Associamo a ciascun programma P del linguaggio $\mathcal{S}$ un numero, che indicheremo con $\#(P)$, in modo tale che il programma può essere integralmente ricostruito a partire da $\#(P)$.

Ordiniamo le variabili come segue

$Y, X_1, Z_1, X_2, Z_2, \ldots$

e le etichette:

$A_1, B_1, C_1, D_1, E_1, A_2, B_2, C_2, \ldots$

Scriviamo $\#(V)$, $\#(L)$ per la posizione di una variabile o di un'etichetta data nell'ordinamento appropriato.

Avremo quindi:

$\#(X_2) = 4$, $\#(Z_1) = \#(Z) = 3$,

$\#(E) = 5$, $\#(B_2) = 7$

Sia ora I un'istruzione (etichettata o non etichettata) del linguaggio $\mathcal{S}$.

Scriviamo allora

$$\#(I) = \langle a, \langle b, c \rangle \rangle$$

dove:

1. Se I <u>non</u> è etichettata allora $a = 0$
   Se I è etichettata L allora $a = \#(L)$

2. Se la variabile V compare in I
   allora $c = \#(V) - 1$

3. Se l'enunciato in I è
   $V \leftarrow V$ oppure $V \leftarrow V+1$ oppure $V \leftarrow V-1$
   allora, rispettivamente, $b = 0$ oppure $b = 1$ oppure $b = 2$.

4. Se l'enunciato in I è
   IF $V \neq 0$ GOTO L'
   allora $b = \#(L') + 2$

ESEMPI

Programma

$$[A] \quad X \leftarrow X+1 \qquad I_1$$
$$\text{IF } X \neq 0 \text{ GOTO } A \qquad I_2$$

(calcola la funzione (calcolabile in $\mathcal{S}$) non definita in nessun punto).

calcoliamo il numero di codice dell'istruzione I_1

abbiamo che:

$$\langle a, \langle b, c \rangle \rangle$$
$$\#(I_1) = \langle 1, \langle 1, 1 \rangle \rangle$$

perché l'etichetta A ha posto 1

perché l'istruzione è del tipo $X \leftarrow X+1$

perché la variabile che compare è X che si trova al secondo posto ed il valore di c è dato da: $c = \#(V) - 1$

effettuando il calcolo si ha:

$$\langle 1, 1 \rangle = 2^1 (2 \cdot 1 + 1) - 1 = 5$$

$$\langle 1, 5 \rangle = 2^1 (2 \cdot 5 + 1) - 1 = 21$$

quindi $\#(I_1) = 21$

$$\left(\langle x, y \rangle = 2^x (2y + 1) - 1 \right)$$

I_2 : IF $X \neq 0$ GOTO A

$$\#(I_2) = \langle a, \langle b, c \rangle\rangle =$$

$$= \langle 0, \langle 3, 1 \rangle\rangle$$

perché l'istruzione NON è etichettata

essendo l'istruzione di "salto" il suo numero è $b = \#(L) + 2 = 1 + 2 = 3$ poiché $\#(A) = 1$

perché la variabile che compare è X

effettuando il calcolo si trova che:

$$\#(I_2) = 46$$

poiché:

$$\langle 3, 1 \rangle = 2^3(2 \cdot 1 + 1) - 1 = 8(2+1) - 1 = 23$$

e $\langle 0, 23 \rangle = 2^0(2 \cdot 23 + 1) - 1 = 46$.

Il numero del programma è quindi dato da:

$$\#(P) = [\#(I_1), \#(I_2)] - 1 = [21, 46] - 1 =$$

$$= 2^{21} \cdot 3^{46} - 1$$

Si osservi che per ogni numero dato q vi è un'unica istruzione I con $\#(I) = q$. Calcoliamo prima $l(q)$. Se $l(q) = 0$, I è non etichettata; in caso contrario I ha la $l(q)$-esima etichetta della nostra lista. Per trovare la variabile che compare in I, calcoliamo $i = r(r(q)) + 1$ e localizziamo la i-esima variabile V nella nostra lista. L'enunciato in I sarà quindi

$V \leftarrow V$	se	$l(r(q)) = 0$,
$V \leftarrow V + 1$	se	$l(r(q)) = 1$,
$V \leftarrow V - 1$	se	$l(r(q)) = 2$,
IF $V \neq 0$ GOTO L	se	$j = l(r(q)) - 2 > 0$

ed L è la j-esima etichetta nella nostra lista.

Infine, un programma P consista delle istruzioni $I_1, I_2, \ldots, I_k$. Poniamo allora

$$\#(P) = [\#(I_1), \#(I_2), \ldots, \#(I_k)] - 1.$$

$$\langle a, \langle b, c \rangle \rangle = q$$

Calcoliamo il numero di codice dell' istruzione non etichettata:

$$Y \leftarrow Y \qquad (*)$$

Ricordando che

- Se I non è etichettata $\qquad a = 0$
- Se I è del tipo $V \leftarrow V$ $\qquad b = 0$
- Se la variabile nominata in I è Y allora $\qquad c = 0$

$\left(c = \#(Y) - 1 = \#(Y) - 1 = 1 - 1 = 0\right)$

il numero associato all' istruzione $(*)$ è $\langle 0, \langle 0, 0 \rangle\rangle = 0$

<u>Ricordiamo</u>:

Unica <u>ambiguità</u> codifiche di Gödel, gli zeri alla fine -

<u>Ambiguità eliminabile</u>:

Decretando che nessun programma può terminare con l'istruzione $Y \leftarrow Y$.

G1

TEOREMA DELLA FERMATA

Per un y dato, sia ora P il programma tale che $\#(P) = y$ allora HALT(x,y) *è vero se* $\Psi_P^{(1)}(x)$ è definito e falso se $\Psi_P^{(1)}(x)$ non è definito. In breve:

HALT(x,y) $\Longleftrightarrow$ il programma numero y si fermerà, prima o poi, sull'ingresso x.

Teorema . HALT(x,y) non è un predicato computabile.

Dimostrazione. Supponiamo che HALT(x,y) sia computabile. Potremmo allora costruire il programma $\mathcal{Q}$:

[A] IF HALT(X,X) GOTO A

(Naturalmente $\mathcal{Q}$ deve essere la macro espansione di questo programma.)
Ovviamente $\mathcal{Q}$ è stato costruito in modo tale che:

$$\Psi_{\mathcal{Q}}^{(1)}(x) = \begin{cases} \text{indefinito} & \text{se} \quad \mathrm{HALT}(x,x) \\ 0 & \text{se} \quad \sim \mathrm{HALT}(x,x). \end{cases}$$

Sia $\#(\mathcal{Q}) = y_0$. Usando allora la definizione del predicato HALT,

$$\mathrm{HALT}(x,y_0) \Longleftrightarrow \sim \mathrm{HALT}(x,x).$$

Poichè quest'equivalenza è vera per ogni x, possiamo porre $x = y_0$:

$$\mathrm{HALT}(y_0,y_0) \Longleftrightarrow \sim \mathrm{HALT}(y_0,y_0).$$

Ma questa è una contraddizione.
Ed il teorema è dimostrato-

Questo teorema ci fornisce, intanto, un esempio di funzione che non è computabile mediante nessun programma del linguaggio S.

Ma possiamo dire dippiù:

Dato un programma di S ed un ingresso a tale programma, non esiste nessun algoritmo in grado di determinare se il programma dato si fermerà o no, prima o poi, sull'ingresso dato.

Sotto questa forma il risultato è chiamato *la insolubilità del problema della fermata.*

Infatti, se un tale algoritmo esistesse, potremmo usarlo per verificare la verità o falsità di HALT(x,y) per x ed y dati, ottenendo prima il programma Q, con #(Q) = y, e verificando quindi se Q, prima o poi, si ferma sull'ingresso x.

Chiediamoci adesso se il predicato $HALT(x,x)$ di una sola variabile è calcolabile:

Supponiamo che lo sia. Allora come nel caso precedente potremmo costruire il programma $\mathcal{P}$

$$[A] \quad \text{IF } HALT(X,X) \text{ GOTO } A$$

e sia, come prima, $y_0 = \#(\mathcal{P})$

Poniamoci adesso la domanda:

$\mathcal{P}$ si ferma sulla variabile d'ingresso y_0?

Avremmo:

$$HALT(y_0, y_0) \Longleftrightarrow \sim HALT(y_0, y_0)$$

Ciò mostra che in realtà la dimostrazione precedente della non calcolabilità di $HALT(x,y)$ è, in nuce, la dimostrazione della non calcolabilità di $HALT(x,x)$.

CONGETTURA DI GOLDBACH

OGNI NUMERO PARI ≥ 4 È LA SOMMA DI DUE NUMERI PRIMI

È facile verificarla per numeri piccol

$$4 = 2+2$$
$$6 = 3+3$$
$$20 = 7+13$$
$$36 = 17+19$$

È possibile scrivere un programma che verifica la congettura cercando controesempi.
Il programma si ferma quando ha trovato un n pari che non soddisfa la congettura.

Collegamento col teorema della fermata

Cosa potremmo dire se il teorema della fermata <u>non</u> fosse vero?

PROGRAMMI UNIVERSALI

Sia $\mathcal{P}$ un programma arbitrario tale che $\#(\mathcal{P})=y$ e sia $\psi_{\mathcal{P}}^{(n)}(x_1,\dots,x_n)$ la funzione (parzialmente calcolabile) calcolata da $\mathcal{P}$.

Per ogni $n>0$, definiamo:

$$\Phi^{(n)}(x_1,\dots,x_n,y) = \psi_{\mathcal{P}}^{(n)}(x_1,\dots,x_n)\text{, con } y=\#(\mathcal{P})$$

TEOREMA

Per ciascun $n>0$, la funzione $\Phi^{(n)}(x_1,\dots,x_n,y)$ è PARZIALMENTE CALCOLABILE.

Osservazioni

L'esistenza di un tale teorema, che è uno dei risultati centrali della teoria della calcolabilità, non era affatto scontata. Ciò che ci dice è abbastanza sorprendente: PER OGNI n, TUTTE LE FUNZIONI DI N VARIABILI SONO CALCOLABILI MEDIANTE UN UNICO PROGRAMMA.
(Ricordando poi che dal punto di vista della calcolabilità possiamo limitarci a considerare soltanto funzioni monoargomentali, otteniamo il risultato che ESISTE UN PROGRAMMA CHE CALCOLA QUALSIASI FUNZIONE CALCOLABILE.

COMMENTO GENERALE

Nel campo della teoria della calcolabilità sembra esistere un livello di "complessità" (o di "ricchezza della teoria") tale che se viene superato non c'è alcuna necessità di arricchire ulteriormente la teoria, qualunque sia ciò che noi vogliamo esprimere in essa.

La "ricchezza della teoria" di cui parlavamo prima ha fondamentalmente a che fare con la possibilità di esprimere opportuni <u>sistemi di codifica</u>.

Sono proprio risultati e considerazioni di questo tipo quelli che sostengono e corroborano il programma di ricerca dell'Intelligenza Artificiale e quegli aspetti delle Scienze (e della Psicologia) Cognitive che cercano di costruire MODELLI COMPUTAZIONALI DELLA MENTE.

Per dimostrare il nostro teorema mostreremo come costruire, per ciascun $n > 0$, un programma universale $\mathcal{U}_n$ che calcola $\Phi^{(n)}$, cioè tale che

$$\psi^{(n+1)}_{\mathcal{U}_n}(x_1,\dots,x_n,x_{n+1}) = \Phi^{(n)}(x_1,\dots,x_n,x_{n+1})$$

```
        Z <-- X_{n+1} + 1

                 n
        S <--   ∏   (p_{2i})^{Xi}
                i=1

        K <-- 1
[C]     IF K = Lt(Z) + 1 v K = 0 GOTO F
        U <-- r((Z)_K)
        P <-- p_{r(U)+1}
        IF l(U) = 0 GOTO N
        IF l(U) = 1 GOTO A
        IF ~(P | S) GOTO N
        IF l(U) = 2 GOTO M
        K <-- min_{i≤Lt(Z)} [ l((Z)_i) + 2 = l(U) ]
        GOTO C
[M]     S <-- ⌊S/P⌋
        GOTO N
[A]     S <-- S . P
[N]     K <-- K + 1
        GOTO C
[F]     Y <-- (S)_1
```

PROGRAMMA UNIVERSALE $\mathcal{U}_n$

Prima di discutere istruzione per istruzione il programma che risolve il nostro problema consideriamo i passi principali che tale program[ma] deve compiere:

I. Deve Memorizzare il numero del programma e le variabili di ingresso

II. Deve poter sapere quale istruzione del programma simulato sta effettuando

III. Deve poter effettuare l'istruzione del programma simulato in modo coerente con tutto il marchingegno

IV. Deve poter passare all'istruzione successiva del programma simulato

V. Deve fermarsi quando ha finito di eseguire tutte le istruzioni del programma simulato e fornire il valore corretto della variabile di uscita.

Tra le variabili ausiliarie ne sceglieremo due che, in deroga alla ferrea legge di chiamare le variabili col nome della loro categoria, verranno indicate con K e con S.

K ci dirà qual'è l'istruzione che sta per essere eseguita.

S conterrà lo stato attuale codificato mediante numeri di Gödel come segue:

- assumeremo che, come al solito, le variabili a cui non è assegnato un valore siano azzerate;
- seguiremo sempre l'ordinamento solito delle variabili:

  $Y\ X_1\ Z_1\ X_2\ Z_2\ \ldots$

  da cui si vede che le variabili di ingresso occupano i posti pari nella lista.

  Invece di considerare un insieme di equazioni $V_i = v_i$, considereremo direttamente il numero di Gödel $[v_1, \ldots, v_m]$ per rappresentare lo stato attuale.

La situazione iniziale del programma $\mathcal{U}_n$ per calcolare la funzione $Y = \Phi^{(n)}(X_1, \ldots, X_n, X_{n+1})$ sarà dunque:

- $X_{n+1} = \#(\mathcal{P})$ (numero del programma)
- $[0, X_1, 0, \ldots, X_n]$ (stato iniziale)
- $K = 1$ (deve essere eseguita la prima istruzione)

Vediamo di trasformare queste informazioni in istruzioni di $\mathcal{U}_n$.

Noi vogliamo conoscere in dettaglio le istruzioni di $\mathcal{P}$ per eseguirle quindi dobbiamo passare dal numero di $\mathcal{P}$ alla codifica delle istruzioni.

Ricordiamo che è:

$$\#(\mathcal{P}) = [\#(I_1), \#(I_2), \dots, \#(I_m)] - 1$$

quindi avremo:

$$[\#(I_1), \#(I_2), \dots, \#(I_m)] = X_{n+1} + 1$$

Diamo questo valore ad una variabile ausiliaria Z:

1. $Z \leftarrow X_{n+1} + 1$ (prima istruzione)

Dobbiamo adesso dare ad S come valore il numero di Gödel dello stato iniziale; abbiamo

$$[0, X_1, 0, X_2, \dots, 0, X_n] = \prod_{i=1}^{n} (p_{2i})^{X_i}$$

quindi porremo:

2. $S \leftarrow \prod_{i=1}^{n} (p_{2i})^{X_i}$ (seconda istruzione)

Assegnamo infine a K il valore 1 perché stiamo adesso iniziando e deve essere eseguita la prima istruzione:

3. $K \leftarrow 1$

Passiamo adesso ad istruzioni che riguardano direttamente il programma specifico che stiamo simulando.
Questa parte sarà più difficile perché dovremo prendere in considerazione qualsiasi situazione, <u>qualsiasi successione generica di istruzioni</u>.

L'istruzione successiva è

4. [C] IF $K = Lt(Z) + 1 \vee K = 0$ GOTO F

Con quest'istruzione esaminiamo il caso in cui il programma simulato si ferma.
Questo può avvenire perché, arrivato all'ultima istruzione, dovrebbe andare oltre oppure perché un salto condizionato lo rinvia ad una istruzione inesistente.
Il primo caso è rispecchiato dall'assegnare a K il valore $Lt(Z)+1$
Il secondo lo rispecchiamo assegnando a K un valore ancora non utilizzato.

In entrambi questi casi il programma viene rinviato ad una istruzione F (finale) che si prenderà cura di assegnare alla variabile di uscita Y il valore che il programma è andato calcolando.

5. $U \leftarrow r((Z)_k)$

Per quanto abbiamo detto prima, questa istruzione verrà attivata solo se $K < Lt(Z)+1$, cioè se vi sono istruzioni del programma P che si possono simulare.

Per prima cosa interpretiamo il simbolismo

Che cos'è $(Z)_k$?

Ricordiamo che

- $Z = [\#(I_1), \dots, \#(I_m)]$ è un numero di Gödel
- la funzione $([\,])_i$ applicata ad un numero di Gödel ci restituisce l'i-esimo termine

Quindi $(Z)_k$ è il numero della k-esima istruzione del programma

Ricordiamo ancora che il codice di un'istruzione viene scritto come:

$$\langle a, \langle b, c \rangle \rangle$$

e che r prende la parte destra di tale scrittura.

Quindi : $r((Z)_k) = \langle b, c \rangle$

L'istruzione 5 ci dice di mettere in U il codice dell'enunciato che deve essere eseguito.

Abbiamo quindi

$$U = \langle b, c \rangle .$$

Ricordiamo che b codifica l'istruzione che deve essere eseguita mentre c codifica la variabile menzionata nell'istruzione.

Individuiamo prima la variabile.

Ricordiamo che $c = \#(V) - 1$

per cui $\#(V) = c + 1$

ma $c = r(U)$ e quindi:

$$\#(V) = r(U) + 1$$

Prendiamo allora l'$(r(U)+1)$-esimo numero primo e diamo tale valore a P:

$$6. \quad P \leftarrow p_{r(U)+1}$$

(Ricordo:
Nel nostro meccanismo di codifica il valore di questa variabile viene ricordato come l'esponente al quale è elevato P nella decomposizione in numeri primi di S).

Avendo individuato la variabile presa in esame nella k-esima istruzione dobbiamo adesso prendere in esame l'istruzione che deve essere eseguita.
Questa è codificata da b, cioè da $\ell(U)$ essendo $U = \langle b, c \rangle$.

Il gruppo formato dalle quattro istruzioni seguenti esaminerà l'istruzione codificata da $b = \ell(U)$ e ci dirà cosa fare in corrispondenza dei vari casi.

7. IF $\ell(U) = 0$ GOTO N

Se $b = \ell(U) = 0$ allora l'istruzione è la banale $V \leftarrow V$ e non c'è nient'altro da fare se non passare all'istruzione successiva del programma simulato.
Il programma universale deve quindi rinviarci ad una successiva istruzione che aumenta di 1 il contenuto del "contatore" K.
Possiamo quindi preannunciare che l'istruzione N sarà:

[N]. $K \leftarrow K+1$

la sistemeremo successivamente in una posizione opportuna.

8. IF $\ell(U) = 1$ GOTO A

Se $b = \ell(U) = 1$ allora l'istruzione è $V \leftarrow V+1$.

Ciò che deve fare il programma è di aumentare di una unità l'esponente al quale è elevato P nella decomposizione in fattori primi di S.

L'istruzione A alla quale veniamo rinviati deve quindi fare proprio ciò:

$$[A] \quad S \leftarrow S \cdot P$$

OSSERVAZIONE

Mentre l'istruzione 7 ci ha rinviato direttamente all'istruzione che aumenta il valore dell'istruzione del programma simulato contenuto in K, la 8. non lo fa. Sia la 7. che la 8. dovrebbero poi rinviarci — dato che sono state effettuate — all'istruzione C per ricominciare il processo (se vi sono altre istruzioni da esaminare) ovvero per andare all'istruzione finale.

Di tutto questo dovremo tenere conto successivamente nell'ordinare opportunamente le istruzioni N, A ed altre che occorreranno.

9. IF $\sim(P|S)$ GOTO N

Se siamo giunti a quest'istruzione vuol dire che $\ell(V)$ non era uguale né a 0 né a 1 (perché la 7 e la 8 rinviano rispettivamente ad A ed N che saranno scritte successivamente).
Questo vuol dire che l'istruzione sarà una delle due rimanenti:

$$V \leftarrow V-1$$
$$\text{IF } V \neq 0 \text{ GOTO } L$$

Per entrambe queste due istruzioni distinguiamo due casi: $V=0$ e $V\neq 0$.

Se $V=0$ in entrambi i casi non dobbiamo fare nulla (e quindi passare all'istruzione successiva del programma simulato $\mathcal{P}$).
Ma $V=0$ significa che P non compare nella decomposizione in fattori primi di S
In formule $\sim(P|S)$.
L'istruzione 9 quindi ci dice che se $V=0$ non c'è niente da fare se non aumentare di 1 la variabile K, cosa che, sappiamo già, fa l'istruzione N.

Consideriamo adesso il caso in cui $V \neq 0$ e sia $\ell(U) = 2$.
In questo caso l'istruzione considerata è la $V \leftarrow V-1$.

Ciò che bisogna quindi fare è diminuire la variabile di una unità e questo può essere ottenuto diminuendo di 1 l'esponente col quale P compare nella decomposizione in fattori primi di S.
Ciò è rappresentato dall'istruzione

10. IF $\ell(U)=2$ GOTO M

che rinvia all'istruzione M:

[M] $S \leftarrow \lfloor S/P \rfloor$

Rimane da esaminare il caso in cui l'istruzione in $\mathcal{P}$ da considerare è della forma IF $V \neq 0$ GOTO L con $V \neq 0$

Come fare a rinviare il programma all'istruzione L ?
Assegnando al contatore K il valore numerico corrispondente alla prima istruzione etichettata L in $\mathcal{P}$.

Dobbiamo quindi assegnare a k il più <u>piccolo</u> i tale che

$$\ell((Z)_i) = \#(L)$$

(Nota: il più piccolo perché potremmo avere più istruzioni con la stessa etichetta).

L'istruzione IF $V \neq 0$ GOTO L è codificata da $b = \#(L) + 2$ quindi si ha

$$\#(L) = b - 2 = \ell(U) - 2$$

e quindi $\ell((Z)_i) = \ell(U) - 2$

Possiamo quindi scrivere:

$$11. \quad K \leftarrow \min_{i \leq Lt(Z)} \left[\ell((Z)_i) + 2 = \ell(U) \right]$$

Facciamo seguire alla 11 un'istruzione che fa ricominciare il ciclo

12. GOTO C.

OSSERVAZIONE

Prima di andare oltre chiediamoci cosa succede se non esiste un'istruzione con l'etichetta richiesta. Intuitivamente il programma dovrebbe fermarsi. Ed è ciò che fa il nostro programma universale, nei tempi e nei modi dovuti.
In questo caso infatti l'operatore di minimalizzazione limitata assume il valore 0; la 11 quindi assegna a k il valore 0, la 12 rimanda a C che, a sua volta rimanda ad F, essendo $k = 0$.
La F, come vedremo, non fa altro che assegnare ad Y il valore finale e poi fermarsi

Non ci rimane adesso che sistemare in ordine opportuno le tre istruzioni scritte ma lasciate in sospeso:

Minus [M] $S \leftarrow \lfloor S/P \rfloor$

Aggiungi [A] $S \leftarrow S \cdot P$

[N] $K \leftarrow K+1$

Sia dopo la M che dopo la A bisogna passare alla N e subito dopo ritornare a C per ricominciare il ciclo.

Tali istruzioni possono perciò ordinarsi come segue:

13. [M] $S \leftarrow \lfloor S/P \rfloor$
14. GOTO N
15. [A] $S \leftarrow S \cdot P$
16. [N] $K \leftarrow K+1$
17. GOTO C

Manca solo un'ultima istruzione:

18. [F] $Y \leftarrow (S)_1$

Quest'ultima viene attivata solo dalla C quando il calcolo è finito e cioè o quando abbiamo esaurito tutte le istruzioni o quando un'istruzione di salto condizionato ci ha rinviato ad un'istruzione che non esiste.

Ciò che fa la 18 è quindi di fornire ad Y, la variabile di uscita, il valore conservato in $(S)_1$, cioè nel primo posto dello stato che è quello che contiene, temporaneamente, il valore della variabile di uscita.

Consideriamo il programma P, $\#(P) = 199$

allora $\qquad e \ X = 2$

$Z \leftarrow 200 \qquad (Lt(Z) = 3)$

$S \leftarrow 3^2 \qquad \left(\prod_{i=1}^{n} p_{z_i}^{x_i} \text{, con } x_i = x_1 = 2 \right)$

$K \leftarrow 1$

$200 = 2^3 \cdot 5^2 \Rightarrow (200)_1 = 3$

$U \leftarrow r((Z)_1) \quad [= r((200)_1) = r(3) = 0]$

$2y+1 =$ [redacted] $\dfrac{3+1}{2^2} = 1$

$P \leftarrow p_{r(U)+1} \quad [= p_{r(0)+1} = p_1 = 2]$

IF $\ell(U) = 0$ GOTO N $\quad [\ell(U) = 0]$

[N] $K \leftarrow 2$; GOTO C

IF $K = Lt(Z)+1 \vee K = 0$ GOTO F

poiché $K = 2$ e $Lt(Z)+1 = 4$

eseguo la successiva istruzione

$U \leftarrow r((Z)_2) \quad [= r((200)_2) = r(0) = 0]$

IF $\ell(U) = 0$ GOTO N $\quad [\ell(U) = 0]$

$K \leftarrow 3$

$U \leftarrow r((Z)_3)\ [= r((200)_3) = r(2) = 1]$

IF $\ell(U) = 0$ GOTO N $\quad [\ell(U) = 1]$

IF $\ell(U) = 1$ GOTO A

[A] $S \leftarrow S \cdot P\ [= 3^2 \cdot 2 = 2^1 \cdot 3^2]$

$1 = 2y = 0 \Rightarrow$
$\Rightarrow y = 0$

$K \leftarrow 4$
GOTO C

IF $K = Lt(Z) + 1$ GOTO F $\quad (k = 4$

[F] $Y \leftarrow (S)_1\ [= ([1,2]_1) = 1]$

$S = 2^1 \cdot 3^2$
$(S)_1 = 1$

Consideriamo il programma P
il cui numero è $\underline{2^{45} \cdot 3^2 \cdot 5^{46} - 1}$

Vediamo cosa calcola per $x = 2$.

Avremo che

$Z \leftarrow 2^{45} \cdot 3^2 \cdot 5^{46}$

$S \leftarrow 3^2 \qquad \left(\prod_{i=1}^{n} p_{2i}^{X_i} \text{ , con } X_i = X_1 = X \right)$

$K \leftarrow 1$

$U \leftarrow r((Z)_1) \quad \left[= r\left(2^{45} \cdot 3^2 \cdot 5^{46}\right)_1 = r(45) = 11 \right]$

$P \leftarrow p_{r(U)+1} \quad \left[= p_{r(11)+1} = p_{1+1} = p_2 = 3 \right]$

IF $\ell(U) = 0$ GOTO N $\quad [\ell(11) = 2]$

IF $\ell(U) = 1$ GOTO A

IF $\sim(P|S)$ GOTO N $\quad [\sim(P|S) = \text{falso}]$

IF $\ell(U) = 2$ GOTO M

[M] $S \leftarrow \lfloor S/P \rfloor \quad [= 3]$

GOTO N

[N] $K \leftarrow K+1 \quad [= 2]$

GOTO C

poiché $K = 2 \quad (\neq 4 \text{ e } \neq 0)$

$$U \leftarrow r((Z)_2) \; \Big[= r\left(2^{45} \cdot 3^2 \cdot 5^{46}\right)_2 = r(2) =$$

$$P \leftarrow p_{r(U)+1} \quad \Big[= p_{r(1)+1} = p_{0+1} = p_1 = 2 \Big]$$

IF $\ell(U) = 0$ GOTO N $\quad [\ell(1) = 1]$

IF $\ell(U) = 1$ GOTO A

[A] $\quad S \leftarrow S \cdot P \quad [= 2 \cdot 3]$

$\quad K \leftarrow K + 1 \quad [= 3]$

GOTO C

poiché $K = 3 \quad (\neq 4 \text{ e } \neq 0)$

$$U \leftarrow r((Z)_3) \quad \Big[= r(46) = 23 \Big]$$

$$P \leftarrow p_{r(U)+1} \quad \Big[= p_{r(23)+1} = p_{1+1} = p_2 = 3 \Big]$$

- - - - - - -

$\ell(23) = 3$ quindi si passa a

$$K \leftarrow \min_{i \leq 3} \left[\ell((Z)_i) + 2 = 3 \right] = 1$$

$S \leftarrow \lfloor S/P \rfloor$ $\left[= 2 \cdot 3/3 = 2 \right]$

$K \leftarrow 2$

$P \leftarrow p_{r(U)+1}$ $\left[= p_1 = 2 \right]$

- - - - -

IF $\ell(U) = 1$ GOTO A

[A] $S \leftarrow S \cdot P$ $\left[= 2^2 \right]$

$K \leftarrow 3$

- - - - - - -

IF $N(P|S)$ GOTO N $\left[\begin{array}{c} P=3\,;\, S=2^2 \\ N(P|S) \text{ è vero} \end{array} \right]$

[N] $K \leftarrow K+1$ $\left[= 4 \right]$

GOTO C

- - - - -

GOTO F

[F] $Y \leftarrow (S)_1 = 2$

Programma

$$
\begin{array}{lll}
[A] & X \leftarrow X-1 & (I_1) \\
 & Y \leftarrow Y+1 & (I_2) \\
 & \text{IF } X \neq 0 \text{ GOTO } A & (I_3)
\end{array}
$$

$$\#(I_1) = \langle a, \langle b, c\rangle\rangle = \langle 1, \langle 2, 1\rangle\rangle =$$
$$= \langle 1, 11\rangle = 45$$

perchè $\langle 2,1\rangle = 2^2(2\cdot 1+1)-1 = 2^2\cdot 3 - 1 = 11$

e $\langle 1, 11\rangle = 2^1(2\cdot 11+1)-1 = 2\cdot 23 - 1 = 45$

$$\#(I_2) = \langle 0, \langle 1, 0\rangle\rangle = \langle 0, 2^1(0+1)-1\rangle =$$
$$= \langle 0, 2-1\rangle = \langle 0, 1\rangle = 2^0(2\cdot 1+1)-1 = 3-1 = 2$$

$$\#(I_3) = \langle 0, \langle 3, 1\rangle\rangle = \langle 0, 2^3(2\cdot 1+1)-1\rangle = \langle 0, 23$$
$$= 2^0(2\cdot 23+1)-1 = 46$$

$$\#(P) = [\#(I_1), \#(I_2), \#(I_3)] - 1 =$$
$$= [45, 2, 46] - 1 =$$
$$= 2^{45}\cdot 3^2\cdot 5^{46} - 1.$$

2. <u>Decodifiche</u>

Scriviamo esplicitamente il programma il cui numero è: 199.

Abbiamo: $\#(P) = 199 = [I_1, \dots, I_n] - 1$

quindi $[I_1, \dots, I_n] = 200 = 2^3 \cdot 5^2$

allora $[3, 0, 2] = 200$

Dobbiamo trovare le istruzioni i cui numeri sono rispettivamente 3, 0, 2:

$I_1 : \; 3 = \langle 2, 0 \rangle = \langle 2, \langle 0, 0 \rangle\rangle$

$I_2 : \; 0 = \langle 0, 0 \rangle = \langle 0, \langle 0, 0 \rangle\rangle$

$I_3 : \; 2 = \langle 0, 1 \rangle = \langle 0, \langle 1, 0 \rangle\rangle$

Si ha che

$I_1 \quad [B] \; Y \leftarrow Y$

$\qquad\qquad Y \leftarrow Y$

$\qquad\qquad Y \leftarrow Y + 1$

(~~..~~ calcola, in modo non semplice, la funzione $y = 1$)

have:

$$U(n,m) = T_n(m)$$

for each (n, m) for which T_n is a correctly specified Turing machine.[6] The machine U, when first fed with the number n, precisely imitates the n^{th} Turing machine!

Since U is a Turing machine, it will itself have a number; i.e. we have

$$U = T_u,$$

for some number u. How big is u? In fact we can take *precisely*

u = 7244855335339317577198395039615711237952360672556559631108144796
6065050594042410903104836136323593656444434583822268832787676265561
4469281411771501784255170755408565768975334635694247848859704693472
5739988582283827795294683460521061169835945938791885546326440925525
5058205559894518907165374148960330967530204315536250349845298323206
5158304766414213070881932971723415105698026273468642992183817215733
4828230734537134214750597403451843723595930906400243210773421788514
9276079759763441512307958639635449226915947965461471134570014504816
7337562172573464522731054482980784965126988788964569760906634204477
9890219144379328300194935709639217039048332708825962013017737272027
1862591991442827543742235135567513408422229988937441053430547104436
8695876405178128019437530813870639942772823156425289237514565443899
0527807932411448261423572861931183326106561227555318102075110853376
3380603108236167504563585216421486954234718742643754442879006248582
7091240422076538754264454133451748566291574299909502623009733738137
7241621727477236102067868540028935660856968226201419824862169890260
9130940298570600174300670086896759034473417412787425581201549366393
8996905817738591654055356704092821332221631410978710814599786695997
0450968184190629944365601514549048809220844800348224920773040304318
8429899393135266882349662101947161910701461968523192847482034495897
7095535611070275817487333272966789987984732840981907648512726310017
4016678736347760585724503696443489799203448999745566240293748766883
9751404451665707750060513883991668814072545544665222050724262392379
2115253181625125363050931728631422004064571305275802307665183351995
6891397481375049264296050100136519801869456394 98

(or some other possibility of at least that kind of size). This number no doubt seems alarmingly large! Indeed it *is* alarmingly large, but I have not been able to see how it could have been made significantly smaller. The coding procedures and specifications that I have given for Turing machines are quite reasonable and simple ones, yet one is inevitably led to a number of this kind of size for the coding of an actual universal Turing machine.[7]

I have said that all modern general purpose computers are, in effect, universal Turing machines. I do not mean to imply that the logical design of such computers need resemble at all closely the kind of description for a universal Turing machine that I have just given. The point is simply that, by supplying any universal Turing machine first with an appropriate program (initial part of the input tape), it can be made to mimic the behaviour of any Turing machine whatever! In the description above, the program simply takes the form of a single number (the number n), but other procedures are possible, there being many variations on Turing's original theme. In fact in my own descriptions I have deviated somewhat from those that Turing originally gave. None of these differences is important for our present needs.