Indecidibilità e computabilità

- nel '37 Turing enuncia l'HALTING PROBLEM, forse ispirato dal lavoro di Gödel
- nel '60 Kolmogorov dà una def. formale di computabilità

Un algoritmo A è anch'esso una seq. binaria — e come tale può essere un input a sua volta.

HALTING PROBLEM: esiste un algoritmo in grado di stabilire se un programma termina, qualunque esso sia?

Teorema Non esiste un tale algoritmo.

Dim. Supp. esista per assurdo e che sia ulog modellato come:

$$\text{TERMINA}(A, D) = \begin{cases} 1 & \text{se } A(D) \text{ termina} \\ 0 & \text{altrimenti} \end{cases}$$

Se allora definiamo:

```
PARADOSSO(X):
    while TERMINA(X, X),
```

vale che:

PARADOSSO (PAR.) termina $\iff$
$\iff$ PARADOSSO (PAR.) non termina, ↯.

Quindi l'algoritmo non esiste. ∎

Vale anche il seguente teorema:

Teorema Esistono problemi che non ammettono algoritmi risolutivi.

Dim Gli algoritmi sono seq. binarie, i quali app. a $\{0,1\}^*$.

$$\{0,1\}^* = \bigcup_{i \geq 0} \{0,1\}^i \longrightarrow \text{insieme finito}$$

($\{0,1\}^i$: stringhe di i cor.)

Poiché gli $\{0,1\}^i$ sono finiti e un'unione numerabile di finiti è numerabile, $|\{0,1\}^*| = \aleph_0$.

Ci contrario i progr. sono funzioni $\{0,1\}^* \to \{0,1\}^*$; e $|\{0,1\}^* \to \{0,1\}^*| = |\mathbb{N}^{\mathbb{N}}|$.

- $|\mathbb{N}^{\mathbb{N}}| \leq |(2^{\mathbb{N}})^{\mathbb{N}}| = |2^{\mathbb{N}\times\mathbb{N}}| = |2^{\mathbb{N}}| = |\mathbb{R}|$
- $|\mathbb{R}| = |2^{\mathbb{N}}| \leq |\mathbb{N}^{\mathbb{N}}|$

$\Rightarrow |\mathbb{N}^{\mathbb{N}}| = |\mathbb{R}|$.

Poiché $|\mathbb{R}| > |\mathbb{N}|$ (arg. diag. di Cantor), si ottiene la tesi. ■

↝ l'HALTING PROBLEM è un esempio di prob. non risolubile

↝ un altro problema irrisol. è quello dell'equiv. tra programmi

Kolmogorov complexity

Vogliamo studiare in modo formale la casualità delle seq. (e.g. stringhe)

- 01234...90 non è casuale! C'è un programma che enumera la stringa in poche righe
- 3826 sembra casuale invece... non ci vengono in mente programmi per compilarla

Def. $K_L : \{0,1\}^* \to \mathbb{N}$ — dove L è il linguaggio utilizzato — restituisce per $x \in \{0,1\}^*$ la lunghezza del prog. più breve per stampare la seq. x.

↝ scriveremo per semplicità $K(S)$ sottintendendo un linguaggio già stabilito a priori.

↝ la formalizzazione completa di questa teoria si studia per mezzo della MACCHINA DI TURING, che non studieremo.

Def. (Kolmogorov complexity) Una stringa x si dice CASUALE se $K(x) \geq |x| - c$ dove c è una costante detta dal linguaggio (anche 0).

Prop. 1 $\forall n \; \exists x$ stringa di n car. casuale.

Dim. Vi sono $2^n - 1 = \sum_{i=0}^{n-1} 2^i$ prog. di massimo $n-1$ bit e ognuno produce al più una stringa di lungh. n. Le stringhe di lungh. n sono però 2^n — per cardinalità una stringa casuale deve allora esistere. ■

Prop. 2 $Pr(K(x) \geq n - c) \geq 1 - \frac{1}{2^c}$ su $x \in \{0,1\}^n$ (prob. uniforme)

Dim. Ogni stringa con $K(x) < n - c$ è descritta da un prog. con meno di $n - c$ bit.

Esistono al più:

$$1 + 2^1 + \cdots + 2^{n-c-1} = 2^{n-c} - 1$$

programmi con meno di $n-c$ non equivalenti, quindi al più $2^{n-c}-1$ stringhe non casuali di n caratteri.

Allora:

$$Pr(K(x) \geq n-c) = 1 - Pr(K(x) < n-c) \geq$$

$$\geq 1 - \frac{2^{n-c}-1}{2^n} = \frac{}{|\{0,1\}^n|}$$

$$= 1 - \frac{1}{2^c} + \frac{1}{2^n} > 1 - \frac{1}{2^c}$$ ■

Grazie a queste due prop Kolmogorov mostra un risultato in sintonia con quelli ott. da Turing:

Teorema Non esiste un algoritmo in grado di stabilire se una stringa è casuale o meno.

Dim Supponiamo che un tale algoritmo esista e chiamiamolo A.

Sia B_n l'alg che enumera in ordine lessicografico le stringhe di n caratt.

e che si ferma alla prima stringa casuale, restituendola. Denotiamo questa stringa come x_n. Allora B_n genera $x_n \Rightarrow$ $\Rightarrow |B_n| \geq K(x_n) \geq n - c$. Tuttavia B_n dipende solo da A — che ha lunghezza fissa! — e da n in bit. Quindi $|B_n| = c' + \log_2(n)$ per una cost c'.

Quindi $c' + \log_2(n) \geq n - c$, che è asintoticamente assurdo, $\lightning$. ■