calcolabili di una sola variabile. Siano f(x) e g(x) due funzioni parzialmente calcolabili tali che $f(x) \in T$ e $g(x) \notin T$. Allora R_T non è ricorsivo.
È impossibile data un'arbitraria definizione di T trovare gli indici che appartengono a R_T.

Consideriamo una funzione a k variabili $\Phi^{(k)}(x_1, \dots, x_k, y)$, dove y è il numero di codice del programma. Esistono delle situazioni nelle quali vorremmo presentare la $\Phi^{(k)}$ come funzione ad esempio di una sola variabile calcolata da un programma opportuno. Questo programma dovrebbe essere ottenibile, in modo calcolabile, a partire dal programma che calcola $\Phi^{(k)}$ e dalle variabili che non si vogliono considerare. Il **teorema del parametro** non solo ci permette di fare ciò, ma afferma anche che la funzione che ci fornisce il numero di Gödel del nuovo programma è ricorsiva primitiva.
Per esempio una funzione di m+1 variabili con un codice di Gödel y, può essere trasformata in una funzione di m variabili con un nuovo codice di Gödel che incorpora la variabile che abbiamo tolto. Si può inserire nel programma una costante che corrisponde alla variabile che togliamo. Per farlo

dobbiamo avere una funzione calcolabile che prende il valore della variabile e modifica opportunamente il codice del programma in modo che quella variabile diventi una specie di costante inserita all'interno del programma. Basta modificare il programma in modo che prima di essere eseguito crei una nuova variabile x e dare a questa variabile il valore della variabile che abbiamo omesso. In questo modo abbiamo creato una funzione di m variabili in cui il nuovo codice del programma rappresenta il programma precedente più le istruzioni per la creazione della nuova variabile.

TEOREMA: Per ciascun n, m > 0 esiste una funzione ricorsiva primitiva $S_m^{(n)}(u_1, \dots, u_n, y)$ tale che $\Phi^{(m+n)}(x_1, \dots, x_m, u_1, \dots, u_n, y) = \Phi^{(m)}(x_1, \dots, x_m, S_m^{(n)}(u_1, \dots, u_n, y))$ dove l'ultimo parametro è il codice del programma.
DIM (per induzione su n): Per n = 1 dobbiamo mostrare che esiste una funzione ricorsiva primitiva $S_m^{(1)}(u, y)$ tale che $\Phi^{(m+1)}(x_1, \dots, x_m, u, y) = \Phi^{(m)}(x_1, \dots, x_m, S_m^{(1)}(u, y))$. Sia y = #(P) allora possiamo interpretare $S_m^{(1)}(u, y)$ come il numero di un programma che prima assegna il valore u alla variabile x_{m+1} e successivamente esegue il programma P.
Programma che assegna u ad X_{m+1}:

$X_{m+1} \leftarrow X_{m+1} + 1$
$X_{m+1} \leftarrow X_{m+1} + 1$
… … … … … … …
(u volte)

Calcoliamo il numero dell'istruzione non etichettata $X_{m+1} \leftarrow X_{m+1} + 1$:
a = 0 perché non c'è etichetta;
b = 1 perché l'istruzione è di incremento
c = #(V)-1 = #(X_{m+1})-1 = 2(m+1)-1 = 2m+1
NOTA: #(X_{m+1}) = 2(m+1) perché la X_{m+1} è la (m+1)-esima variabile di ingresso e tali variabili occupano sempre posto pari nell'ordinamento dato.
Quindi abbiamo #(I) = <a, <b, c>> = <0, <1, 2m+1>>
<b, c> = $2^1(2(2m+1)+1)-1 = 2(4m+3)-1 = 8m+5$
<a, <b, c>> = $2^0(2(8m+5)+1)-1 = 16m+10$

$S_m^{(1)}(u, y)$ è il numero del nuovo programma, mentre y è il numero del programma originario P.
y = #(P) = [#(I_1), … , #(I_q)]-1 q = Lt([#(I_1), … , #(I_q)]) = Lt(y+1)
L'esponente da dare al numero primo p_j è dato da $([\#(I_1), \dots, \#(I_q)])_j = (y+1)_j$.
Quindi abbiamo che $y = \Pi_{1 \le j \le q} p_j^{(y+1)_j}$.
$S_m^{(1)}(u, y)$ è il numero di codice del programma che si ottiene premettendo le u istruzioni $X_{m+1} \leftarrow X_{m+1} + 1$ al programma P. Per trovare il numero di codice del nuovo programma dobbiamo quindi traslare di u unità i numeri primi p_j che compaiono in P e calcolare il codice delle u istruzioni. Il codice delle u istruzioni $X_{m+1} \leftarrow X_{m+1} + 1$ è dato da $\Pi_{1 \le i \le u} p_i^{(16m+10)}$ e quindi si ha che:
$S_m^{(1)}(u, y) = [\Pi_{1 \le i \le u} p_i^{(16m+10)} \cdot \Pi_{1 \le j \le q} p_{u+j}^{(y+1)_j}] - 1$.
È evidente che $S_m^{(1)}$ è una funzione ricorsiva primitiva.

Supponiamo il risultato valido per n = k e dimostriamolo per n = k+1. Si ha che:
$\Phi^{(m+k+1)}(x_1, \dots, x_m, u_1, \dots, u_k, u_{k+1}, y) =$
$\Phi^{(m+k)}(x_1, \dots, x_m, u_1, \dots, u_k, S_{m+k}^{(1)}(u_{k+1}, y)) =$
$\Phi^{(m)}(x_1, \dots, x_m, S_m^{(k)}(u_1, \dots, u_k, S_{m+k}^{(1)}(u_{k+1}, y)))$
Se adesso poniamo $S_m^{(k+1)}(u_1, \dots, u_k, u_{k+1}, y) = S_m^{(k)}(u_1, \dots, u_k, S_m^{(1)}(u_{k+1}, y))$ il risultato risulta dimostrato.