

## 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. Ed è 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 e considerare soltanto funzioni monovarientali, 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 "completezza" (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 sistemi di codifica.

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 mostriamo come costruire, per ciascun  $n > 0$ , un programma universale  $\mu_n$  che calcola  $\Phi^{(n)}$ , cioè tale che

$$\Psi_{\mu_n}^{(n+1)}(x_1, \dots, x_n, x_{n+1}) = \Phi^{(n)}(x_1, \dots, x_n, x_{n+1})$$

$Z \leftarrow X_{n+1} + 1$

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

$K \leftarrow 1$

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

$U \leftarrow r((Z)_K)$

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

IF  $l(U) = 0$  GOTO N

IF  $l(U) = 1$  GOTO A

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

IF  $l(U) = 2$  GOTO M

$K \leftarrow \min_{i \leq Lt(Z)} [l((Z)_i) + 2 - l(U)]$

GOTO C

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

GOTO N

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

[N]  $K \leftarrow K + 1$

GOTO C

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

PROGRAMMA UNIVERSALE  $U_M$

Prima di discutere istruzione per istruzione il programma che risolve il nostro problema consideriamo i passi principali che tale programma 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 due, come al solito, le variabili a cui non è assegnato un valore fisso azzerrate;
- seguiremo sempre l'ordinamento solito delle variabili;

$Y \ X_1 \ Z_1 \ X_2 \ Z_2 \dots$

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  $[v_1, \dots, v_m]$  per rappresentare lo stato attuale.

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

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

Vediamo di trasformare queste informazioni in istruzioni di  $U_n$ .

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

Ricordiamo che  $\bar{e}$ :

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

quindi avremo:

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

Diamo questo valore ad una variabile ausiliaria  $Z$ :

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

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:

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

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

$$3. \quad K \leftarrow 1 \quad (\text{terza istruzione})$$

Parliamo adesso ad istruzioni che riguardano direttamente il programma specifico che stiamo simulando.

Questa parte sarà più difficile perché dovremo prendere in considerazione qualsiasi situazione, qualsiasi situazione generica di istruzioni.

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 rinvio ad una istruzione F (finale) che si prenderà cura di assegnare alla variabile di uscita  $Y$  il valore che il programma è andato calcolando.

$$5. \quad U \leftarrow \pi((Z)_k)$$

Per quanto abbiamo detto prima, questa istruzione verrà attivata solo se  $k < L(Z) + 1$ , 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  $([I])_i$  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  $\pi$  prende la parte destra di tale scrittura.

$$\text{Quindi: } \pi((Z)_k) = \langle b, c \rangle$$

L'istruzione 5 ci dice di mettere in  $U$  il codice dell'enuciato 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  $c = \kappa(U)$  e quindi:

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

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

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

(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  $S$ ).

Avevamo 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  $l(U)$  essendo  $U = \langle b, c \rangle$ .

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

7. IF  $l(U) = 0$  GOTO  $N$ .

Se  $b = l(U) = 0$  allora l'istruzione è la banale  $V \leftarrow V$  e non c'è niente altro da fare se non passare all'istruzione successiva del programma simulato.

Il programma universale deve quindi rinviare ad una successiva istruzione che aumenti di 1 il contenuto del "contatore"  $k$ .

Poniamo quindi preannunciare che l'istruzione  $N$  sarà:

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

La risulteremo necessariamente in una posizione opportuna.

8. IF  $l(U) = 1$  GOTO A

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

cioè 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 rinviiati deve quindi fare proprio ciò:

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

### OSSERVAZIONE

Mentre l'istruzione 7 ci ha rinviiato direttamente all'istruzione che aumenta il valore dell'istruzione del programma simulato contenuto in  $k$ , la 8. non lo fa. Se la 7. e la 8. dovrebbero poi rinviiarci - 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 ricorrendo nell'ordinare opportunamente le istruzioni N, A ed altre che occorreranno.

9. IF  $v(P|S)$  GOTO N

Se siamo giunti a quest'istruzione vuol dire che  $l(V)$  non era uguale né a 0 né a 1 (perché la 7 e la 8 rinviano rispettivamente ad A ed N che saranno scritte necessariamente).

Questo vuol dire che l'istruzione sarà una delle due rimanenti:

$V \leftarrow V-1$

IF  $V \neq 0$  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 P).

Ma  $V=0$  significa che P non compare nella decomposizione in fattori primi di S. In formule  $v(P|S)$ .

L'istruzione 9 quindi ci dice che se  $V=0$  non c'è niente da fare se non aumentare di 1 la variabile K, con che, rappiamo già, fa l'istruzione N.

Consideriamo adesso il caso in cui  $V \neq 0$   
e sia  $l(V) = 2$ .

In questo caso l'istruzione considerata è  
la  $V \leftarrow V - 1$ .

Cioè che bisogna quindi fare è dimi-  
nuire 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  $S$ .

Cioè è rappresentato dall'istruzione

10. IF  $l(V) = 2$  GOTO M

che riinvia all'istruzione M:

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

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

Come fare a riinvia il programma  
all'istruzione L?

Assegnando al contatore  $k$  il valore  
numerico corrispondente alla prima  
istruzione etichettata L in  $P$ .

Dobbiamo quindi assegnare a  $k$  il più piccolo  $i$  tale che

$$l((z)_i) = \#(L)$$

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

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

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

$$\text{e quindi } l((z)_i) = l(U) - 2$$

Possiamo quindi scrivere:

$$11. \quad k \leftarrow \min_{i \in L(z)} [l((z)_i) + 2 = l(U)]$$

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

12. `GOTO C.`

#### OSSERVAZIONE

Prima di andare oltre chiediamoci con eccede 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 minimizzazione 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

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

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

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

[N]  $k \leftarrow k+1$

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

Tali istruzioni possono perciò ordinarsi come segue:

13. [M]  $S \leftarrow [S/P]$

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 rinviiato ad un'istruzione che non esiste.

Ecco che fa la 18 e 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

$$e X = 2$$

$$Z \leftarrow 200$$

$$(Lt(Z) = 3)$$

$$S \leftarrow 3^2$$

$$\left( \prod_{i=1}^n P_{2i}^{X_i}, \text{ con } X_i = X_1 = 2 \right)$$

$$K \leftarrow 1$$

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

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

$\frac{3+1}{2^2} = 2$

$$P \leftarrow P_{r(U)+1} \left[ = P_{r(0)+1} = P_1 = 2 \right]$$

IF  $f(U) = 0$  GOTO N  $[f(U) = 0]$

[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

$$U \leftarrow r((Z)_2) \left[ = r((200)_2) = r(6) = 0 \right]$$

IF  $f(U) = 0$  GOTO N  $[f(U) = 0]$



$$K \leftarrow 3$$

$$U \leftarrow r((z)_3) \left[ = r((200)_3) = r(z) = 1 \right]$$

IF  $l(U) = 0$  GOTO N  $[l(U) = 1]$

IF  $l(U) = 1$  GOTO A

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

$1 \rightarrow 2y = 0 \rightarrow$   
 $\rightarrow y = 0$

$$K \leftarrow 4$$

GOTO C

IF  $K = Lt(z) + 1$  GOTO F  $(K = 4)$

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

$$S = 2^1 \cdot 3^2$$

$$(S)_1 = 1$$

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

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

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

$$K \leftarrow 1$$

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

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

IF  $l(U) = 0$  GOTO N  $[l(11) = 2]$

IF  $l(U) = 1$  GOTO A

IF  $\nu(P|S)$  GOTO N  $[\nu(P|S) = \text{false}]$

IF  $l(U) = 2$  GOTO M

[M]  $S \leftarrow LS/P$   $[= 3]$

GOTO N

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

GOTO C

poiché  $k=2$  ( $\neq 4$  e  $\neq 0$ )

$$U \leftarrow \pi((z)_2) \left[ \pi(2^{45} \cdot 3^2 \cdot 5^{46})_2 = \pi(2) = \right]$$

$$P \leftarrow p_{\pi(U)+1} \left[ = p_{\pi(2)+1} = p_{0+1} = p_1 = 2 \right]$$

IF  $l(U)=0$  GOTO N  $[l(1)=1]$

IF  $l(U)=1$  GOTO A

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

$$k \leftarrow k+1 \left[ = 3 \right]$$

GOTO C

poiché  $k=3$  ( $\neq 4$  e  $\neq 0$ )

$$U \leftarrow \pi((z)_3) \left[ = \pi(46)_3 = 23 \right]$$

$$P \leftarrow p_{\pi(U)+1} \left[ = p_{\pi(23)+1} = p_{1+1} = p_2 = 3 \right]$$

$l(23)=3$  quindi si passa a

$$k \leftarrow \min_{i \leq 3} [l(z)_i + 2 = 3] = 1$$

$$S \leftarrow \lfloor S/P \rfloor \quad [ = 2.3/3 = 2 ]$$

$$K \leftarrow 2$$

$$P \leftarrow P_{n(u)+1} \quad [ = P_1 = 2 ]$$

-----  
IF  $l(u) = 1$  GOTO A

$$[A] \quad S \leftarrow S.P \quad [ = 2^2 ]$$

$$K \leftarrow 3$$

-----  
IF  $u(P|S) = 1$  GOTO N  $[ P=3; S=2^3 ]$   
 $[ u(P|S) \neq \text{new} ]$

$$[N] \quad K \leftarrow K+1 \quad [ = 4 ]$$

GOTO C

-----  
GOTO F

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

# Programma

$$[A] \quad X \leftarrow X-1 \quad (I_1)$$

$$Y \leftarrow Y+1 \quad (I_2)$$

$$\text{IF } X \neq 0 \text{ GOTO } A \quad (I_3)$$

$$\begin{aligned} \#(I_1) &= \langle a, \langle b, c \rangle \rangle = \langle 1, \langle 2, 1 \rangle \rangle = \\ &= \langle 1, 11 \rangle = 45 \end{aligned}$$

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

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

$$\begin{aligned} \#(I_2) &= \langle 0, \langle 1, 0 \rangle \rangle = \langle 0, 2^1(0+1) - 1 \rangle = \\ &= \langle 0, 2-1 \rangle = \langle 0, 1 \rangle = 2^0(2 \cdot 1 + 1) - 1 = 3-1=2 \end{aligned}$$

$$\begin{aligned} \#(I_3) &= \langle 0, \langle 3, 1 \rangle \rangle = \langle 0, 2^3(2 \cdot 1 + 1) - 1 \rangle = \langle 0, 23 \rangle \\ &= 2^0(2 \cdot 23 + 1) - 1 = 46 \end{aligned}$$

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

## 2. Decodifiche

Scriviamo esplicitamente il programma il cui numero è : 199.

$$\text{Abbiamo: } \#(P) = 199 = [I_1, \dots, I_n] + 1$$

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

$$\text{allora } [3, 0, 2] = 200$$

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

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

$$Y \leftarrow Y$$

$$Y \leftarrow Y + 1$$

(~~la~~ calcolo, 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.<sup>6</sup> The machine  $U$ , when first fed with the number  $n$ , precisely imitates the  $n^{\text{th}}$  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*

$u = 724485533533931757719839503961571123795236067255655963110814479$   
 $6606505059404241090310483613632359365644443458382226883278767626556$   
 $1446928141177150178425517075540856576897533463569424784885970469347$   
 $2573998858228382779529468346052106116983594593879188554632644092552$   
 $5505820555989451890716537414896033096753020431553625034984529832320$   
 $6515830476641421307088193297172341510569802627346864299218381721573$   
 $3348282307345371342147505974034518437235959309064002432107734217885$   
 $1492760797597634415123079586396354492269159479654614711345700145048$   
 $1673375621725734645227310544829807849651269887889645697609066342044$   
 $7798902191443793283001949357096392170390483327088259620130177372720$   
 $271862591991442827543742235135567513408422299889374410534305471044$   
 $3686958764051781280194375308138706399427728231564252892375145654438$   
 $9905278079324114482614235728619311833261065612275553181020751108533$   
 $7633806031082361675045635852164214869542347187426437544428790062485$

8270912404220765387542644541334517485662915742999095026230097337381  
 3772416217274772361020678685400289356608569682262014198248621698902  
 6091309402985706001743006700868967590344734174127874255812015493663  
 9389969058177385916540553567040928213322216314109787108145997866959  
 9704509681841906299443656015145490488092208448003482249207730403043  
 1884298993931352668823496621019471619107014619685231928474820344958  
 9770955356110702758174873332729667899879847328409819076485127263100  
 1740166787363477605857245036964434897992034489997455662402937487668  
 8397514044516657077500605138839916688140725455446652220507242623923  
 79211525318162512536305093172863142200406457130527580223076651833519  
 95689139748137504926429605010013651980186945639498

(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.<sup>7</sup>

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.

uni  
suc  
uni  
sup  
(ini  
Tu  
the  
pos  
my  
orig  
  
The  
  
We  
idea  
ther  
belo  
phra  
or n  
num  
mat  
any