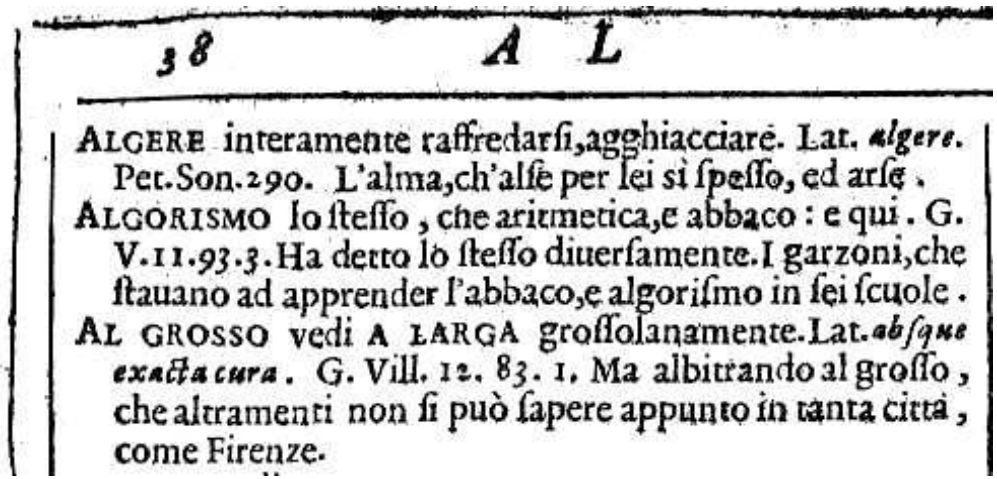
Algoritmi, programmi, teoremi



38 A L

ALGERE interamente raffredarſi,agghiacciare. Lat. *algere*. Pet.Son.290. L'alma,ch'alſe per lei sì ſpeſſo, ed arſe.

ALGORISMO lo ſteſſo, che aritmetica,e abbaco: e qui. G. V.11.93.3.Ha detto lo ſteſſo diuerſamente.I garzoni,che ſtauano ad apprender l'abbaco,e algoriſmo in ſei ſcuole.

AL GROSSO vedi A LARGA groſſolanamente.Lat. *abſque exacta cura*. G. Vill. 12. 83. 1. Ma albitrando al groſſo, che altramenti non ſi può ſapere appunto in tanta città, come Firenze.

Dal Vocabolario dell'Accademia della Crusca I edizione, 1612

Il concetto di algoritmo

Per **algoritmo** si intende *una sequenza finita di passi discreti non ambigui che porta alla soluzione di un problema*. Questa definizione è alquanto involuta e deve essere interpretata parola per parola.

una sequenza finita; l'algoritmo è qualcosa che è descrivibile con un numero finito di simboli, nel giusto ordine, e può essere oggetto di comunicazione tra esseri umani oppure tra uomini e macchine.

di passi discreti, sono escluse azioni continue quali potrebbero essere un colpo di tennis, una mossa di Karate o una nota al violoncello.

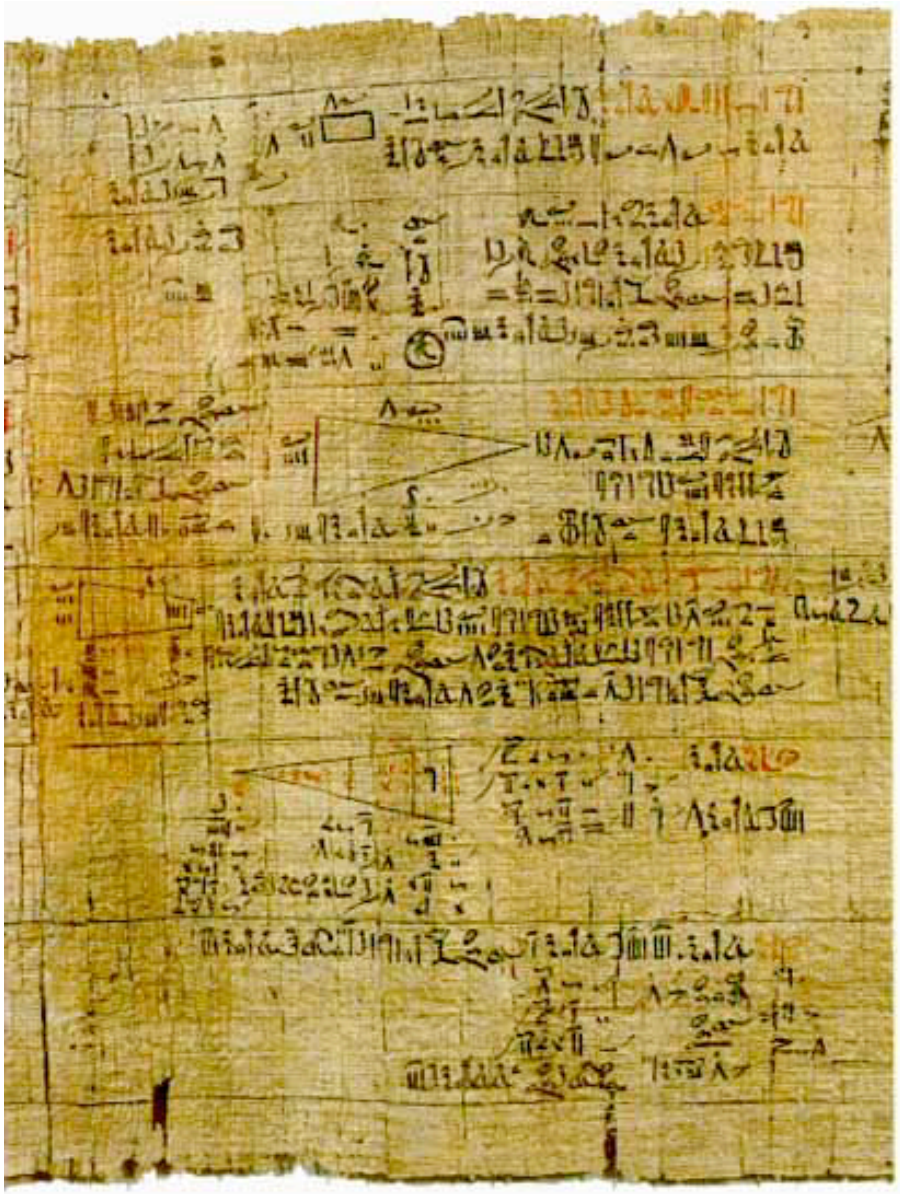
non ambigui questa è la caratteristica più importante che differenzia un vero algoritmo da una qualunque sequenza di chiacchiere. Bisogna che sia definito in modo non ambiguo un linguaggio di comunicazione provvisto con la sua sintassi.

Una volta così strutturato, un algoritmo permette di risolvere il problema per cui è stato pensato.

Bisogna notare che il concetto di algoritmo è antichissimo. Talvolta per esemplificare il concetto di algoritmo si usano gli esempi delle ricette di cucina o delle indicazioni stradali, ma gli esempi più significativi si trovano in matematica e in informatica, dove sequenze anche molto complicate di azioni opportunamente codificate possono servire per comunicare tra studiosi particolari procedimenti oppure per far compiere ai computer determinate azioni.

Algoritmi antichi

Papiro Rhind



Algoritmo di Euclide

Il massimo comune divisore tra due interi positivi è (come dice il nome) il più grande numero che li divide entrambi.

Per esempio MCD(2,7)=1, MCD(36,30)=6.

Si indichi con mod(a,b) il resto delle divisione tra a e b, ovvero

se $a = q\,p + r$ e $r<b$ allora $r = mod(a,b)$.

Euclide notò che MCD(a,b) = MCD(b, mod(a,b))

Algoritmo di Euclide: Calcolare di MCD(m,n)

1) dividi m per n e chiama con r il resto della divisione;
2) se il resto è 0 allora n è il MCD cercato: STOP;
3) altrimenti scambia m con n, scambia n con r (in modo che il vecchio n ora si chiami m e r si chiami n) e vai al punto 1.

L' Algoritmo di Euclide vale anche per i polinomi.

Macchine e programmi

Sistemi formali

Un sistema formale è un tentativo di descrivere in modo rigoroso tecniche per operare sugli oggetti. Vediamo alcuni esempi.

Grammatiche formali

E'un formalismo che permette di imitare le grammatiche dei linguaggi naturali. Vi sono molte specie di grammatiche formali alcune delle quali molto potenti

Esempio 1 (Grammatica lineare o di Tipo 3)

A -> 1A | 0B | 0

B -> 1B | 0C | 1

C -> 0B | 1B | 0 | 1

A è il metasimbolo iniziale

{0, 1} sono i simboli del linguaggio

{A,B,C} sono i simboli del metalinguaggio

Frasi generate:

"11011", "1101011", "11011111101111", "0100"

ma non

"11111", "110110"

Esempio 2 (Grammatica libera da contesto o di Tipo 2)

<frase> -> <soggetto> <verbo> <c. oggetto>.

<soggetto> -> <articolo> cane

<c. oggetto> -> <articolo> cane | <articolo> mela

<articolo> -> Il | La

<verbo> -> mangia | beve

Genera un ristrettissimo sottoinsieme dell'Italiano.

{Il, La, cane, mangia, beve, mela, .} sono i simboli del linguaggio

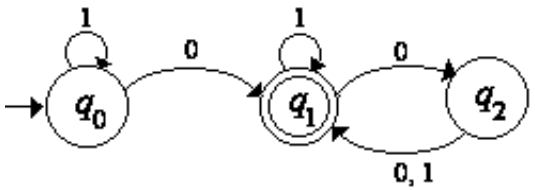
{<frase>, <soggetto>, <articolo>, verbo>, <c. oggetto>} sono i simboli del metalinguaggio

Frasi generate: "Il cane mangia la mela" ma anche "La cane beve il mela".

Macchine Astratte

Automi a stati finiti

Sono macchine dotate di un insieme finito di stati e di un insieme finito di regole per cui al verificarsi di determinati eventi si passa da uno stato all'altro.



L'automa in figura è capace di riconoscere se una stringa appartiene al linguaggio generato dalla grammatica dell'Esempio 1.

Sono nel linguaggio

"11011"

"1101011"

"11011111101111"

"0100"

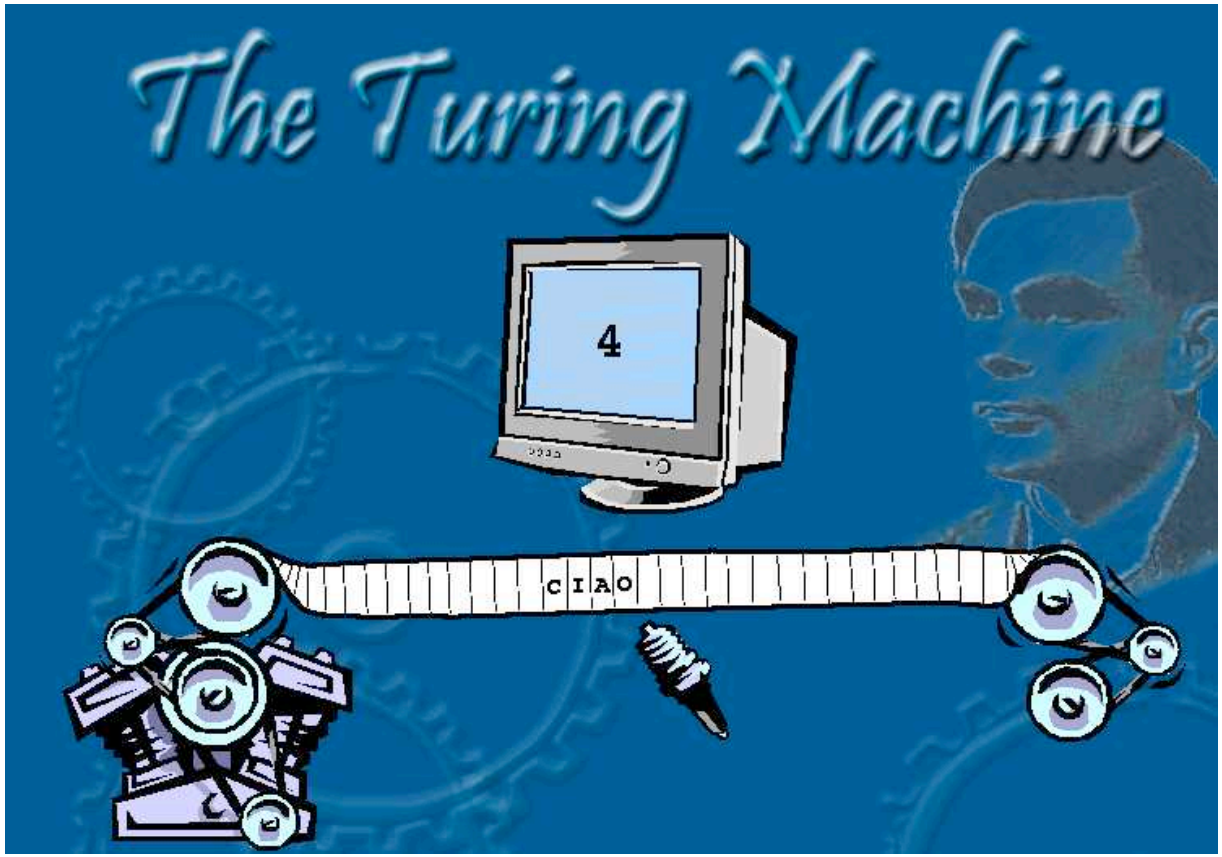
Non sono nel linguaggio

"11111"

"110110"

Macchine di Turing

Nel 1936 il matematico inglese Alan Turing propose l'idea di una macchina astratta che fosse capace di eseguire ogni tipo di calcolo su numeri e simboli.



Una Macchina di Turing (MdT) è definita da un insieme di regole che definiscono il comportamento della macchina (in pratica un automa a stati finiti) e un nastro di input/output. Il nastro può essere immaginato come un nastro di lunghezza infinita, diviso in quadratini dette celle. Ogni cella contiene un simbolo oppure è vuota. La MdT ha una testina che si sposta lungo il nastro leggendo, scrivendo

oppure cancellando simboli nelle celle del nastro. La macchina analizza il nastro, una cella alla volta, iniziando dalla cella che contiene il simbolo più a sinistra nel nastro.

Ad ogni passo, la macchina legge un simbolo sul nastro e in accordo al suo stato interno corrente:

- decide il suo prossimo stato interno,
- scrive un simbolo sul nastro
- decide se spostare o meno la testina a sinistra o a destra di una posizione.

Il comportamento di una MdT può essere programmato definendo l'automa a stati finiti che la controlla. Ciò può essere fatto con un insieme di quintuple, del tipo:

(stato-interno-corrente, simbolo-letto, prossimo-stato-interno, simbolo-scritto, direzione).

NOTA BENE

- Esistono MdT universali, le quali, ricevendo in input l'insieme delle quintuple sono in grado di sumulare il comportamenteo di qualunque altra possibile MdT .
- Le MdT sono goedelizzabili ovvero è possibile determinare la i-esima MdT (basta ordinare i possibili insiemi di quintuple che a loro volta sono stringhe).

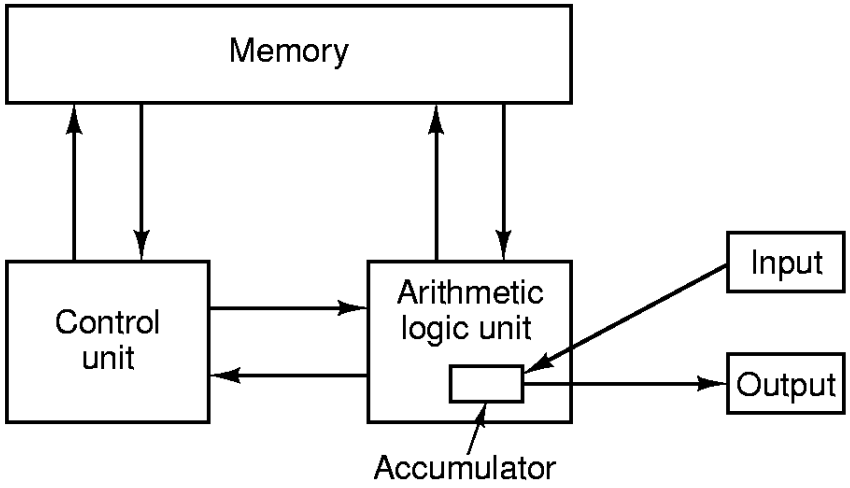
Nessuno usa le MdT per fare i calcoli e nessuno ha mai programmato scrivendo le quintuple (a parte nelle gare di informatica o a scopi didattici).

Un Linguaggio di Programmazione è un linguaggio la cui sintassi è definita rigorosamente (con una grammatica formale) e per cui esiste una macchina virtuale capace di interpretarne i comandi. La coppia linguaggio-macchina costituisce un formalismo equivalente ad una MdT universale ma con il vantaggio che lo stesso programma può essere eseguito su un calcolatore reale.

Calcolatori reali

Un calcolatori reale, da un punto di vista teorico, è assimilabile ad un Automa a stati finiti (in quanto, sulla nera terra, tutto è in quantità finita). Se si immagina invece di disporre di una quantità illimitata di memoria secondaria allora si può simulare una MdT

Lo schema di un calcolatore è tuttoggi basato sulla Macchina di Von Neumann.



Con questa struttura, apparentemente primitiva, si riesce a effettuare qualunque tipo di calcolo, sia su dati numerici sia su testi che immagini.

Teorie logiche

Un posto a parte tra i sistemi formali lo occupano le teorie logiche. Invece di entrare nel dettaglio (servirebbe un corso ad hoc) vediamo esempi del ragionamento matematico.

Definizioni

Le definizioni servono a dare nomi alle cose di cui si parla. Possono essere solo abbreviazioni sintattiche (e in tal caso fanno parte della teoria) oppure possono essere interpretazioni semantiche e in tal caso non fanno parte della teoria ma di un suo modello.

Esempio di definizione come abbreviazione.

Def di limite in Analisi

Si dice che il limite di $f(x)$ per $x \to x_0$ è a se per ogni $\varepsilon>0$, piccolo a piacere,

esiste un $\delta>0$ tale che, se $|x - x_0|<\delta$, allora $|f(x) - a| < \varepsilon$

Esempio di definizione come interpretazione

Def di punto e retta in Euclide

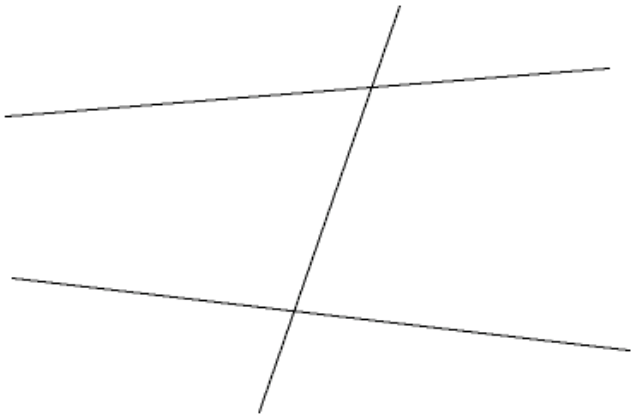
I Σημεῖόν ἐστιν, οὗ μέρος οὐθέν Punto è ciò che non ha parti

Postulati - Assiomi

Una volta si ritenevano i postulati come affermazioni vere nella realtà da cui si partiva con la teoria (= processione). Ora si preferisce parlare di assiomi che sono semplicemente il punto di partenza sintattico (anche se quasi sempre nella mente del matematico vi è il riferimento ad un modello).

Esempio (postulato delle parallele)

> Καὶ ἐὰν εἰς δύο ευθεῖας ευθεῖα ἐμπίπτουσα τὰς ἐντὸς καὶ ἐπὶ τὰ αὐτὰ μέρη γωνίας δύο ὀρθῶν ἐλάσσονας ποιῇ, ἐκβαλλομένας τὰς δύο εὐθείας ἐπῷ ἄπειρον συμπίπτειν, ἐφ᾽ ἃ μέρη εἰσὶν αἱ τῶν δύο ὀρθῶν ἐλάσσονες.
>
> Se una retta, intersecando due rette, forma gli angoli interni sullo stesso lato minori di due angoli retti allora le due rette, se prolungate indefinitamente, si incontrano dalla parte in cui gli angoli sono minori di due angoli retti.



ovvero

> Per un punto ad una retta passa una e una sola parallela.

Teorema

Ipotesi L'insieme delle affermazioni che si suppongono vere.

Tesi L'affermazione che si vuol far discendere dalla tesi.

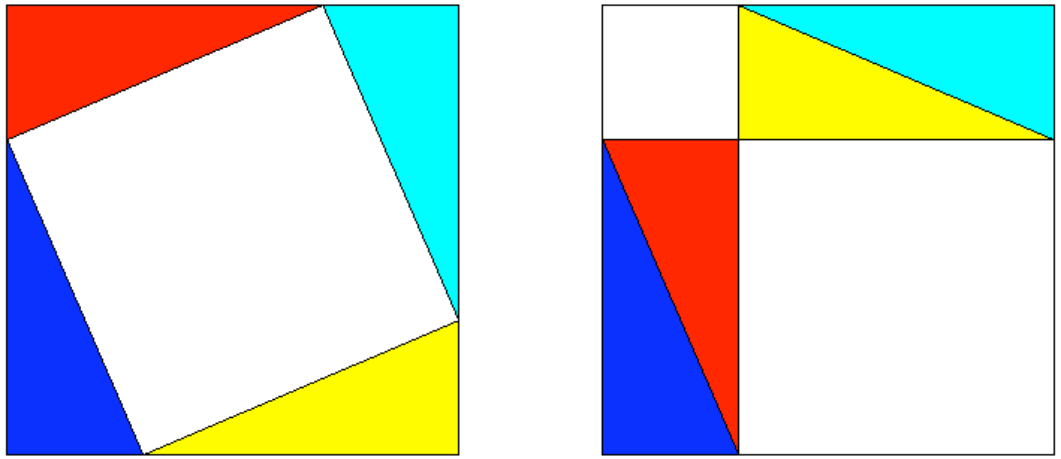
Dimostrazione Una catena di applicazioni delle regole di inferenza ammesse nella teoria che a partire dalle ipotesi conducono alla tesi.

Esempio Teorema di Pitagora

Ipotesi: T è un triangolo rettangolo.

Tesi: La somma delle aree dei quadrati costruiti sui cateti è uguale all'area del quadrato costruito sull'ipotenusa.

Dimostrazione: Guarda!



Lemma

Un lemma è un teorema che viene dimostrato allo scopo di semplificare la dimostrazione di un teorema successivo

Esempio Lemma di Gauss

Un polinomio di grado almeno 1 ha almeno una soluzione nel campo dei numeri complessi.

Teorema fondamentale dell'algebra

Un polinomio di grado n ha esattamente n soluzioni nel campo dei numeri complessi.

Corollario

E' una conseguenza spesso banale di un teorema appena dimostrato

Osservazioni

- Aspetto puramente sintattico di una teoria logica (non si può dimostrare l'esistenza di Dio e neppure che gli uomini hanno una testa e due gambe).
- La scelta degli assiomi è libera ma, una volta fissati gli assiomi, i teoremi sono bloccati.
- Attenzione! Se si parte da ipotesi contraddittorie si può dimostrare qualsiasi cosa. Questa è una terribile trappola per i matematici perché dopo il primo errore nella dimostrazione arrivare in fondo è solo questione di tecnica e il risultato non vale nulla.

Tesi di Church-Turing

Un fatto di fondamentale importanza è che tutti i formalismi sufficientemente potenti sono equivalenti tra loro.

Una volta definiti con precisione i vari formalismi si può dimostrare che: Macchine di Turing, Grammatiche formali, Teorie logiche del primo ordine, Funzioni ricorsive, Programmi, Calcolatori (con memoria illimitata) calcolano tutti lo stesso insieme di funzioni (a meno di operazioni di codifica che sono a loro volta calcolabili).

La Tesi di Church-Turing afferma grosso modo che

All computable functions are computable by a Turing machine

- Solito problema dei termini
- Esistono molte formulazioni diverse la chiave del problema è cosa si intenda per ***computable***. Ci possiamo informalmente rifare ad ogni funzione per cui esiste un algoritmo (inteso nel senso che abbiamo visto nella introduzione)
- Non è un teorema !! Spesso è formulata come una tautologia.
- Ha molte implicazioni filosofiche.

I teoremi di Gödel

Nel suo teorema di completezza Gödel mostra che il calcolo logico del primo ordine è un procedimento di enumerazione effettiva, meccanica e puramente formale, di tutti i teoremi.

Chiamiamo teoria aritmetica ogni teoria tale che i tutti suoi teoremi sono veri nel modello tradizionale dell'aritmetica elementare, quello cioè dei numeri naturali 0,1,2,3,… con addizione e moltiplicazione. Ci riferiremo a questa interpretazione con il nome modello standard dell'aritmetica. Quando in seguito parliamo di enunciati intendiamo delle espressioni che in ogni interpretazione fissata hanno un valore di verità ben definito. Una teoria aritmetica è completa se ogni enunciato che può essere formulato nel linguaggio della teoria o è un teorema della teoria oppure la sua negazione è dimostrabile nella teoria.

Nel 1931, Gödel dimostrò il suo primo teorema di incompletezza:

non esiste nessuna teoria aritmetica completa

La dimostrazione introduce il metodo di diagonalizzazione che permette di costruire per ogni teoria aritmetica un enunciato concreto che è vero nel modello standard ma non è dimostrabile né rifiutabile nella teoria data.

Cerchiamo di capire il significato epistemologico del teorema: ogni possibile tentativo di dedurre all'interno di una teoria aritmetica costruita assiomaticamente tutte e soltanto le proposizioni vere nel modello standard dell'aritmetica è destinato a fallire perché sarà sempre possibile trovare almeno una proposizione vera nel modello standard ma non deducibile né rifiutabile nel sistema assiomatico.

La nozione di verità è quindi intrinsecamente più ricca della nozione di dimostrabilità.

Il Teorema di Incompletezza di Gödel ci costringe a distinguere nelle teorie matematiche l'aspetto sintattico dall'aspetto semantico.

L'importanza dell'aspetto semantico per il concetto di verità è illustrato bene dall'esempio famoso degli assiomi di Euclide.

Mentre per duemila anni si è creduto alla verità *tout-court* di questi assiomi, col tempo ci si è accorti che il contenuto semantico degli assiomi di Euclide non è univocamente determinato e che diverse scelte per l'interpretazione dei concetti di base descritti dagli assiomi portano a teorie diverse tutte tecnicamente valide.

Per esempio le tre possibili forme del postulato delle parallele (per un punto esterno ad una retta passa una, nessuna, infinite parallele) portano a tre diverse geometrie (la geometria euclidea, e quelle non euclidee), tutte ugualmente valide come ha poi mostrato la teoria della relatività generale.

Il Teorema di Incompletezza di Gödel ha distrutto le antiche speranze di un metodo assiomatico universale proprio nel momento in cui sembravano prendere consistenza in seguito alle ricerche moderne sui fondamenti della matematica.

E' stato però un terremoto benefico che ha chiarito il vero ruolo del metodo assiomatico ed ha fornito una tecnica originale che ha aperto nuovi orizzonti di ricerca matematica. Abbiamo imparato che per molte teorie matematiche non si può fornire alcuna garanzia assoluta che siano esenti da contraddizioni interne.

Calcolabilità e decidibilità

Consideriamo le funzioni dai naturali ai naturali.

Esempio: quadrato(n) = n × n.

Teorema: Esistono funzioni non calcolabili.

Dimostrazione: I programmi (in qualunque forma siano dati) sono numerabili: le funzioni sono tante quanti i numeri reali, *ergo* le funzioni sono molte di più dei programmi.

Esempio di una funzione non calcolabile: (Presentazione informale da un idea di Egon Börger)

Non esiste un programma che calcoli la funzione

```
boolean halt(String pgm, String inp);
```

che vale true se il programma pgm con l'input inp termina e false altrimenti.

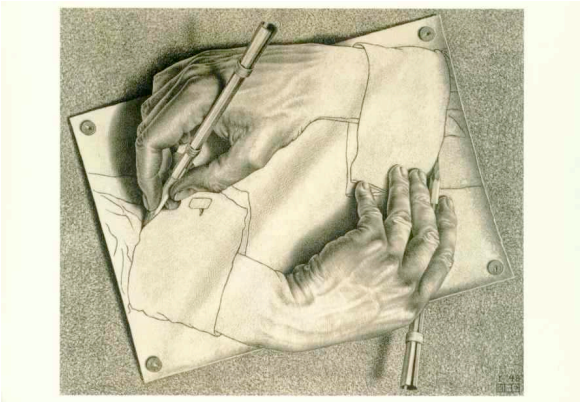
Dimostrazione. Data il programma

```
void diag(String pgm){while(halt(pgm,pgm));}
```

Si vede facilmente che:

- se `diag(diag)` termina, `halt(pgm,pgm)=true` e allora `diag(diag)` va in loop e non termina.
- se `diag(diag)` non termina, `halt(pgm,pgm)=false` e allora `diag(diag)` termina.

Si noti la somiglianza con il paradosso del barbiere e gli altri ragionamenti diagonali.



Attenzione questi sono limiti intrinseci dell'approccio sintattico. Sono limiti logici che valgono per l'uomo prima ancora che per le macchine.

Come vedremo i limiti per l'uomo sono ancora più stretti.

Ogni tentativo di usare queste limitazioni per asserire una superiorità teorica dell'uomo sulle macchine é stupido prima ancora che falso.

D'altra parte se si comincia a tirare dentro l'intuizione, i sentimenti e la coscienza (che sono i campi su cui si combatte la "battaglia" uomo-macchina) si esce completamente da questo contesto.

Complessità in tempo

Le funzioni non calcolabili non esauriscono la classe dei problemi impossibili da risolvere. Molti problemi sono perfettamente risolubili in teoria ma assolutamente intrattabili in pratica.

Una vita umana lunga dura, diciamo più o meno 100 anni, sono

$$100 \times 365 \times 24 \times 60 \times 60 = 3.153.600.000 \text{ secondi}$$

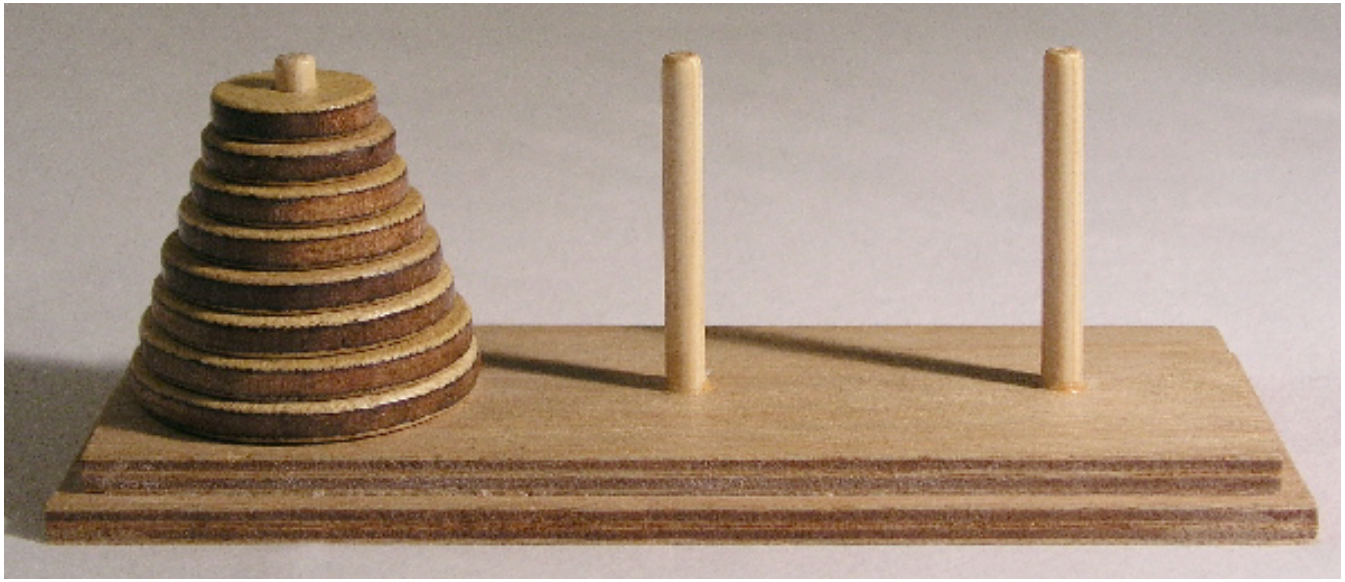
Nessun uomo può contare fino a 4 miliardi, anche non facendo null'altro nella vita.

Usando un computer che fa mille miliardi di operazioni al secondo (un Teraflop) e supponendo che questo possa funzionare senza interruzioni per 100 anni si riescono a fare solo

$$3.153.600.000.000.000.000.000 = 3.2\ 10^{21} \text{ operazioni}$$

Sembrano tante ma è facile trovare problemi apparentemente semplici che ne richiedono di più.

La Torre di Hanoi



Si tratta di un rompicapo molto famoso la cui soluzione non è difficile ma lunga.

Per portare n dischi dal primo piolo all'ultimo, senza mettere mai un disco grande sopra uno più piccolo, sono necessarie 2^n-1 mosse.

La soluzione consiste nel

- portare n-1 dischi nel piolo intermedio (seguendo le regole del gioco).
- spostare l'ultimo disco (il più grande)
- muovere di nuovo gli n-1 dischi.

Il primo e il terzo passo sono un problema analogo di dimensione n-1. L'equazione della complessità diviene

$T(n) = 2\,T(n-1) + 1$ con la condizione iniziale $T(1) = 1$

e si dimostra facilmente

$T(n) = 2^n - 1$

Soluzione animata

Una torre di Hanoi con 100 dischi (facilmente costruibile) richiederebbe

1.267.650.600.228.229.401.496.703.205.375 operazioni

ovvero circa 40 miliardi di anni col nostro velocissimo computer.

Quando si studia la complessità (ovvero il costo di risoluzione) di un problema è bene cercare di valutare come questa cresce al variare delle dimensioni dello stesso.

Classi di complessità ed esempi,

Vediamo alcuni esempi di problemi di difficoltà crescente, per ognuno di essi diamo la massima dimensione del problema affrontabile in un'ora su una macchina da un miliardo di operazioni al secondo (un PC da 1000 Euro) e su un supercomputer mille volte più potente (costo qualche milione di euro)

Complessità logaritmica

Ricerca binaria (per esempio la ricerca di un numero su un elenco telefonico, dato il nome).

Ad ogni passo l'insieme da analizzare si dimezza.

$R(n) = R(n/2) + 1$, $R(1) = 1$, da cui $R(n) = \lceil \mathrm{Log}_2(n) \rceil$.

se n = 1.000.000.000 il PC ci mette circa 30 passi (tempo trascurabile)

Complessità lineare

Ricerca lineare (per esempio la ricerca di un nome su un elenco telefonico dato il numero), Somma di n numeri.

Dimensione limite sul PC: $n = 3.6\ 10^{12}$

Dimensione limite sul supercomputer $n = 3.6\ 10^{15}$

Complessità quasi lineare

Ordinamento di n numeri. Ha complessità n log n.

Dimensione limite sul PC: $n = 1.4\ 10^{11}$

Dimensione limite sul supercomputer $n = 1.1\ 10^{14}$

Un problema di complessità quasi-lineare può essere solo apparentemente un problema semplice.

Biblioteca del Cairo

Complessità quadratica

Moltiplicazione di interi - Somma di matrici

Dimensione limite sul PC: $n = 1.8\ 10^6$

Dimensione limite sul supercomputer $n = 6\ 10^7$

Complessità cubica

Moltiplicazione di matrici (con l'algoritmo banale)

Dimensione limite sul PC: $n = 15.000$

Dimensione limite sul supercomputer $n = 150.000$

Complessità esponenziale

Torre di Hanoi $T(n) = 2^n-1$

Dimensione limite sul PC: $n = 40$

Dimensione limite sul supercomputer $n = 50$

Chi non ha testa abbia gambe

Vi sono due modi per rendere più veloce la soluzione di un problema:

- usare un computer più veloce (approccio *brute-force*);
- trovare un algoritmo più furbo (usare il cervello).

Tempo di ordinamento di n numeri con due diversi algoritmi su due diverse macchine

n	Quicksort ($n \log n$) sul PC	Bolle (n^2) sul supercomputer
10	<0.001 sec	<0.001 sec
100	<0.001 sec	<0.001 sec
1000	<0.001 sec	<0.001 sec
10^6	0.01 sec	1 sec
10^9	30 sec	$10^6 \approx$ 11 giorni
10^{12}	39.800 ≈ 10 ore	$10^{12} \approx$ 30000 anni

Necessità di trovare limitazioni inferiori per non scervellarsi inutilmente.

Finora abbiamo parlato di algoritmi che RISOLVONO un problema ma purtroppo non viviamo in un mondo di frutta candita ... di libellule appena volate e la dura realtà influenza anche l'informatica.

Vi sono molte variazioni sul tema della quasi-soluzione di problemi.

Errore di arrotondamento.

I numeri interi hanno una rappresentazione finita (e se piccoli sono anche maneggevoli) , i numeri reali hanno una rappresentazione illimitata e non sono maneggevoli.

Purtroppo i matematici, gli ingegneri, i chimici, i fisici ecc. gradirebbero lavorare con numeri reali che vanno quindi approssimati. Questo porta ad errori detti di arrotondamento che capitano anche quando si fanno i conti della spesa.

Se i numeri reali sono rappresentati usandolo solo un numero finito di cifre ogni operazione matematica (somma, sottrazione, moltiplicazione, divisione) introduce quasi sempre un errore dovuto al modo con cui i dati sono rappresentati.

Errore algoritmico

Un altro problema da affrontare quando si fanno calcoli numerici è che l'errore dipende dal modo di effettuarli.

Sommiamo i primi 100.000 numeri interi, il risultato esatto è 5.000.050.000. Vediamo cosa accade se lavoriamo con numeri reali rappresentati su 4 byte con tre algoritmi diversi.

1) somma crescente si somma 1 con 2, il risultato con 3 e così via fino a sommare 100.000; si ottiene come risultato 4.999.990.300 con un errore di 59.728.

2) somma decrescente si somma 100.000 con 99.999, il risultato con 99.998 e così via fino a sommare 1; si ottiene come risultato 5.000.086.500 con un errore di -36.528.

3) somma bilanciata: si somma 1 con 2, 3 con 4, … 99.999 con 100.000, poi a due a due si sommano i 500.000 risultati parziali e si continua così fino ad ottenere un numero solo; si ottiene come risultato 5.000.049.700 con un errore di sole 336.

Metodi Iterativi

Sono metodi che approssimano una soluzione numerica con un errore inevitabile sempre più piccolo utilizzando ogni volta i risultati del passo precedente..

Calcolo della Radice di 2 col Metodo di Newton

Valore esatto

1.41421356237309504880168872420969807856967187537694807317667973799073247846210703885038753432764157 3...

Metodo Iterativo

$$x(i) = \frac{1}{2}\left(x(i-1) + \frac{2}{x(i-1)}\right)$$

Valori calcolati

1.8473...

1.41429...

1.41421356421...

1.414213562373095049999929...

1.414213562373095048801688724209698079067550972 3...

1.414213562373095048801688724209698078569671875376948073176679
7379907324785530174220938328253104 3...

La precisione del calcolo aumenta con le iterazioni

Calcolo di π col metodo di Ramanujan

Metodo di Ramanujan delle equazioni modulari, π viene approssimato da 1/a(n)

$$y(0) = \frac{1}{\sqrt{2}}; \quad a(0) = \frac{1}{2};$$

$$y(n\_) := \frac{1 - \sqrt{1 - y(n-1)^2}}{\sqrt{1 - y(n-1)^2} + 1}$$

$$a(n\_) := (y(n) + 1)^2 a(n-1) - 2^n y(n)$$

Valore esatto di π

3.14159265358979323846264338327950288419716939937510582097494459230
781640628620899862803 48...

Valore calcolato nelle varie iterazioni

2.91421356237309504880168872420969807856967187537694807317667973799
07324784621070388503875343276415 73

3.14057925052216824831133126897582331177344023751294833564348669334
55827580349029078272876215527669 01

3.14159264621354228214934443198269577431443722334560279455953948482
14347672207952646946434489179913 06

3.14159265358979323827951277480186397438122550483544693578733070202
63821378389273990314169420434690 58

3.14159265358979323846264338327950288419711467828364892155661710697
60267645006430617110065777265980 68

3.14159265358979323846264338327950288419716939937510582097494459230
78164062862089986256287032116720 36

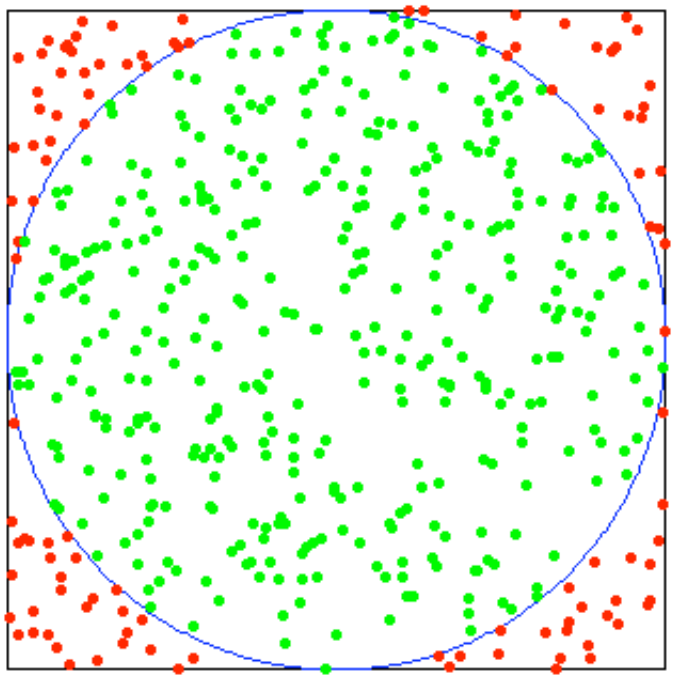
In questo caso i calcoli vanno fatti **tutti** alla massima precisione.

Algoritmi probabilistici (da non confondere con i non-deterministici)

Una strada molto promettente per risolvere problemi difficili è quella di risolverli utilizzando il caso.

Algoritmi Montecarlo. Si ottiene un risultato probabilmente scorretto in tempo finito.

Esempio: calcolo dell'area del cerchio. Si "sparano" a caso n proiettili in un quadrato di lato 2 (di area 4), di questi m cadono nel cerchio (di area π) (diciamo m) il rapporto m/n tende all'area del cerchio ovvero a π. Sembra un approccio ridicolo ma questo è l'unico modo pratico per approssimare la soluzione di difficili problemi in molte dimensioni che si presentano in fisica e in ingegneria.



Algoritmi Las Vegas. Si ottiene un risultato corretto in tempo probabilmente piccolo.

Un bell'esempio di calcolo Las Vegas è l'algoritmo Quicksort (uno dei più efficienti ed usati algoritmo di ordinamento) che in media richiede n log n operazioni ma che nel caso peggiore (altamente improbabile) può arrivare a n^2.

Algoritmi Atlantic City. Si ottiene un risultato probabilmente corretto in tempo probabilmente piccolo.

Test di primalità. I numeri primi molto grandi (migliaia di cifre) sono molto richiesti in crittografia (dove vengono usati per generare le chiavi). Algoritmi deterministici per verificare la primalità di un numero molto grande sono in genere inapplicabili per il loro costo, ma esistono test di primalità probabilistici.

Se uno di questi algoritmi stabilisce che un numero p non è primo questo è certamente vero, se invece l'algoritmo afferma che p è primo questo risultato ha certa probabilità di errore.

Applicando più volte l'algoritmo si può ridurre a piacere la probabilità di errore fino ad un renderla in pratica trascurabile.

Complessità in spazio

E' la misura della quantità di memoria che serve per risolvere un problema.

A volte se si ha poco spazio di memoria a disposizione si devono ripetere più volte gli stessi calcoli (invece di immagazzinare i risultati intermedi) e questo aumenta il tempo di esecuzione. Viceversa aumentare la memoria non sempre rende più rapida la soluzione

Lo spazio diviene un problema quando si usano grandi quantità di dati

Nelle macchine moderne la memoria è distribuita si molti livelli di dimensione e tempo di accesso crescenti:

tipo	dimensione tipica	tempo di accesso
cache del processore	100 Kilo Byte	$<10^{-9}$ sec
cache di II livello	1 Mega Byte	$\sim 10^{-9}$ sec
memoria RAM	1 Giga Byte	$\sim 10^{-8}$ sec
hard disk	250 Giga Byte	$\sim 10^{-3}$ sec
archivi fuori linea (nastri, DVD)	illimitata	1 sec - ore

Complessità Program-size

Una versione alternativa del paradosso di Berry divide i numeri in interessanti (quelli dotati di una qualunque proprietà) e noiosi (gli altri, la stragrande maggioranza).

Il più piccolo numero noioso è interessante!!

Tutti questi discorsi vaghi divengono precisi se si comincia da usare un formalismo rigoroso: per esempio una teoria logica o una MdT universale.

Definizione: Fissata una MdT universale e una stringa binaria α la complessità program-size di α è la lunghezza in bit del più piccolo programma che stampa α.

NB ogni stringa α può essere stampata con il programma "STAMPA α" la cui lunghezza però è maggiore di quella di α.

Potremmo definire interessanti le stringhe per cui la complessità program-size è piccola e noiose le altre. In pratica si preferisce chiamare casuali (random) le stringhe non prevedibili il cui unico modo di essere espresse è l'enumerazione diretta.

In questo senso le stringhe casuali sono anche incomprimibili.

Il paradosso di Berry diviene un enunciato di incompletezza molto simile al Teorema di Goedel.

Teorema (Chaitin). In un sistema assiomatico coerente, con n bit di assiomi, non si può dimostrare la casualità di stringhe con più di n+c bit.