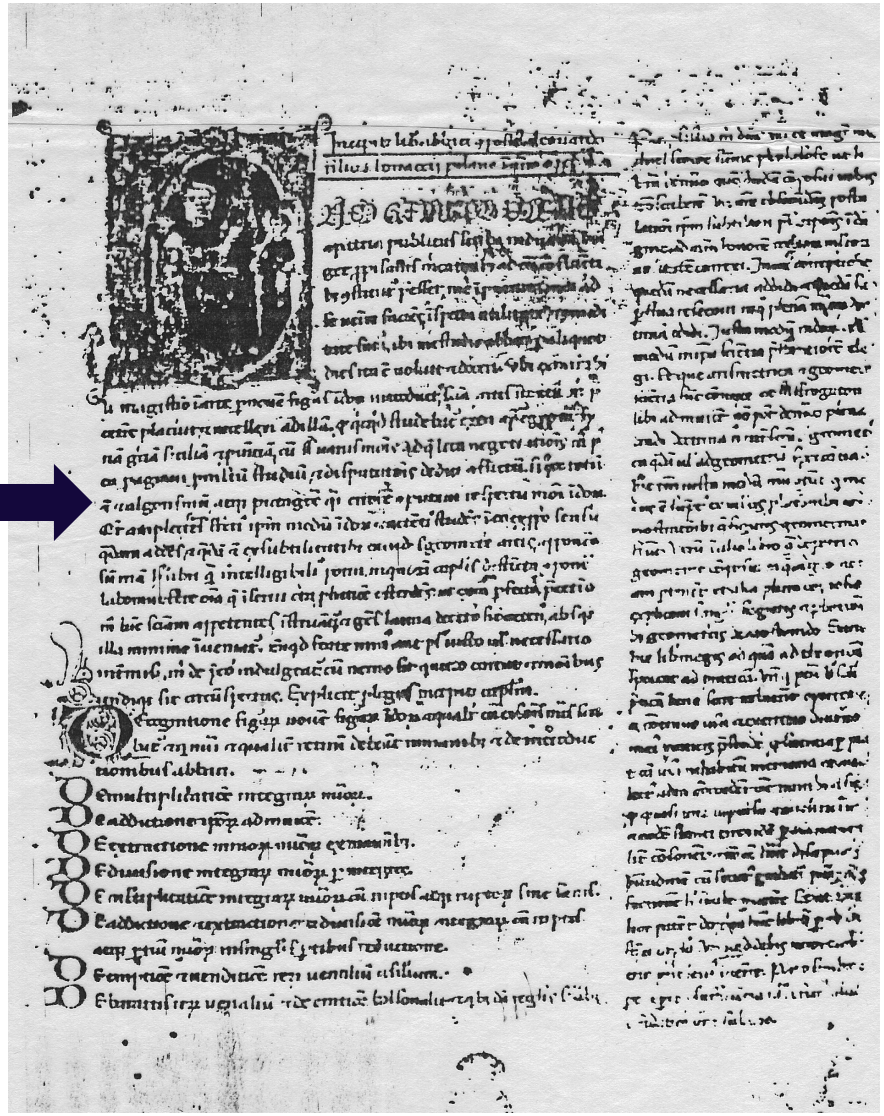
Turing e la nascita dell'algoritmica

Fabrizio luccio

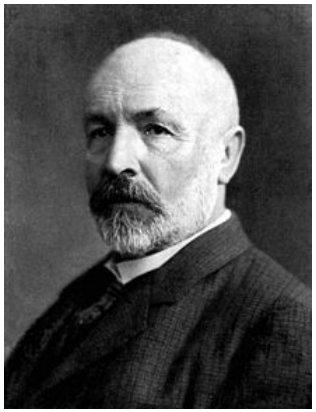
Fibonacci
Liber Abaci
1202



Turing è spesso indicato come padre dell'informatica teorica e dell'intelligenza artificiale, o mitizzato come crittoanalista.

Ma il suo contributo fondamentale è stato una definizione rigorosa del concetto di algoritmo e lo studio delle sue implicazioni.

Cinque personaggi nella nostra storia



Cantor

Hilbert

Church

Gödel

Turing

Verso la fine del 1800 Georg Cantor riconobbe che esiste una gerarchia di grandezza tra gli insiemi infiniti, a partire dagli insiemi numerabili i cui elementi possono essere messi in corrispondenza biunivoca con i numeri naturali.

Già Galileo aveva discusso il paradosso dei quadrati:

0	1	2	3	4	5	6	7	.....
0	1	4	9	16	25	36	49	.....

L'insieme dei quadrati è numerabile,
come lo è l'insieme dei numeri razionali:
ma l'insieme dei numeri reali non è numerabile.

Nel nostro discorso questi concetti sono
fondamentali ragionando sulle sequenze.

Consideriamo un alfabeto finito A = {a,b,c} e le sequenze di lunghezza finita (parole) che si possono costruire con A, e cerchiamo di ordinarle

ε	a	b	c	aa	ab	ac	ba	bb	bc	ca	cb	cc	aaa	aab . . . . .
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14 . . . . .

Dunque le sequenze finite sono numerabili

Gli algoritmi, comunque siano definiti, devono essere rappresentati con sequenze finite e dunque sono numerabili

Consideriamo ora le sequenze infinite

	0	1	2	3	4	5	6	.	.
S_0	1	7	0	3	3	9	2	.	.
S_1	8	3	4	1	5	5	9	.	.
S_2	0	7	7	5	1	0	3	.	.
S_3	6	6	6	2	9	9	1	.	.
..	.	.	.	.	.	.	.	.	.
S_j	2	4	8	3	.	.	.	.	.

$S_2(3)=5$

$$S_j(i) = S_i(i)+1$$

Dunque le sequenze infinite non sono numerabili

Ma le sequenze S_i possono essere la rappresentazioni di funzioni (nell'esempio, dai naturali sui naturali): dunque queste funzioni non sono numerabili

Dunque devono esistere funzioni non calcolabili perché gli algoritmi per calcolarle sono (infinitamente) meno delle funzioni.

Su queste basi sono nati gli studi sulla calcolabilità, che hanno richiesto anzitutto di porre una definizione formale al concetto di algoritmo.

Era l'inizio del 1900.

Dall'inizio del 1900 David Hilbert destinò una grande attenzione al Problema della Decisione, formalizzato nel 1928: stabilire se esiste una procedura meccanica che permetta di decidere in tempo finito se, nell'ambito di una teoria matematica, una qualsiasi affermazione sia vera o falsa.

La procedura che cercava Hilbert era un algoritmo, ma la definizione di algoritmo ancora mancava

Il problema fu risolto da Kurt Gödel in senso negativo (1931): la procedura di Hilbert non può esistere per la aritmetica degli interi.

A tale scopo Gödel costruì una particolare proposizione matematica G e dimostrò che se G fosse dimostrabile lo sarebbe anche la sua negazione. Costruì cioè una antinomia.

Il meccanismo di dimostrazione è implicito in un'epistola di San Paolo (ma il sant'uomo non dovrebbe essersene reso conto).

Si tratta del famoso paradosso del mentitore

Nel 1936 Alonzo Church definì l'algoritmo attraverso un sofisticato paradigma matematico noto come λ-calcolo

Successivamente, nello stesso anno, Alan Turing che era stato allievo di Church definì l'algoritmo attraverso una "macchina"

Più tardi si dimostrerà che le due definizioni sono equivalenti nel senso che consentono di eseguire esattamente gli stessi calcoli, e che questi sono tutti i calcoli che può eseguire un calcolatore di oggi. È la Tesi di Churh -Turing.

La Macchina di Turing: MT = {S, A, B, T}

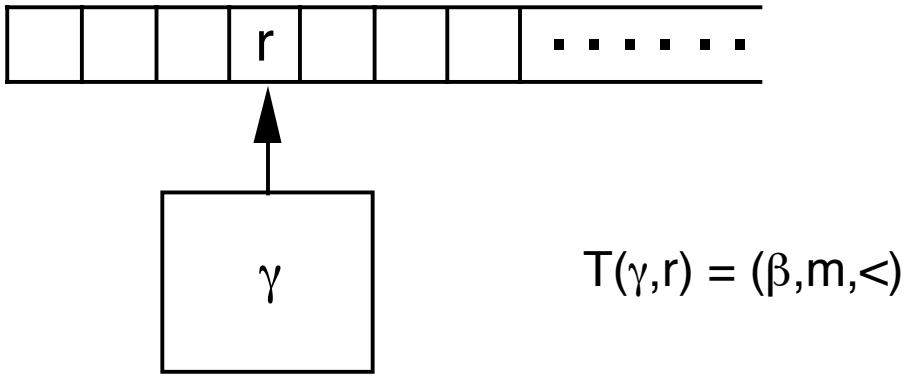
$S = \{\alpha, \beta, \gamma, \ldots, \omega\}$ α stato iniziale
ω stato finale

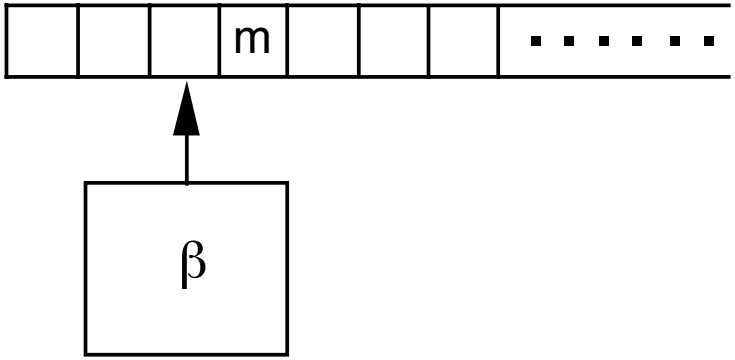
$A = \{a, b, c, \ldots, z\}$ z = blank

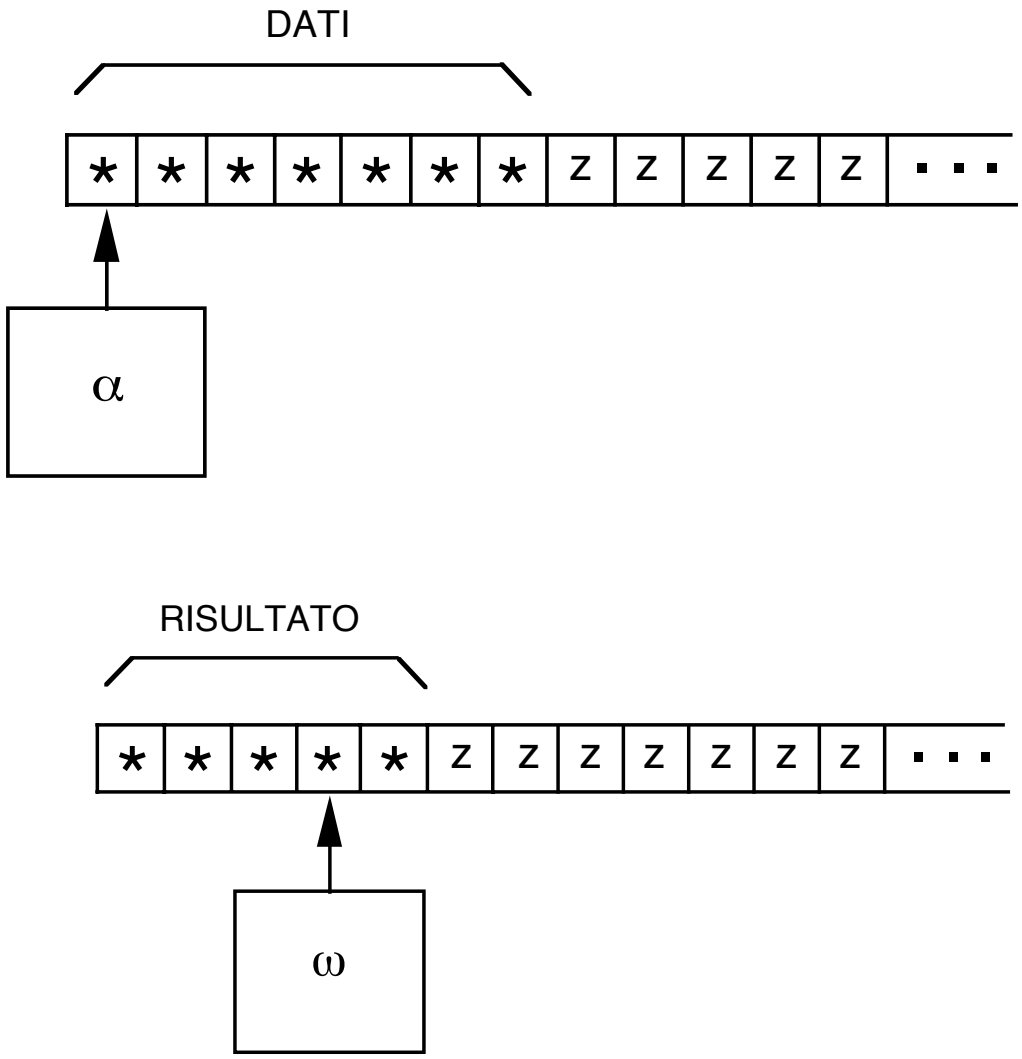
$B = \{<, -, >\}$ < sinistra, – fermo, > destra

T: $S \times A \rightarrow S \times A \times M$

per esempio: $T(\gamma, r) \rightarrow (\beta, m, <)$


r
γ
T(γ,r) = (β,m,<)

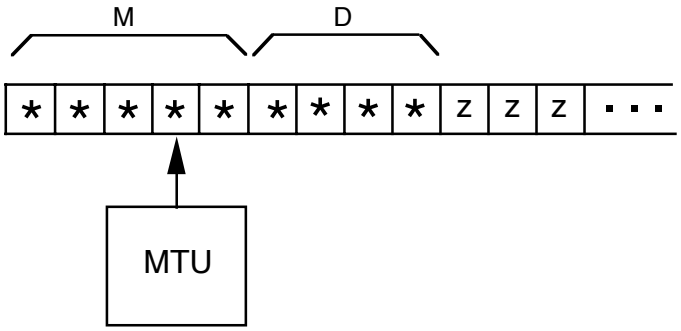

m
β


DATI
* * * * * * * Z Z Z Z Z ...
α
RISULTATO
* * * * * Z Z Z Z Z Z Z ...
ω

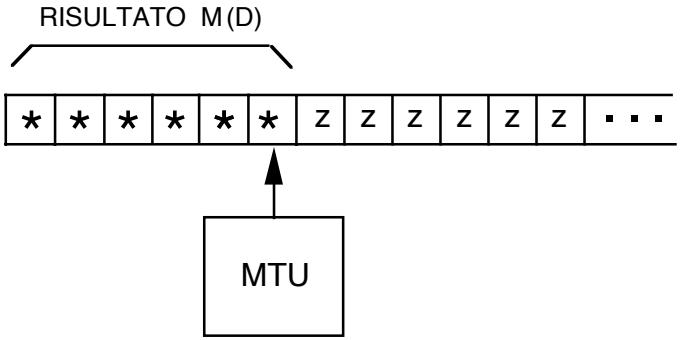
Una Macchina di Turing MT che termina il suo calcolo in tempo finito è un algoritmo (in effetti sarebbe meglio dire il contrario). Algoritmi diversi corrispondono a MT diverse.

Ma osservando che una MT M = {S,A,B,T} si descrive con una sequenza finita di simboli, essa ha la stessa natura della sequenza D dei suoi dati. Dunque si può fornire la M come sequenza di dati per a un'altra MT.

Tutto questo ha permesso a Turing di definire la Macchina Universale MTU che accetta come dati M e D e simula il calcolo di M su D



MTU simula il calcolo di M su D



Dunque la Macchina Universale è un calcolatore

Stabilita una definizione di algoritmo, cerchiamo una funzione non calcolabile

nella teoria del calcolo si parla indifferentemente di funzioni (da calcolare) o di problemi (da risolvere)

Problema della terminazione. Per una MT arbitraria M e i suoi dati arbitrari D decidere (in tempo finito) se il calcolo di M su D termina in tempo finito.

Turing dimostrò anche che il problema della terminazione è non calcolabile

in termini algoritmici, non esiste un algoritmo ALT che decide se un altro algoritmo arbitrario A, operando su dati arbitrari D, termina il suo calcolo in un tempo finito:

```
ALT(A,D)=true,  se A(D) termina

ALT(A,D)=false, se A(D) non termina
```

Consideriamo un nuovo algoritmo P che impiega ALT e lavora su una sequenza A che rappresenta un algoritmo arbitrario:

```
P(A)
se (ALT(A,A)=true) ripeti il test
```

P(A) termina se e solo se A(A) non termina

dunque P(P) termina se e solo se P(P) non termina !!!

il punto debole è l'ammissione che ALT esista

In seguito è stato possibile dimostrare che molti problemi non sono calcolabili, per trasformazione dal problema della terminazione. Per esempio:

- decidere se due programmi arbitrari sugli stessi dati arbitrari generano gli stessi risultati
- decidere se una sequenza arbitraria è random (con conseguenze sulla compressione dei dati)
- decidere se un'equazione polinomiale arbitraria ha soluzione intera

E per concludere studiamo la relazione tra Problema della Terminazione (Turing) e Problema della decisione (Hilbert)

La congettura di Goldbach afferma che ogni numero pari $n > 2$ si può esprimere come somma di due numeri primi

per esempio: $8 = 4+4$ (non primi), ma anche $8 = 3+5$ (primi)

La congettura non è stata mai dimostrata, ma è stata verificata fino a valori altissimi di n.

Consideriamo allora l'algoritmo:

```
GOLD
   per ogni n pari a partire da 4
     costruisci tutte le coppie a,b con a+b=n
     se esiste una coppia a e b di primi
        procedi con il prossimo valore di n
     altrimenti stampa n e arresta il calcolo.
```

Se la congettura fosse falsa, GOLD si arresterebbe sul primo valore di *n* che non la verifica, altrimenti continuerebbe a girare all'infinito. Quindi GOLD termina se e solo se la congettura è falsa per un certo valore di n.

ma se avessimo un algoritmo ALT:

ALT(GOLD)= true (cioè GOLD si arresta)

$\Rightarrow$ la congettura è falsa

ALT(Gold)= false (cioè GOLD non si arresta)

$\Rightarrow$ la congettura è vera

Se esistesse (e io conoscessi) l'algoritmo ALT, avrei un modo banale di dimostrare quasi gratis moltissimi teoremi e congetture sui numeri interi !