

## Limiti della Calcolabilità

## Sommario

- Tesi di Church, di Turing, e di Church-Turing
- Gödelizzazione
- Macchina di Turing Universale
- Problema della fermata
- Altri problemi indecidibili

## Tesi di Church

- Nel 1936 Church propone la seguente ipotesi di lavoro, nota come Tesi di Church

**Qualunque funzione sui numeri naturali  
effettivamente calcolabile è ricorsiva**

- dove
  - il concetto di "**funzione ricorsiva**" è definito rigorosamente
  - il concetto di "**effettivamente calcolabile**" non è definito matematicamente, ma solo fisicamente/meccanicamente

## Tesi di Turing

- Nel 1936 Turing propone la seguente ipotesi di lavoro, nota come Tesi di Turing

**Una Logical Computing Machine può eseguire  
qualsiasi calcolo che può essere descritto  
come puramente meccanico**

- Quindi
  - se esiste una procedura effettiva per ottenere il valore di una funzione matematica, allora tale funzione è calcolata da una LCM/MdT
- La nozione di calcolabilità mediante MdT è formale (come quella di funzione ricorsiva)

## Tesi di Church-Turing (1)

- Le due tesi stabiliscono una corrispondenza
  - tra funzioni ricorsive e funzioni T-calcolabili
  - tra esse e la nozione informale di procedura effettiva
- Kleene ha dimostrato l'equivalenza tra le due Tesi
  - Quindi ha senso parlare di Tesi di Church-Turing

## Tesi di Church-Turing (2)

- Ogni algoritmo può essere espresso da un'opportuna MdT → Tutto ciò che è calcolabile, è calcolabile attraverso una MdT
- Ma, dal momento che la classe delle funzioni T-calcolabili coincide con la classe delle funzioni ricorsive generali, allora la tesi è riformulata come
- Una funzione è effettivamente calcolabile sse è ricorsiva generale

## Tesi di Church (2)

- Si dimostra che
  - ogni funzione T-calcolabile è effettivamente calcolabile
  - ogni funzione ricorsiva generale è effettivamente calcolabile

## Gödelizzazione (1)

- Problema: è possibile trattare una TM mediante una TM?
  - Le TM elaborano codifiche di numeri naturali
  - Se riuscissimo a codificare in numeri naturali il comportamento di una TM, allora si potrebbe definire un'altra TM che elabora la prima

## Gödelizzazione (2)

- Il comportamento di una generica TM è completamente definito dalle sue istruzioni
- Ogni istruzione associa alla configurazione corrente un'operazione atomica
  - Configurazione della macchina
    - stato corrente
    - simbolo contenuto nella cella puntata dalla testina
  - Operazione atomica
    - lettura/scrittura di un simbolo
    - spostamento della testina

## Gödelizzazione (3)

- Per codificare una TM è quindi necessario codificare opportunamente:
  - gli stati (finiti)
  - i simboli dell'alfabeto di input (finiti)
  - gli spostamenti della testina (finiti)
  - il blank

## Gödelizzazione (4)

- Sia una TM provvista
  - degli stati  $q_0, q_1, q_2, \dots, q_n$
  - dell'alfabeto di input con simboli  $s_1, s_2, \dots, s_m$
  - del simbolo di blank  $s_0$
  - degli spostamenti L, R, S
- Associamo ad ogni elemento un numero dispari nel modo seguente

L	R	S	$s_0$	$q_0$	$s_1$	$q_1$	$s_2$	$q_2$	$s_3$	$q_3$	...
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
1	3	5	7	9	11	13	15	17	19	21	...

## Gödelizzazione di una Istruzione

- Ogni istruzione I può quindi essere associata al numero naturale  $g(I)$  (numero di Gödel) moltiplicando le cinque potenze aventi come base i primi cinque numeri primi e come esponente rispettivamente i numeri dispari associati agli elementi della istruzione
- Esempio: sia I la istruzione definita dalla configurazione  $\langle q_3, s_1 \rangle$  e con operazione associata  $\langle s_0, L, q_1 \rangle$ , allora  $g(I) = 2^{21} * 3^{11} * 5^7 * 7^1 * 11^{13}$

## Gödelizzazione di una TM

- Dal momento che una TM è identificata dall'insieme (finito) delle sue istruzioni, allora è possibile associare a ogni TM uno specifico numero di Gödel  $g(TM)$ , ottenuto nel modo seguente
  - Sia  $n$  il numero di istruzioni della TM
  - Si moltiplichino le  $n$  potenze aventi come base i primi  $n$  numeri primi, e come esponenti, nell'ordine, i numeri di Gödel delle  $n$  istruzioni di TM

$$g(TM) = 2^{g(I_1)} * 3^{g(I_2)} * 5^{g(I_3)} * \dots * p_n^{g(I_n)}$$

## Macchina di Turing Universale (1)

- È possibile definire una particolare TM (la UTM) capace di simulare il comportamento di qualsiasi altra macchina di Turing  $M$
- La UTM è una TM che riceve in input
  - la codifica di  $M = g(M)$
  - l'input a  $M = I$
- Per ogni  $M$  e per ogni  $I$ , la UTM decodifica le quintuple di  $M$  e le applica a  $I$  ottenendo così lo stesso output che si sarebbe ottenuto se si fosse eseguita la macchina di Turing  $M$  con input  $I$

## Macchina di Turing Universale (2)

- La UTM è in grado di simulare qualsiasi TM
- Quindi, per la tesi di Church, è in grado di calcolare qualunque funzione computabile
- Una TM è una macchina specifica per calcolo di un'unica funzione
- La UTM è un'evoluzione della TM in quanto è programmabile: si può programmare qualsiasi TM, e quindi calcolare qualsiasi funzione (calcolabile)
- La UTM rappresenta il passo dalla computabilità alla programmazione

## Il problema della Fermata (1)

- Stabilire se per ogni TM  $M$  e per ogni input  $I$ , l'esecuzione di  $M$  con input  $I$  termina o prosegue all'infinito?
- Il problema è indecidibile
  - Non esiste alcun algoritmo che, prendendo in input una generica TM  $M$  e un suo generico input  $I$ , produca in output un valore che stabilisce se l'esecuzione di  $M$  su  $I$  termina o meno

## Il problema della Fermata (2)

- Supponiamo per assurdo che il problema della fermata sia decidibile
- Allora (tesi di Church-Turing) esiste una TM Halt che riceve in input la codifica  $g(M)$  di una generica TM  $M$  e un suo generico input  $I$ .
- Halt produce in output
  - 1 se il calcolo di  $M$  con input  $I$  termina
  - 0 se il calcolo di  $M$  con input  $I$  **non** termina

## Il problema della Fermata (3)

- Ma se esiste Halt come quella definita allora è possibile definire un'altra TM Halt' che riceve in input  $g(M)$  e produce in output
  - 1 se il calcolo di  $M$  con input  $g(M)$  termina
  - 0 se il calcolo di  $M$  con input  $g(M)$  **non** termina
- Infatti Halt' è un caso particolare di Halt
  - Non ha più in input la coppia  $\langle g(M), I \rangle$ , ma il solo elemento  $g(M)$

## Il problema della Fermata (4)

- Ma se esiste Halt' come quella definita allora è possibile definire un'altra TM Confuse che riceve in input la codifica di una TM  $g(M)$  e
  - produce in output 0, se Halt' con input  $g(M) = 0$ 
    - cioè Confuse termina con output 0 se Halt' con input  $g(M)$  non termina
  - genera un calcolo che **non** termina, se Halt' con input  $g(M) = 1$ 
    - cioè Confuse non termina se Halt' con input  $g(M)$  termina

## Il problema della Fermata (5)

- Confuse è una macchina **assurda**
- Se applicata a sé stessa, cioè eseguita con input uguale alla sua stessa codifica  $g(\text{Confuse})$ 
  - Confuse termina (con output 0) sse Confuse non termina
  - Confuse non termina sse Confuse termina

## Altri problemi indecidibili

- Una generica istruzione di una TM  $M$  sarà eseguita almeno una volta quando  $M$  è eseguita con input  $I$ ?
- Conseguenza del precedente: una TM con input  $I$  si comporta correttamente?
- Date due TM, esse sono equivalenti?
- Una generica funzione  $f$  è una funzione totale?
- ...

## Conseguenze sulla Programmazione

- Non è possibile essere certi (**in assoluto**) che un dato programma è privo di difetti
- Non è possibile essere certi (**in assoluto**) che un dato programma esegue la funzione per cui è stato creato
- ...