

Teoria della computabilità

Lezioni 1 – 3

Pierluigi Minari

Dipartimento di Filosofia
Università di Firenze

AILA, Scuola Estiva di Logica 2007

Schema

- 1 Ricorsività e Computabilità
 - Ricorsività dal punto di vista intuitivo
 - Computabilità dal punto di vista intuitivo
- 2 1930–1936: nasce la Teoria della Computabilità
 - Perché (proprio allora)?
 - Una varietà di modelli computazionali (MdT escluse)
- 3 Macchine di Turing
 - Analisi e Definizioni principali
 - T-indecidibilità
 - La Tesi di Turing–Church: sintesi

Punto 1 – Letture consigliate.



R. Soare.

The History and Concept of Computability.

in: (E. Griffor, Ed.) *Handbook of Computability Theory.*
Elsevier, 1999.

Ricorsività: il concetto intuitivo.

- modo di definire funzioni: **per** induzione
- strettamente legato alla peculiare caratterizzazione *genetica* di \mathbb{N}
- ma estendibile ad altri tipi di collezioni
- definire $f(x)$ facendo uso dei valori precedenti $\{f(y) \mid y < x\}$ (e di certe funzioni “date”)
- tipo più semplice: definizione per recursione primitiva

$$\begin{cases} f(x, 0) = g(x) \\ f(x, y') = h(x, y, f(x)) \end{cases}$$

Hermann Graßmann.

- *Lehrbuch der Arithmetik* (1861)
 - «*L'aritmetica è la scienza che tratta delle grandezze che si ottengono da una sola grandezza per mezzo di un'operazione*»
 - definizione per recursione primitiva di somma e prodotto:

$$\begin{cases} x + 0 = x \\ x + y' = (x + y)' \end{cases}$$

$$\begin{cases} x \cdot 0 = 0 \\ x \cdot y' = (x \cdot y) + x \end{cases}$$

Richard Dedekind.

- *Was sind und was sollen die Zahlen* (1888)
 - caratterizzazione categorica (insiemistica) della struttura

$$\mathbb{N} = \langle N, <, s, 0 \rangle$$

- dimostrazione del **Teorema di recursione aritmetica**
(serve per provare la categoricità)

ecc. ecc.

- G. Peano, 1889
- T. Skolem, 1923
- D. Hilbert, 1926 (e W. Ackermann)
- ⋮
- K. Gödel, 1930/31
- ⋮

La classe PR delle funzioni ricorsive primitive

PR è la più piccola classe di funzioni numeriche *totali* che:

- contiene le funzioni Z, s, p_i^n ($1 \leq i \leq n$)

$$Z(x) = 0, \quad s(x) = x+1, \quad p_i^n(x_1, \dots, x_n) = x_i$$

- è chiusa sotto *sostituzione*

$$h^m, g_1^n, \dots, g_m^n \mapsto f^n \quad f(\vec{x}) = h(g_1(\vec{x}), \dots, g_m(\vec{x}))$$

- e *recursione primitiva*

$$h^{n+2}, g^n \mapsto f^{n+1} \quad \begin{cases} f(\vec{x}, 0) = g(\vec{x}) \\ f(\vec{x}, s(y)) = h(\vec{x}, y, f(\vec{x})) \end{cases}$$

Ma c'è anche ... E. Husserl.

- *Philosophie der Arithmetik* (1891)
 - tassonomia della “**totalità di tutte le operazioni aritmetiche concepibili**”
 - operazione aritmetica = metodo di riduzione a forma normale
 - parzialità
 - operazioni *elementari* (somma, prodotto, sottrazione, divisione)
 - operazioni *superiori*, generate da schemi di costruzione
 - sostituzione (*Mischung*)
 - recursione primitiva (\rightsquigarrow successione di Ackermann)
 - inversione (*Umkehrung*)

Parole chiave

- algoritmo (es.: algoritmo euclideo per trovare $\text{MCD}(n, k)$)
- calcolare / computare
- decisione effettiva di un *problema*
N.B.: problema = insieme di domande tipo: “ n è primo?”

Ma anche:

- calcolo = ragionamento deduttivo

Ragionare = Calcolare.

Leibniz: il **calculus ratiocinator**

«Ma per tornare all'espressione dei pensieri mediante caratteri, è mia opinione che le controversie non finiranno mai e che non si potrà imporre il silenzio alle sette se non passiamo dai ragionamenti complicati a calcoli semplici, e da vocaboli di incerto e vago significato a caratteri determinati. Bisogna cioè fare in modo che ogni paralogismo non sia altro che un errore di calcolo [. . .] E non appena ciò sia stato realizzato, allorché sorgeranno controversie, non sarà necessaria una disputa tra due filosofi più di quanto lo sia tra due persone che fanno calcoli. Basterà infatti prendere in mano la penna, sedersi a una tavola per fare calcoli (ad abacos) e dirsi l'un l'altro (in presenza, se si è d'accordo, di un amico): calcoliamo! (calculemus!)»

Computabilità: il concetto intuitivo (1).

algoritmo / procedura effettiva (*)

Una “ricetta finita” (*insieme finito di istruzioni finite*) in base alla quale **un essere umano** è in grado

- partendo da un qualsiasi elemento I di una certa classe \mathbb{C} di *input simbolici finiti*
- di eseguire “*con carta e matita*” e in modo *meccanico* un processo di calcolo *discreto (step by step)* e *finito*

= una *computazione* : $I \curvearrowright \bullet \curvearrowright \bullet \curvearrowright \bullet \curvearrowright \dots \curvearrowright \mathbf{O}$

- in cui ogni transizione è *univocamente determinata* dallo stato corrente del processo in base alle istruzioni
- e al termine del quale è leggibile un ben determinato *risultato* (simbolico) \mathbf{O} .

Computabilità: il concetto intuitivo (2).

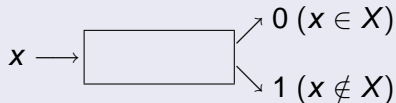
Da ora ci restringiamo ▶ (ma non è una restrizione essenziale) al caso in cui gli input/output ammissibili sono (opportune rappresentazioni simboliche dei) numeri naturali \mathbb{N}

Due prime nozioni fondamentali e **irriducibili**:

- funzione numerica *totale* effettivamente **calcolabile**

$$x_1 \dots x_n \longrightarrow \boxed{\phantom{\text{funzione}}} \longrightarrow f(x_1, \dots, x_n)$$

- insieme $X \subseteq \mathbb{N}$ **decidibile** (analogam. per $R \subseteq \mathbb{N}^n$)



Computabilità: il concetto intuitivo (3).

- N.B. 1** stiamo parlando di computabilità *idealizzata* o *in linea di principio*
- ▶ non si pongono limiti alle risorse: tempo, spazio, memoria ecc.
- N.B. 2** gli **esistenziali** (“esiste un algoritmo tale che . . .”) in generale **non sono costruttivi**
- N.B. 3** solo per motivi di cardinalità (Teor. di Cantor), ci sono 2^{\aleph_0} funzioni numeriche **non calcolabili**, e altrettanti sottinsiemi di \mathbb{N} **non decidibili!**

Computabilità: il concetto intuitivo (4).

Totalità? [vedi (*)]

- non è una richiesta molto naturale
- rende **non effettiva** la nozione stessa di algoritmo

Computabilità: il concetto intuitivo (5).

Totalità? Un problema serio (diagonalizzazione!)

Sia \mathbf{C} un insieme di algoritmi* per il quale si disponga di una *enumerazione effettiva* (con possibili ripetizioni), ossia:

$$\mathbf{C} \equiv A_0, A_1, A_2, \dots \quad \text{dove } n \mapsto A_n \text{ è calcolabile}$$

Allora si può sempre esibire un algoritmo Δ tale che $\Delta \notin \mathbf{C}$.

Δ := “dato l’input n trova A_n , applicalo a n , aggiungi 1 al risultato e fermati”. Ossia: $\Delta[n] = A_n[n] + 1$

Se $\Delta \in \mathbf{C}$ allora $\Delta = A_d$ per un $d \in \mathbb{N}$. Ma allora

$$A_d[d] + 1 = \Delta[d] = A_d[d] \quad \text{contraddizione!}$$

Computabilità: il concetto intuitivo (6).

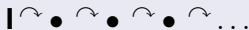
Morale: Non conviene richiedere **in anticipo** che un algoritmo sia totale (non si perde niente, anzi si semplificano le cose)

Quindi in generale, dato un algoritmo A e un input ammissibile I , ammettiamo che sia possibile uno (e uno solo) dei due casi:

- $A[I]$ produce una computazione con output O ($A[I] \downarrow O$)



- $A[I]$ produce una sequenza che prosegue all' ∞ ($A[I] \uparrow$)

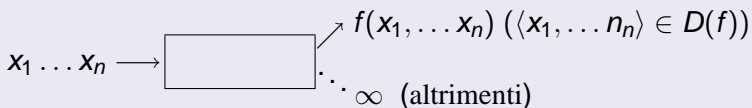


Notare come la diagonalizzazione è ora **neutralizzata** (\simeq)

Computabilità: il concetto intuitivo (7).

Una terza nozione fondamentale

- funzione numerica *parziale* effettivamente **calcolabile**

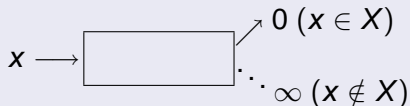


N.B.: la nozione non è banale. Non ogni f parziale e calcolabile è estendibile a una f^* totale e calcolabile

Computabilità: il concetto intuitivo (8).

Una quarta nozione fondamentale (in due versioni **equivalenti**)

- insieme $X \subseteq \mathbb{N}$ **semidecidibile**



- insieme $X \subseteq \mathbb{N}$ **effettivamente generabile (enumerabile)**



cioè esiste un algoritmo che genera uno dopo l'altro tutti e soli gli elementi di X (con possibili ripetizioni).

Computabilità: il concetto intuitivo (9).

FATTO FONDAMENTALE

Le quattro nozioni **informali** di:

- funzione totale calcolabile
- funzione parziale calcolabile
- insieme decidibile
- insieme semidecidibile / generabile

risultano essere (**argomentazione informale**) due a due **interriducibili**.

Una caratterizzazione **formale** soddisfacente di una qualunque delle quattro nozioni consente di ottenerne una altrettanto soddisfacente delle restanti tre.

Punto 2 – Letture consigliate.



S.C. Kleene.

Introductory notes a Gödel 1931 e Gödel 1934.

in: (S. Feferman et Al., Eds.) *KG Collected Works*, vol. I.
Oxford University Press, 1986.



J.R. Shoenfield.

The Mathematical Work of S.C. Kleene.

The Bulletin of Symbolic Logic, 1(1):9–40, 1995.



J.P. Seldin.

The Logic of Curry and Church.

www.cs.uleth.ca/~Seldin (2006).

Tre motivi

- ... gli accidenti della storia (p. es. i denti del giudizio di Kleene)
- specifici problemi particolarmente “resistenti” a una soluzione algoritmica
 - ↪ come si fa a dimostrare che **non esiste alcun algoritmo tale che ... ?** (soluzioni **negative**)
- la **generalizzabilità** dei teoremi limitativi di Gödel
 - ◁▷ definizione della nozione di **sistema formale**

Il X problema di Hilbert (1)

«[Problema] 10. Determination of the solvability of a Diophantine equation. Given a Diophantine equation with any number of unknown quantities and with rational integral numerical coefficients: To devise a process according to which it can be determined in a finite number of operations whether the equation is solvable in rational integers.

D. Hilbert, *Mathematische Probleme*, 1900.

Il *X* problema di Hilbert (2)

$$\begin{aligned}x - 3y + 5 &= 0 \\3x^7 + 6y^2 + z^{13} + 2u^2 &= 0 \\&\vdots\end{aligned}$$

N.B.: si ottiene un problema equivalente restringendo ai naturali sia i coefficienti che le soluzioni cercate

Risolto negativamente^{TC} da Y. Matjaševic nel 1970

L'Entscheidungsproblem (1)

«*The Entscheidungsproblem [per la logica del primo ordine] is solved when we know a procedure that allows for any given logical expression to decide by finitely many operations its validity (resp. satisfiability) The Entscheidungsproblem must be considered the main problem of mathematical logic*

The solution of the Entscheidungsproblem is of fundamental significance for the theory of all domains whose propositions could be developed on the basis of a finite number of axioms»

D. Hilbert, W. Ackermann, *Grundzüge der theoretischen Logik*, 1928.

L'Entscheidungsproblem (2)

Nota Bene: una risposta positiva avrebbe comportato

- la decidibilità algoritmica di tutte le teorie (al primo ordine) finitamente assiomatizzate
- e in particolare la possibilità di decidere algebricamente circa la loro consistenza
- e (col senno di poi) la possibilità di decidere algebricamente ciascun enunciato aritmetico di forma Π_1^0 , e.g. la **congettura di Goldbach**

Risolto negativamente^{TC} da Church e da Turing nel 1936

Teoremi di Gödel (1).

La dimostrazione che Gödel dà (1931 / abstr. 1930b) del **Teorema di incompletezza** è relativa a un un ben preciso sistema formale, ma:

Einige metamathematische Resultate ... (1930b)

«Il Teorema I vale anche per tutte le estensioni ω -consistenti del sistema S ottenute mediante aggiunta di un insieme infinito di assiomi, purché tale insieme sia decidibile [Entscheidungsdefinit]»

Teoremi di Gödel (2).

Über formal unentscheidbare Sätze ... (1931)

«In the proof of Theorem VI no properties of the system P were used besides the following:

- 1. The class of axioms and the rules of inference ... are recursively definable ...*
- 2. Every recursive relation is definable (in the sense of theorem V) in the system P .*

Therefore, in every formal system that ... »

N.B.: per Gödel, **sempre**, *ricorsivo* = *ricorsivo primitivo*.

Teoremi di Gödel (3).

Lecture Notes di Princeton (1934) — §1

[sta definendo il concetto di ‘sistema formale’]

«We require that the rules of inference, and the definitions of meaningful formulas and axioms, be **constructive**; that is, for each rule of inference there shall be a finite procedure for determining whether a given formula B is an immediate consequence (by that rule) of given formulas A_1, \dots, A_n , and there shall be a finite procedure for determining whether a given formula A is a meaningful formula or an axiom ... »

Teoremi di Gödel (4).

Lecture Notes di Princeton (1934) — §6

«[Conditions that a formal system must satisfy in order that the foregoing arguments apply]

(1) Supposing that the symbols and formulas be numbered in a manner similar to that used for the particular system considered above, **then the class of axioms and the relation of immediate consequence shall be recursive.**

This is a **precise condition** which in practice suffices as a substitute for the **unprecise** requirement of §1 that the class of axioms and the relation of immediate consequence be **constructive**.

(2) ... »

1930–36: cinque (sei) modelli computazionali estensionalmente equivalenti.

- **λ -definibilità** [Church 1933]
- **Ricorsività generale / HG-ricorsività** [Gödel 1934]
- **μ -ricorsività** [Kleene <1936 (1938)]
- **S-calcolabilità / -rappresentabilità** [\approx Gödel 1936]
- **Turing-computabilità** [Turing 1936]
- **(“Processi combinatori finiti” di Post** [Post 1936])

λ -definibilità (1)

- fine anni '20 / inizio anni '30:
A. Church elabora (più versioni di) un “sistema di logica”
con intenti fondazionali
- basato sulla nozione **type free di funzione in intensione**:
 - applicazione: $F, G \mapsto FG$ (Church: $\{F\}(G)$)
 - astrazione: $F \mapsto \lambda x.F$
 - $(\lambda x.F)G = F[x := G]$ (β -conversione)
 - le usuali **costanti logiche** sono incluse fra gli abitanti di questo universo non tipato (gestite da opportuni assiomi e regole)
- nello stesso periodo H.B. Curry elabora sistemi concettualmente molto vicini (\rightsquigarrow **Logica combinatoria**)

λ -definibilità (2)

Alla fine del 1934 S.C. Kleene e J.B. Rosser dimostrano che il sistema di Church e il più forte dei sistemi di Curry sono **contraddittori**

Paradosso di Curry (1941, 1942)

Completezza combinatoria
+
Completezza deduttiva
(un condizionale '→' decente)
sono contraddittorie

λ -definibilità (3)

Torniamo indietro \approx 1931-32

- λ := il frammento puro (= senza costanti logiche) del sistema (oggi: λ I-calcolo)
- Church sa che in λ si possono rappresentare adeguatamente i numeri naturali
- (e *indirettamente* sa anche che λ è consistente)

numerali di Church (\pm)

$$\mathbf{n} := \lambda F \lambda x. \overbrace{F(\dots(F x))}^n$$

λ -definibilità (4)

inoltre: c'è una naturale nozione di 'passo di calcolo'

$$\underbrace{\dots (\lambda x. F) G \dots}_M \rightarrow \underbrace{\dots F[x := G] \dots}_N$$

Funzione numerica totale λ -definibile

f è λ -definibile sse esiste un λ -termine F_f t.c.

$$f(n_1, \dots, n_k) = m \quad \Rightarrow \quad \vdash_{\lambda} F_f \mathbf{n}_1 \dots \mathbf{n}_k = \mathbf{m}$$

Ogni funzione λ -definibile è intuitivamente *calcolabile*

λ -definibilità (5)

- Church sa come λ -definire le funzioni s e $+$
- negli anni 1932-33 Kleene scopre gradualmente

(grazie anche all'estrazione di due (quattro?) denti del giudizio)

che la classe delle funzioni λ -definibili è molto vasta e soprattutto ha ottime proprietà di chiusura

λ -definibilità (6)

- Alla fine del '33 Church inizia a concepire l'idea che ogni funzione intuitivamente computabile sia λ -definibile

Tesi di Church, vers. I (1934)

Se si prescinde da tutti gli aspetti intensionali e ci si limita ai soli aspetti estensionali, allora il concetto intuitivo di funzione computabile può essere surrogato dal concetto matematico ben definito di funzione λ -definibile.

- Kleene non è affatto convinto
- e tanto meno lo è Gödel.

HG-ricorsività (1)

Esempio

Si consideri il **sistema finito di equazioni funzionali**:

$$\Sigma := \begin{cases} F(0, y) = y' \\ F(x', 0) = F(x, 1) \\ F(x', y') = F(x, F(x', y)) \end{cases} \quad \textit{nella sola incognita } F$$

- definizione per recursione ‘in scatolata’
(non è recursione primitiva!)
- si dimostra: c'è una sola funzione (totale) f che soddisfa Σ :
 $\models \exists! F \forall xy. \Sigma(F, x, y)$
- questa f è **computabile**: per dati n e m , $f(n, m)$ può essere calcolato da Σ in modo analogo a come si calcola $n + m$ dalla definizione di $+$ per recursione primitiva
- f **non è ricorsiva primitiva**.

HG-ricorsività (2)

K. Gödel, 1934 (J. Herbrand, 1931)

Idea:

- catturare “in questo modo” altre forme di **definizione per recursione** (tutte?)
- f è ricorsiva generale sse soddisfa un qualche sistema $\Sigma \equiv \Sigma(F, G_1, \dots, G_k)$ di equazioni funzionali e i suoi valori possono essere **calcolati formalmente** sulla base delle equazioni in Σ mediante due sole **regole**:

$$\frac{t = r}{t[x/n] = r[x/n]}^S \qquad \frac{t = r[H(\vec{n})] \quad H(\vec{n}) = \mathbf{m}}{t = r[\mathbf{m}]}_{RR} \text{ (} t \text{ chiuso)}$$

HG-ricorsività (3)

Definizione più precisa

Una funzione numerica (totale) f^k è **ricorsiva generale** (o: **HG-ricorsiva**) sse esiste un sistema finito

$$\Sigma \equiv \Sigma(F^k, \vec{G}) \equiv E_1, \dots, E_i$$

di equazioni funzionali tale che, per ogni \vec{n}, m :

$$f(\vec{n}) = m \quad \Leftrightarrow \quad \Sigma \vdash F(\vec{n}) = \mathbf{m}$$

- $r :: 0 \mid x \mid r' \mid H(\vec{r})$, $E :: r_1 = r_2$
- $\mathbf{1} := 0'$, $\mathbf{2} := \mathbf{1}'$, ...
- l'ultima equazione di Σ ha forma $F(\vec{r}) = t$.

HG-ricorsività (4)

Gödel, lettera a M. Davis, 1965

«... it is not true that footnote 3 is a statement of Church's Thesis. The conjecture stated there only refers to the equivalence of 'finite (computation) procedure' and 'recursive procedure'. However, I was, at the time of these lectures, not at all convinced that my concept of recursion comprises all possible recursions ... »

HG-ricorsività (5)

Reazione di Church (questa volta anche Kleene si è convinto; Gödel non lo sarà)

Tesi di Church, vers. II (Relazione 1935, AMS)

*«... Gödel has proposed ... a definition of the term recursive function, in a very general sense. In this paper ... it is maintained that **the notion of an effectively calculable function of positive integers should be identified with that of a recursive function**, since other plausible definitions of effective calculability turn out to yield notions that are either equivalent to or weaker than recursiveness.»*

HG-ricorsività (6)

Tesi di Church, vers. II' (*An Unsolvability Problem . . .*, 1936)

«*The purpose of the present paper is to propose a **definition** of effective calculability which is thought to correspond satisfactorily to the somewhat vague intuitive notion in terms of which problems of this class are often stated, and to show, by means of an example, that not every problem of this class is solvable . . .*

We now *define the notion . . . of an effectively calculable function* of positive integers by *identifying it with the notion of a recursive function* of positive integers (or of a λ -definable function of positive integers).»

HG-ricorsività (7)

N.B.: In Church 1936 ci sono i **primi risultati di indecidibilità**^{HG}

- della **λ -convertibilità**
- delle **teorie ω -consistenti e “sufficientemente potenti”**

μ -ricorsività

\mathcal{R}_μ è la più piccola classe di funzioni numeriche **totali** che:

- contiene le funzioni iniziali Z, s, p_i^n
- è chiusa sotto **sostituzione, recursione primitiva e minimalizzazione illimitata normale**

$$\forall \vec{x} \exists y \left[h^{n+1}(\vec{x}, y) = 0 \right] \mapsto f^n : f(\vec{x}) = \mu y (h(\vec{x}, y) = 0)$$

\mathcal{P}_μ (1938) è la più piccola classe di funzioni num. **parziali** che:

come sopra, tranne: **minimalizzazione illimitata**

$$h^{n+1} \text{ totale} \mapsto f^n : f(\vec{x}) \simeq \mu y (h(\vec{x}, y) = 0)$$

S-calcolabilità (1)

Sia S un sistema formale numerico

Una funzione numerica totale f è S -calcolabile sse

esiste una formula $A(\vec{x}, y)$ di S tale che per ogni $\vec{n}, m \in \omega$

$$f(\vec{n}) = m \quad \Leftrightarrow \quad \vdash_S A[\vec{n}, m]$$

Una funzione numerica totale f è S -rappresentabile sse

esiste una formula $A(\vec{x}, y) \dots$

$$f(\vec{n}) = m \quad \Rightarrow \quad \begin{cases} \vdash_S A[\vec{n}, m] \\ \vdash_S \forall y (A[\vec{n}, y] \rightarrow y = m) \end{cases}$$

S-calcolabilità (2)

Gödel, *Über die Länge von Beweisen*, 1936a

- funzione S-calcolabile (*berechenbar in S*)
- Osservazione finale (aggiunta in bozze):

«It can, moreover, be shown that a function computable in one of the systems S_i , or even in a system of transfinite order, is computable already in S_1 . Thus the notion 'computable' is in a certain sense 'absolute', while almost all metamathematical notions otherwise known (for example, provable, definable, and so on) quite essentially depend upon the system adopted.»

S-calcolabilità (3)

A. Tarski, A. Mostowski, R. Robinson \approx fine anni '40

Per S si può prendere il sistema formale Q , **finitamente assiomatizzato**:

$$Q.1 \quad \forall x(x \neq 0)$$

$$Q.2 \quad \forall xy(x' = y' \rightarrow x = y)$$

$$Q.3 \quad \forall xy(x + 0 = x \wedge x + y' = (x + y)')$$

$$Q.4 \quad \forall xy(x \cdot 0 = 0 \wedge x \cdot y' = (x \cdot y) + y)$$

$$Q.5 \quad \forall x(x = 0 \vee \exists y(x = y'))$$

sul linguaggio $\mathcal{L}_Q = \{0, ', +, \cdot\}$.

Equivalenze (1)

Si dimostrano:

HG \equiv \mathcal{R}_μ (Kleene, 1935)

HG \equiv λ -Def (Church e Kleene, 1936)

T-Comp \equiv λ -Def (Turing 1936)

S-Calc \equiv **HG** (...)

Equivalenze (2)

La chiave è il **Teorema della forma normale** di Kleene, 1935

Data $\mathcal{C} \dots$ si possono trovare predicati $\{T_n^{n+2}\}_{n \geq 1}$ e una funzione U^1 , primitivi ricorsivi, tali che

- Funzioni totali: per ogni $f^n \in \mathcal{C}$ esiste $e \in \omega$ t.c.

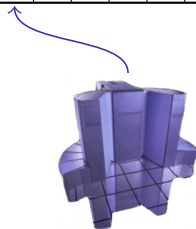
$$\text{per ogni } \vec{x} \left[\begin{array}{l} \text{esiste } y \text{ tale che } T_n(e, \vec{x}, y) \\ f(\vec{x}) = U(\mu y. T_n(e, \vec{x}, y)) \end{array} \right]$$

- Funzioni parziali: per ogni $f^n \in \mathcal{C}$ esiste $e \in \omega$ t.c.

$$\text{per ogni } \vec{x} \left[f(x) \simeq U(\mu y. T_n(e, \vec{x}, y)) \right]$$

A. Turing, *On computable numbers, with an application to the Entscheidungsproblem*, 1936–37

0	0	1	0	0	1	0	0
---	---	---	---	---	---	---	---



Logical Computing Machine = Macchina di Turing

Punto 3.1 e 3.2 - Letture consigliate.



P. Smith.

An Introduction to Gödel's Theorems.

Cambridge University Press, 2007.



W.F. Dowling.

There Are No Safe Virus Tests.

American Mathematical Monthly, 96(9):835–836, 1989.

Turing: l'analisi – “Un appello diretto all'intuizione” (1)

Analisi di una *computazione* eseguita “con carta e matita” da un *calcolatore umano* (**computer**) U a partire da certi dati iniziali

- processo che si sviluppa deterministicamente per passi discreti: **istante**₀ → **istante**₁ → **istante**₂ → ...
- **MEZZO (spazio)** sul quale U effettua operazioni di lettura/ scrittura: immagazzinamento dei dati iniziali, calcoli intermedi, risultato
 - foglio di carta dotato di una certa regolarità (quadretti, ...)
 - disponibile in quantità illimitata
 - ---→ **nastro (potenzialmente) infinito suddiviso in celle**
 - **(F.1)**: c'è un limite finito B al numero di celle che U può osservare in ogni istante
 - ⇒ *osserva esattamente una cella*

Turing: l'analisi (2)

- **ALFABETO**: una collezione di simboli usati per la rappresentazione dei dati (iniziali, intermedi, ...)
- in ogni cella può trovarsi impresso al massimo un simbolo
- **(F.2)**: l'alfabeto è **finito**
- **(F.3)**: a ogni istante della computazione, sul nastro si trovano impressi solo un numero **finito** di simboli (= “quasi tutte” le celle sono vuote)
- \Rightarrow l'alfabeto contiene un simbolo “vuoto”, *:
 - cella non scritta = cella con impresso *
 - cancellare = scrivere *

Turing: l'analisi (3)

- Un insieme di **STATI** (interni, mentali, . . .)
A ogni istante U “ricorda” quanto ha osservato / fatto agli stadi precedenti e “sa” a che punto è del suo computo
“memoria interna” + “controllo”
- ad ogni istante U si trova in esattamente uno di questi stati
- **(F.4)**: il numero degli stati è **finito**

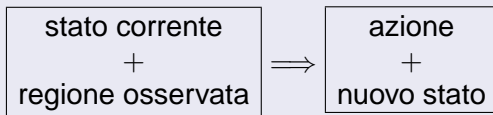
«L'uso di stati mentali più complessi può essere evitato scrivendo più simboli sul nastro» (T. 1936, §9)

Turing: l'analisi (4)

- Nel passaggio da un istante al successivo U compie una **AZIONE**, ossia effettua una modificazione del sistema:
 - scrittura / cancellazione di certi simboli su certe celle **osservate**
 - osservazione / lettura di una nuova regione (\rightsquigarrow spostamento)
 - registrazione nella memoria interna = cambiamento di stato
- **(F.5)**: ogni azione altera la situazione in modo **finito**
- **(L)**: c'è un limite finito L alla distanza di ogni spostamento
- \Rightarrow **AZ = operazione elementare + cambiamento di stato**
dove **operazione elementare** = una di:
 - **scrivere un simbolo nella cella osservata (incl. cancellare)**
 - **spostarsi di una cella a destra o a sinistra.**

Turing: l'analisi (5)

- Come sono fatte le **ISTRUZIONI** in base alle quali U effettua la transizione da un istante al successivo?
- *indipendentemente da come esse siano di fatto formulate*, segue dall'analisi precedente e dal fatto che la computazione è *deterministica* che U sta seguendo un insieme **finito** e **consistente** di istruzioni aventi il formato:



- \Rightarrow formato “a quadrupla” $\langle s, a, \alpha, s' \rangle$
o “a quintupla” $\langle s, a, b, \sigma, s' \rangle$.

Definizione formale di ‘Macchina di Turing’ (1)

(una fra le tante!)

Fissiamo tre insiemi disgiunti di simboli

- $\mathcal{S} = \{s_0, s_1, s_2 \dots\}$ simboli *di stato*
- $\mathcal{A} = \{a_0, a_1, a_2 \dots\}$ simboli *dell’alfabeto*
 - $* := a_0, \quad | := a_1$
- $\mathcal{O} = \{L, R\}$ simboli *di spostamento*

Def.: istruzione / incompatibilità tra istruzioni

- una **istruzione** q è una quadrupla $\langle s, a, \alpha, s' \rangle$ dove:
 - $s, s' \in \mathcal{S}$
 - $a \in \mathcal{A}$
 - $\alpha \in \mathcal{A} \cup \mathcal{O}$
- q e q' sono **incompatibili** se coincidono nelle prime due coordinate e differiscono in almeno una delle restanti

Definizione formale di ‘Macchina di Turing’ (2)

Def.: Macchina di Turing (MdT)

Una **macchina di Turing** M è un insieme finito e non vuoto di quadruple che soddisfa la condizione di *coerenza*:
se $q, q' \in M$ allora q e q' sono compatibili.

convenzione sul nastro e sugli input

- nastro **finito**, ma estendibile agli estremi (S/D) con una cella vuota $\boxed{*}$ ogni volta che nel corso della computazione “ M osserva” la cella più a sinistra (destra) e deve eseguire uno spostamento a sinistra (destra)
- espressione $\pi :=$ parola non vuota sull’alfabeto \mathcal{A}
- “far partire M sull’input π ” := far partire M nello stato q_0 sulla cella più a sinistra di un nastro che contiene scritta esattamente l’espressione π .

Definizione formale di ‘Macchina di Turing’ (3)

Come opera una MdT (questa Def. si può ovviamente precisare)

Viene dato un input π a una MdT M :

- 1 Siano a il simbolo osservato e s lo stato in cui si trova M (che all’inizio è per def. s_0)
 - se non ci sono istruzioni $q \in M$ che cominciano con s , a
 M si ferma
 - nel caso contrario, per la condizione di *coerenza* esiste una sola $q = \langle s, a, \alpha, s' \rangle \in M$, e M **la esegue**:
 - scrive b nella cella osservata se α è il simbolo b di \mathcal{A} , oppure si sposta di una cella a destra (sinistra) se α è il simbolo R (L) di \mathcal{O}
 - e passa nel nuovo stato s' (ovviam. può essere che $s' \equiv s!$)
- 2 se non si è fermata, procede come in (1), e così via ...

Convenzioni

- $M[\pi] \downarrow := M$, fatta partire sull'input π , si ferma dopo un numero finito di passi (eventualmente zero!)
- $M[\pi] \uparrow := \text{non}(M[\pi] \downarrow)$

rappresentazione degli I/O numerici

- per $n \in \mathbb{N}$, $\bar{n} :=$ l'espressione $\overbrace{|\dots|}^{n+1}$
- $\overline{n_1, \dots, n_k} :=$ l'espressione $\bar{n}_1 * \bar{n}_2 * \dots * \bar{n}_k$
- $M[\pi] \downarrow m$ ($m \in \mathbb{N}$) significa:
 - $M[\pi] \downarrow$, **e**
 - l'espressione scritta sul nastro al termine della computazione contiene *esattamente* m sbarre (|).

T-computabilità

una funzione numerica *totale* f^n è T-computabile

se esiste una MdT M tale che per ogni $x_1, \dots, x_n \in \mathbb{N}$:

$$M[\overline{x_1, \dots, x_n}] \downarrow f(x_1, \dots, x_n)$$

una funzione numerica *parziale* f^n è T-computabile

se esiste una MdT M tale che per ogni $x_1, \dots, x_n \in \mathbb{N}$:

$$M[\overline{x_1, \dots, x_n}] \begin{cases} \downarrow f(x_1, \dots, x_n), & \text{se } \langle x_1, \dots, x_n \rangle \in D(f) \\ \uparrow, & \text{altrimenti} \end{cases}$$

T-decidibilità e T-semidecidibilità

una relazione numerica R^n è T-decidibile

se esiste una MdT M tale che per ogni $x_1, \dots, x_n \in \mathbb{N}$:

$$M[\overline{x_1, \dots, x_n}] \begin{cases} \downarrow 0, & \text{se } R(x_1, \dots, x_n) \\ \downarrow 1, & \text{se } \neg R(x_1, \dots, x_n) \end{cases}$$

una relazione numerica R^n è T-semidecidibile

se esiste una MdT M tale che per ogni $x_1, \dots, x_n \in \mathbb{N}$:

$$M[\overline{x_1, \dots, x_n}] \begin{cases} \downarrow 0, & \text{se } R(x_1, \dots, x_n) \\ \uparrow, & \text{altrimenti} \end{cases}$$

Oss.: convenzioni relative agli I/O numerici (1)

Il formalismo di Turing è fortemente *intensionale*; di conseguenza ci sono tanti modi diversi di gestire adeguatamente gli I/O e in particolare quelli numerici (p. es.: in che formato si deve trovare il risultato di una computazione numerica?). Ognuno ha i suoi vantaggi e i suoi svantaggi; la cosa importante è che le nozioni di T-computabilità ecc. **restano invarianti**.

Le convenzioni che stiamo seguendo sono quelle “Davis – Rogers”. Uno svantaggio: non è immediato (sotto altre convenzioni sì) dimostrare alcuni fatti intuitivamente ovvi, come p. es. i seguenti

- se R è T-decidibile allora anche il complemento $\neg R$ lo è;
- ogni R T-decidibile è T-semidecidibile.

Oss.: convenzioni relative agli I / O numerici (2)

Lemma di Davis

Data una qualsiasi MdT M , si può effettivamente trovare una MdT M^* con le seguenti proprietà, dove q^* è lo stato con l'indice più grande che compare in M^* :

- nessuna istruzione di M^* inizia con q^*
- per ogni $x_1, \dots, x_n, y \in \mathbb{N}$:
 - $M[\overline{x_1, \dots, x_n}] \uparrow$ sse $M^*[\overline{x_1, \dots, x_n}] \uparrow$
 - $M[\overline{x_1, \dots, x_n}] \downarrow y$ sse $M^*[\overline{x_1, \dots, x_n}]$ si ferma
 - (i) trovandosi nello stato q^* ;
 - (ii) osservando la cella più a sinistra del nastro;
 - (iii) e l'espressione del nastro è \bar{y} .

Fatto + Tesi di Turing

Fatto

I T-concetti (T-computabilità, T-decidibilità ecc.) sono estensionalmente **inclusi** nei corrispondenti concetti intuitivi

Il converso, cioè il fatto che i T-concetti sussumano i corrispondenti concetti intuitivi, è la:

Tesi di Turing

I concetti intuitivi di *funzione computabile*, *insieme decidibile* ecc. sono, a tutti gli effetti estensionali, surrogati dai concetti matematici ben definiti di *funzione T-computabile*, *insieme T-decidibile* ecc.

Gödel sulla *Tesi di Turing*

Postscriptum (1964) a Gödel 1934

«... due to A. M. Turing work, a precise and unquestionably adequate definition of the general concept of formal system can now be given ...

Turing's work gives an analysis of the concept of "mechanical procedure" (alias "algorithm" or "computation procedure" or "finite combinatorial procedure"). This concept is shown to be equivalent with that of a "Turing machine" ...

(Note that the question of whether there exist finite non-mechanical procedures, not equivalent with any algorithm, has nothing whatsoever to do with the adequacy of the definition of "formal system" and "mechanical procedure"). »

Codifiche effettive (1)

Sia \mathbb{D} un dominio numerabile di “oggetti simbolici finiti” (parole non vuote su un alfabeto numerabile).

Una **codifica effettiva** di \mathbb{D} è un’applicazione

$$\ulcorner \ \lrcorner : \mathbb{D} \longrightarrow \mathbb{N}$$

con le proprietà:

- è iniettiva
- è effettiva (effettivamente calcolabile)
- anche l’inversa è effettiva: c’è un algoritmo che per ogni dato $n \in \mathbb{N}$ permette di decidere se n è un codice (ossia $n = \ulcorner d \lrcorner$ per un qualche $d \in \mathbb{D}$) e, in caso affermativo, di risalire in modo effettivo all’unico d tale che $n = \ulcorner d \lrcorner$

Codifiche effettive (2)

Fatto fondamentale: *riduzione*

Sia $\ulcorner \cdot \urcorner$ una codifica effettiva di \mathbb{D} . Per $X \subseteq \mathbb{D}$ sia

$$\ulcorner X \urcorner := \{\ulcorner d \urcorner \mid d \in X\}.$$

Allora:

- X è decidibile $\Leftrightarrow \ulcorner X \urcorner$ è decidibile
- X è semidecidibile $\Leftrightarrow \ulcorner X \urcorner$ è semidecidibile
- \vdots
- e analogamente per le funzioni.

▶ INDIETRO

Codifica effettiva delle MdT

Con una delle tante tecniche di gödelizzazione è possibile definire una codifica **effettiva** $\ulcorner \urcorner$ delle MdT.

per esempio

- $g(L) := 2$, $g(R) := 3$
- $g(a_i) := 4 + 2i$, $g(s_i) := 5 + 2i$ ($i \geq 0$)
- per $q \equiv \langle s, a, \alpha, s' \rangle$:

$$g(q) := 2^{g(s)} \cdot 3^{g(a)} \cdot 5^{g(\alpha)} \cdot 7^{g(s')}$$

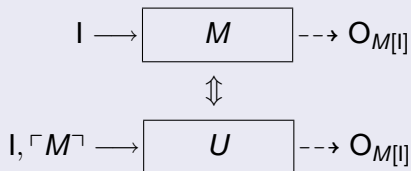
- infine per una MdT $M \equiv \{q_0, \dots, q_n\}$:

$$\ulcorner M \urcorner := \prod_{i=0}^n p_i^{g(q_i)}$$

La Macchina Universale (Turing 1936)

- $M \sim$ stringa finita di simboli \sim dato
 \rightsquigarrow **omogeneità programmi – dati (I/O)**
- L'esecuzione di un algoritmo su un certo input *ha a sua volta natura algoritmica*
 \rightsquigarrow **“algoritmo generale di esecuzione”**

(con le opportune precisazioni, v. più avanti) — **Teorema:**
 si può costruire una MdT U tale che per ogni MdT M e ogni I



Indecidibilità del *problema della fermata* (1)

Problema generale della fermata

Decidere, per ogni data MdT M e input π , se $M[\pi] \downarrow$

Problema speciale della fermata

Decidere, per ogni data MdT M , se $M[\ulcorner M \urcorner] \downarrow$

Ovviamente:

- una risposta positiva al problema **generale** comporta una risposta positiva al problema **speciale**
- e quindi per contrapposizione una risposta negativa al problema **speciale** comporta una risposta negativa al problema **generale**

Indecidibilità del *problema della fermata* (2)

Per semplicità, assumiamo qui che la codifica delle MdT sia **biiettiva** (la nostra non lo è, ma è facile definirne una che lo è)

- $\mathbf{H} := \{M \in \text{MdT} \mid M[\ulcorner M \urcorner] \downarrow\}$ (Probl. spec. della fermata)
- $\overline{\mathbf{H}} := \{M \in \text{MdT} \mid M[\ulcorner M \urcorner] \uparrow\}$ ($\overline{\mathbf{H}} = \text{MdT} \setminus \mathbf{H}$)
- cosicchè (per suriettività di $\ulcorner \urcorner$) $\ulcorner \overline{\mathbf{H}} \urcorner = \overline{\ulcorner \mathbf{H} \urcorner}$ e dunque $\ulcorner \mathbf{H} \urcorner$ e $\ulcorner \overline{\mathbf{H}} \urcorner$ sono \mathbb{N} -complementari.

Dimostreremo che $\ulcorner \overline{\mathbf{H}} \urcorner$ **non è T-semidecidibile**. Segue per semplici fatti già discussi che il complemento di $\ulcorner \overline{\mathbf{H}} \urcorner$, cioè $\ulcorner \mathbf{H} \urcorner$, **non è T-decidibile**. Da questo, **per la Tesi di TC**, segue che $\ulcorner \mathbf{H} \urcorner$ **non è decidibile** il che finalmente implica, per l'effettività della codifica, che **H non è decidibile**.

Indecidibilità del *problema della fermata* (3)

Teorema. $\neg H$ non è T-semidecidibile

Dimostrazione *per assurdo*.

Esista una MdT Δ che T-semidecide $\neg H$, e quindi per ogni n :

$$\Delta[\bar{n}] \begin{cases} \downarrow 0, & \text{se } n \in \neg H \\ \uparrow, & \text{se } n \notin \neg H \end{cases}$$

(*) per ogni n : $\Delta[\bar{n}] \downarrow$ sse $\Delta[\bar{n}] \downarrow 0$ (per costruzione)

$$\bullet \Delta[\overline{\neg \Delta}] \downarrow 0 \Rightarrow \neg \Delta \notin \neg H \Rightarrow \Delta[\overline{\neg \Delta}] \uparrow$$

$$\bullet \Delta[\overline{\neg \Delta}] \uparrow \Rightarrow \neg \Delta \in \neg H \Rightarrow \Delta[\overline{\neg \Delta}] \downarrow 0$$

► **contraddizione !**

Soluzione **negativa** dell'*Entscheidungsproblem*

[Turing 1936, Church 1936]

Solo l'idea (Turing):

- ad ogni MdT M si associa **in modo effettivo** una formula chiusa A_M nel linguaggio della logica dei predicati del primo ordine
- in modo tale che si dimostri, per M qualsiasi

$$M[\ulcorner M \urcorner] \downarrow \quad \Leftrightarrow \quad \models A_M.$$

- cosicché se fosse possibile decidere algebricamente la classe delle formule valide sarebbe possibile decidere algebricamente $\mathbf{H} := \{M \in \text{MdT} \mid M[\ulcorner M \urcorner] \downarrow\}$
contro l'indecidibilità del Problema della Fermata

Codifica biiettiva delle MdT

- sfruttando la codifica effettiva $\ulcorner \urcorner$ non è difficile definire una **biiezione effettiva** e_n da \mathbb{N} sull'insieme delle MdT. D'ora in avanti scriveremo M_k invece di $e_n(k)$. Dunque

$$M_0, M_1, M_2 \dots$$

è una **enumerazione effettiva e senza ripetizioni** dell'insieme delle MdT.

- analogamente a quanto fatto con $\ulcorner \urcorner$ possiamo ora associare a ciascun insieme \mathcal{M} di MdT l'insieme di **numeri**:

$$\mathcal{M}^\# := \{n \in \mathbb{N} \mid M_n \in \mathcal{M}\}$$

- che risulta (nozioni intuitive!)
 - decidibile sse \mathcal{M} è decidibile ,
 - semidecidibile sse \mathcal{M} è semidecidibile ,
 - ecc.

Indici

Sia $e \in \mathbb{N}$. Grazie alla codifica delle MdT possiamo, per ogni $k \geq 1$, vedere e come *indice* di una funzione parziale k -aria T-computabile. Più esattamente

- per ogni $e \geq 0$ e $k \geq 1$, sia $\{e\}^k$ la funzione numerica parziale k -aria tale che per ogni $x_1, \dots, x_k \in \mathbb{N}$:

$$\{e\}^k(x_1, \dots, x_k) = \begin{cases} y, & \text{se } M_e[\overline{x_1, \dots, x_k}] \downarrow y \\ \uparrow, & \text{altrimenti.} \end{cases}$$

Segue immediatamente dalla definizione che una funzione parziale k -aria f è T-computabile se e solo se $f = \{e\}^k$ per un qualche $e \in \mathbb{N}$.

*** Enumerazione (Macchina Universale)

[Turing, Kleene]

Per ogni $k \geq 1$ esiste un numero $u \in \mathbb{N}$ tale che per ogni $e, x_1, \dots, x_k \in \mathbb{N}$:

$$\{u\}^{k+1}(e, x_1, \dots, x_k) \simeq \{e\}^k(x_1, \dots, x_k)$$

e dunque (**Macchina universale**):

$$\begin{aligned} M_u[\overline{e, x_1, \dots, x_k}] \downarrow y &\Leftrightarrow M_e[\overline{x_1, \dots, x_k}] \downarrow y \\ M_u[\overline{e, x_1, \dots, x_k}] \uparrow &\Leftrightarrow M_e[\overline{x_1, \dots, x_k}] \uparrow \end{aligned}$$

Punto 3.3 - Letture consigliate.



W. Sieg e J. Byrnes.

K-graph Machines: Generalizing Turing's Machines and Arguments.

in: (P. Hajek, Ed.) *Gödel 96.*

Lecture Notes in Logic, vol. 6, Springer, 1996.



M. Davis.

Why Gödel Didn't Have Church's Thesis?.

Information and Control, 54:3–24, 1982.



S.C. Kleene.

Reflections on Church's Thesis.

Notre Dame Journal of Formal Logic, 28(4):490–498, 1987.



W. Sieg.

Step by Recursive Step: Church's Analysis of Effective Computability.

Bulletin of Symbolic Logic, 3(2):154–180, 1997.



W. Sieg.

Church Without Dogma – Axioms for Computability.

www.hss.cmu.edu/philosophy/sieg..., 2006.

Sintesi dei punti relativi alla Tesi di Turing–Church

- argomenti a favore di TTC
 - induttivo: assenza di controesempi (debole)
 - invarianza o convergenza (e se ci fosse un “errore sistematico”?)
 - cogenza dell’analisi di Turing
- dimostrare TTC sotto opportuni **postulati sul computer?**
- cosa non dice la TTC