Calcoli: da Aristotele a Turing.

Elio Giovannetti
Dipartimento di Informatica
Università di Torino
corso Svizzera 185 – Torino

Parte 2 – Da Boole a Turing.

Sommario (1)

- Geometrie non euclidee e "crisi dei fondamenti".
- Nascita del metodo assiomatico.
- Problemi della coerenza e della completezza:
  il programma di Hilbert.
- Il problema della decisione (Entscheidungsproblem).
- Church dimostra l'insolubilità dell' Entscheidungsproblem secondo una definizione rigorosa di "calcolabilità" da lui formulata per mezzo di un sistema di calcolo detto λ-calcolo, basato sul concetto di funzione come concetto primitivo.
- Gödel, per dimostrare i suoi teoremi di incompletezza, definisce la nozione di "ricorsione primitiva".
- Estendendo tale nozione, Gödel e Kleene formulano due altre definizioni rigorose di "calcolabilità".

Sommario (2)

- Turing dà un'altra, completamente diversa, definizione di "calcolabilità" e dimostra l'insolubilità, secondo questa sua definizione, dell' Entscheidungsproblem.
- Si è dimostrato che tutte le diverse definizioni di calcolabilità sono fra loro equivalenti, cioè individuano la stessa classe di funzioni.
- Si è dunque trovata la nozione precisa corrispondente alla nozione intuitiva di "calcolabilità".
- Dunque l'Entscheidungsproblem è insolubile in senso assoluto.
- Esistono altri problemi insolubili: si pone ben chiara la separazione fra ciò che è calcolabile e ciò che non lo è.

La geometria euclidea.

Per molti secoli, è stata il paradigma della certezza scientifica.

I postulati di Euclide sono veri in quanto evidenti.

Le dimostrazioni di teoremi permettono di ricavare (dedurre), partendo dai postulati, tutta un'ampia classe di enunciati altrettanto veri.

"Che cos' è la verità?" (Ponzio Pilato)

L'aggettivo "vero" e il sostantivo "verità", e i loro analoghi nelle altre lingue antiche e moderne, hanno molteplici significati.

- "vero" in italiano può significare, ad es., "autentico", cioè il contrario di "finto", "contraffatto":
  oro vero, amore vero, rolex vero, ecc.
- così "falso" può significare "finto", "contraffatto":
  falso oro, falso amore, rolex falso, ecc.
- "true" (vero) in inglese significa anche "fedele":
  But I'm always true to you, in my fashion (Ella Fitzgerald)
  (ma ti sono sempre fedele, a modo mio)
- "faux" (falso) in francese significa anche "sbagliato":
  le numéro de téléphone que tu m'as donné est faux
  (il numero di telefono che mi hai dato è sbagliato)

Logica: significato delle parole "vero" e "falso".

"True and False are attributes of speech, not of things.
And where speech is not, there is neither Truth or Falsehood."

Thomas Hobbes

Parole e cose: verità come corrispondenza coi fatti.

- La nozione di verità che interessa la logica è una proprietà di proposizioni o enunciati, cioè di particolari espressioni linguistiche, non una proprietà di "cose" in generale, se non nel senso che anche le frasi che diciamo o scriviamo sono "cose".
- Essa ha subito, nel corso della storia dall'antichità fino ai giorni nostri, numerose trasformazioni sia in filosofia che in logica, in matematica, e nelle scienze.
- Partiamo da una nozione di verità nel senso comune.
  Una proposizione (o enunciato, o asserzione) enuncia che nella realtà certe cose stanno in un certo modo:
  allora la proposizione si dice vera se nella realtà le cose stanno davvero (!) in quel modo, si dice falsa se nella realtà le cose non stanno in quel modo.
  Nella realtà le cose o stanno in quel modo oppure no, quindi una proposizione è o vera o falsa (principio di bivalenza).

Verità come corrispondenza, nella logica moderna.

- Nella logica del '900 la tradizionale nozione di verità come corrispondenza è stata ripresa dal logico Alfred Tarski:

  La proposizione "la neve è bianca" è vera
  se e solo se la neve è bianca, altrimenti è falsa.
  La proposizione "Paolo ama Francesca" è vera se e solo se Paolo ama Francesca, altrimenti è falsa.
  (Attenzione alle virgolette!)

  Il grande logico contemporaneo J.-Y. Girard l'ha chiamata "la piatta banalità della semantica(1) tarskiana".

- In realtà Tarski diede di tale nozione una definizione tecnica rigorosa per mezzo della teoria degli insiemi, e la semantica tarskiana è servita di base al grande sviluppo della logica matematica nel '900, in particolare di alcune sue aree.

(1) Nota: la semantica è la teoria del significato

Altre culture, altri concetti?

- Secondo alcuni studiosi, nel pensiero di una grande civiltà quale quella cinese non vi è una nozione esattamente corrispondente a quella occidentale di verità.
- I Cinesi non avrebbero "l'ossessione occidentale per la (ricerca della) verità", sarebbero interessati piuttosto a raggiungere la "saggezza", l'armonia con il mondo.
- Altri studiosi contestano l'affermazione di una radicale alterità della cultura cinese rispetto alle culture occidentali (vedasi ad esempio l'accesa polemica sviluppatosi anni fa in area francofona fra il filosofo e sinologo francese François Jullien e il sinologo svizzero Jean-François Billeter).
- La logica matematica e più in generale le scienze e le tecnologie moderne che oggi si studiano e si usano in Cina sono comunque le stesse che in tutto il mondo.

Percorsi tra logica e informatica

Peraltro anche nella logica matematica si sono intrapresi, nel contatto con l'informatica, percorsi diversi da quelli tradizionali.

Ad esempio la semantica tarskiana, basata sulla distinzione fra linguaggio e meta-linguaggio, cioè fra il linguaggio oggetto di studio e il (meta-) linguaggio in cui si parla di tale linguaggio oggetto, viene oggi da qualcuno messa in discussione

(vedi Appendice).

Nota linguistica

Una contraddizione è un enunciato della forma

A e non A

Una teoria è contradditoria se i suoi princìpi-base (diremmo meglio assiomi) sono direttamente uno la negazione di un altro (il che sarebbe un po'strano …) oppure se da essi si può derivare (cioè dedurre) una contraddizione, cioè appunto un enunciato e la sua negazione.

Una teoria si dice, come è ovvio, non-contradditoria se non è contradditoria, cioè se in essa non si può derivare (dedurre) una contraddizione.

Un sinonimo molto usato di "non-contradditorio" è "coerente", abbastanza in accordo con l'uso della parola nel linguaggio comune. Analogamente, il sostativo "coerenza" è di solito usato invece di "non-contradditorietà" (anche perché più corto!)

Nota linguistica 2

Si sono anche diffusi, come sinonimi di "coerente" e "coerenza", gli inglesismi "consistente" e "consistenza" (ingl. "consistent" e "consistence"), che nel linguaggio comune hanno un altro significato (una salsa consistente …).

Anche in francese, accanto ai termini "cohérent" e "cohérence" si sono ampiamente diffusi "consistant" e "consistance".

In queste slides ho scelto, anche per ragioni didattiche, di non usare i due termini derivati dall'inglese, ma è bene conoscerli.

Coerenza della geometria: prima dell'Ottocento.

- I postulati della geometria euclidea sono coerenti, cioè da essi non sono derivabili (deducibili) contraddizioni, perché sono veri, cioè descrivono la realtà (la realtà in sé per Newton, la realtà fenomenica per Kant), e la realtà non è contradditoria (in senso logico): le cose o stanno in un certo modo o non stanno in quel modo.
- Il problema della coerenza della geometria, fino all'Ottocento, neppure sfiorava le menti dei matematici e dei filosofi.
- La geometria euclidea era considerata da secoli l'esempio paradigmatico della conoscenza certa, perché il metodo deduttivo, partendo da fatti evidentemente veri, permetteva di ricavare solo conseguenze vere.
- Il quinto postulato, in realtà, a molti era sembrato non così evidente, perché aveva a che fare con l'infinito.

Le geometrie non euclidee

- Nell'Ottocento la creazione delle geometrie non-euclidee, ottenute negando il Quinto Postulato e (nel caso della geometria ellittica) modificando alcuni degli altri postulati, costituirono quindi una vera rivoluzione scientifica e filosofica, che distruggeva certezze secolari.
- Gli inventori delle geometrie non euclidee mostrarono subito che esse hanno dei modelli all'interno della geometria euclidea, cioè che gli enunciati di tale teoria possono essere interpretati, assegnando opportuni ma naturali significati alle parole, come enunciati veri su particolari oggetti della geometria euclidea considerati isolatamente.
- Un esempio molto naturale è quello dell'interpretazione della geometria ellittica su una superficie sferica, come all'incirca quella della terra: le rette sono i cerchi massimi (se sulla terra si va "diritti" si segue appunto un cerchio massimo), ecc.

Geometrie non euclidee: coerenza relativa.

- La coerenza delle geometrie non euclidee è assicurata proprio dal fatto che esse hanno dei modelli nella geometria euclidea, cioè che descrivono "pezzi di realtà" globalmente descritta dalla geometria euclidea.

  Se quindi da esse derivasse una contraddizione, essa sarebbe derivabile anche nella geometria euclidea.
- È ciò che si chiama una "dimostrazione di coerenza relativa": le varie geometrie non euclidee sono coerenti, se lo è la geometria euclidea (cosa di cui, come abbiamo detto, nessuno dubitava).

Definizioni degli enti primitivi.

- Negli Elementi di Euclide ci sono ben 23 definizioni di entità primitive, fra cui alcune diventate famose:

α΄. Σημεῖόν ἐστιν, οὗ μέρος οὐθέν.

1. Punto è ciò in cui non vi sono parti.

β΄. Γραμμὴ δὲ μῆκος ἀπλατές.

2. Linea è lunghezza senza larghezza.

...

- Per ognuna delle geometrie non euclidee non vi era un solo modello possibile, ma in generale più d'uno: una geometria era perciò in grado di descrivere realtà diverse; poteva avere interpretazioni diverse. Allora che cosa sono gli enti primitivi dipende dall'interpretazione.

Geometria proiettiva.

- Nell'Ottocento ha un ruolo importante la geometria proiettiva (all'ingrosso, la geometria della prospettiva pittorica), che si ottiene dalla geometria euclidea aggiungendo dei "punti all'infinito".
- In geometria proiettiva vale un principio di dualità: da ogni enunciato vero si può ottenere un altro enunciato vero scambiando fra loro le parole "punto" e "retta", le locuzioni "intersecarsi in" e "passare per", ecc. Esempio:

  Per due punti distinti passa una e una sola retta.

  Due rette distinte si intersecano in uno e un sol punto (se le rette sono parallele il punto di incontro è "il punto all'infinito", come appunto nella prospettiva).
- Il significato degli enti primitivi è perciò interscambiabile.

Abolire le definizioni degli enti primitivi

- Si rafforza allora l'idea che, in una teoria, le definizioni intuitive e informali degli enti primitivi non devono giocare alcun ruolo nelle dimostrazioni. Quindi, possono essere eliminate.
- Che cosa siano gli enti primitivi è irrilevante. Ciò che ha significato non sono i concetti, ma le relazioni fra i concetti (G. Lolli). Si afferma una sorta di ontologia relazionale.
- Moritz Pasch, *Vorlesungen über neuere Geometrie* (Lezioni di geometria moderna), 1882.
  La geometria descrive la realtà, tuttavia:
  - i concetti-base, cioè gli enti primitivi, non possono essere definiti, ma solo mostrati nella realtà naturale;
  - "il processo di deduzione deve essere in tutte le sue parti indipendente dal significato dei concetti geometrici, così come deve essere indipendente dai disegni".

Giuseppe Peano, Torino, 1889

Una presentazione assiomatica della geometria (proiettiva), in un linguaggio molto formalizzato:

I principii di geometria, logicamente esposti.

Alexander Ziwet

I

PRINCIPII DI GEOMETRIA

LOGICAMENTE ESPOSTI

SAGGIO

DI

GIUSEPPE PEANO

Professore nella R. Accademia militare
libero docente nella R. Università di Torino.

TORINO
FRATELLI BOCCA EDITORI
LIBRAI DI S. M. IL RE D'ITALIA

SUCCURSALI
ROMA
Via del Corso, 216-217
FIRENZE
Via Cerretani, 8

DEPOSITI
PALERMO
Università, 12
(N. Carosio)
MESSINA
(Daly)
CATANIA
S. Maria al Ros.°, 23
(N. Carosio)

1889

da G. Peano, Principii di geometria logicamente esposti

§ 1. Punto e segmento.

Il segno **1** leggasi *punto.*

Il segno $=$, fra due punti, indica la loro identità (coincidenza).

Si ha così una categoria di enti, chiamati punti. Questi enti non sono definiti. Inoltre, dati tre punti, si considera una relazione fra essi, indicata colla scrittura $c \in ab$, la quale relazione non è parimenti definita. Il lettore può intendere col segno **1** una categoria qualunque di enti, e con $c \in ab$ una relazione qualunque fra tre enti di quella categoria; avranno sempre valore tutte le definizioni che seguono (§ 2), e sussisteranno tutte le proposizioni del § 3. Dipendentemente dal significato attribuito ai segni non definiti **1** e $c \in ab$, potranno essere soddisfatti, oppure no, gli assiomi. Se un certo gruppo di assiomi è verificato, saranno pure vere tutte le proposizioni che si deducono, non essendo queste proposizioni che trasformazioni di quegli assiomi e delle definizioni.

Una pagina a caso ...

28. $r \in \mathbf{2} . a \in \mathbf{1} . a - \in r . b \in r . c \in b'a . d \in a'r : \supset . d \in c'r.$

$\{\text{Hp} . \supset . r \cap ad - = \Lambda.$ (α)

$\text{Hp} . e \in r . e \in ad . e = b : \supset : d \in a'b . a'b = c'b : \supset . \text{Ts.}$ (β)

$\text{Hp} . e \in r . e \in ad . e - = b : \supset . a, b, d - \in \text{Cl} . r = (be)'' . b'e \cap dc$
$- = \Lambda : \supset : r \cap dc - = \Lambda : \supset . \text{Ts.}$ (γ)

$\text{Hp} . (\alpha)\ (\beta)\ (\gamma) : \supset \text{Ts.}\}$

29. $r \in \mathbf{2} . a \in \mathbf{1} . a - \in r . b \in r . c \in b'a : \supset . a'r \supset c'r$ $\{\text{P29} = \text{P28}\}$

30. $r \in \mathbf{2} . c \in \mathbf{1} . c - \in r . a \in cr : \supset . a'r \supset c'r.$ $\{\text{P30} = \text{P29}\}$

31. $a, b, c \in \mathbf{1} . - \text{Cl} . p \in abc : \supset . p'abc \supset a \cup \ldots \cup ab \cup \ldots \cup a'b \cup \ldots \cup$
$abc \cup a'bc \cup \ldots \cup a'b'c \cup \ldots$

$\{\text{Hp} . \supset . p'pa = pa \cup a \cup p'a \supset abc \cup a \cup b'c'a.$ (α)

$\text{Hp} . \supset . p'pab = pab \cup ab \cup p'ab \supset abc \cup ab \cup p'(ab)''.$ (β)

$\text{Hp} . \supset . p'(ab)'' \supset c'(ab)''.$ (γ)

$\text{Hp} . (\alpha)\ (\beta)\ (\gamma) : \supset \text{Ts.}\}$

32. $a, b, c \in \mathbf{1} - \text{Cl} : \supset . (abc)'' = a \cup b \cup c \cup ab \cup ac \cup bc \cup a'b \cup ab'$
$\cup b'c \cup cb' \cup c'a \cup ac' \cup abc \cup a'bc \cup b'ca \cup c'ab \cup a'b'c \cup b'c'a$
$\cup c'a'b.$ $\{\text{P22} . \text{P31} : \supset . \text{P32}\}$

Gino Fano

matematico insigne, professore all'Università di Torino
dal 1901 al 1938

Sui postulati fondamentali della geometria proiettiva in uno spazio lineare a un numero qualunque di dimensioni, 1892

"Come base per il nostro studio assumiamo una collezione arbitraria di enti; enti che, per brevità, chiameremo punti, e ciò del tutto indipendentemente dalla loro natura."

Mario Pieri
insegnò all'Università di Torino dal 1891 al 1900

Mario Pieri

I principii della geometria di posizione composti in un sistema logico-deduttivo, 1898

"Carattere principalissimo degli *enti primitivi* d'un qualsivoglia sistema ipotetico-deduttivo è l'essere questi capaci d'interpretazioni arbitrarie, dentro certi confini assegnati dalle *proposizioni primitive* [...].

Il contenuto ideale delle parole e dei segni, che dinotano un qualche soggetto primitivo, è determinato soltanto dalle proposizioni primitive che versano intorno al medesimo:

e il Lettore ha la facoltà di annettere a quelle parole e a que' segni un significato *ad libitum*, purché questo sia compatibile con gli attributi generici imposti a quell'ente dalle proposizioni primitive."

David Hilbert (1862 -1943)

Forse il più grande dei matematici a cavallo fra '800 e '900.

David Hilbert

Si deve sempre poter dire, invece di ``punti, rette, e piani'', ``tavoli, sedie, e boccali di birra''.
(frase pronunciata alla stazione ferroviaria di Berlino nel 1891)

Lettera a Frege, 1899

"Se con i miei punti voglio intendere un qualunque sistema di enti, per esempio il sistema: amore, legge, spazzacamino , ..., allora basterà che assuma tutti i miei assiomi come relazioni tra questi enti perché le mie proposizioni, per esempio il teorema di Pitagora, valgano anche per essi."

Hilbert, Fondamenti della geometria, 1899 (Grundlagen der Geometrie)

Consideriamo tre distinti sistemi di cose. Le cose che compongono il primo sistema le chiamiamo punti e le designiamo con le lettere A, B, C, ...; quelle del secondo le chiamiamo rette e le designiamo con le lettere a, b, c, ...; e quelle del terzo sistema le chiamiamo piani e le designiamo con le lettere α, β, γ, ...

Pensiamo a questi punti, rette, e piani come aventi fra di loro certe relazioni, che indichiamo per mezzo di parole come "sono situati", "fra", "parallela", "congruente", "continuo", ecc.

La completa ed esatta descrizione di queste relazioni segue come conseguenza degli assiomi della geometria.

L'aritmetizzazione della matematica.

Con l'introduzione dei sistemi di coordinate cartesiane, i punti diventano terne di numeri reali, le linee, le superfici, e le altre entità geometriche diventano equazioni o disequazioni, e la geometria diventa una teoria dei numeri reali.

È la geometria analitica, una geometria "senza figure", ben nota agli studenti del liceo scientifico, in contrasto con la "geometria sintetica" di Euclide.

Nei Grundlagen, Hilbert dimostra che i suoi assiomi della geometria, se interpretati come enunciati della geometria analitica, sono teoremi della teoria dei numeri reali.

I numeri reali vengono definiti nell' Ottocento per mezzo dei numeri razionali (ad es., come successioni illimitate di numeri razionali, secondo Weierstrass).

I numeri razionali non sono altro che coppie di numeri interi.

La situazione si poteva dunque riassumere nella famosa affermazione del matematico Kronecker, un precursore del costruttivismo:

I numeri interi li ha fatti il Buon Dio,
tutto il resto è opera dell'uomo.

L'Aritmetica di Peano.

Nel 1889 Peano aveva pubblicato, oltre ai Principii di geometria, una presentazione assiomatica e formalizzata dell'aritmetica, in latino:

Arithmetices principia, nova methodo exposita.

ARITHMETICES PRINCIPIA

NOVA METHODO EXPOSITA

A

IOSEPH PEANO

in R. Academia militari professore

Analysin infinitorum in R. Taurinensi Athenaeo docente.

AUGUSTAE TAURINORUM

EDIDERUNT FRATRES BOCCA

REGIAE BIBLIOPOLAE

ROMAE
Via del Corso, 216-217.

FLORENTIAE
Via Cerretani, 8.

1889

§ 1. De numeris et de additione.

Explicationes.

Signo N significatur *numerus (integer positivus).*
» 1 » *unitas.*
» $a+1$ » *sequens* a, sive a *plus* 1.
» $=$ » *est aequalis.* Hoc ut novum signum considerandum est, etsi logicae signi figuram habeat.

Axiomata.

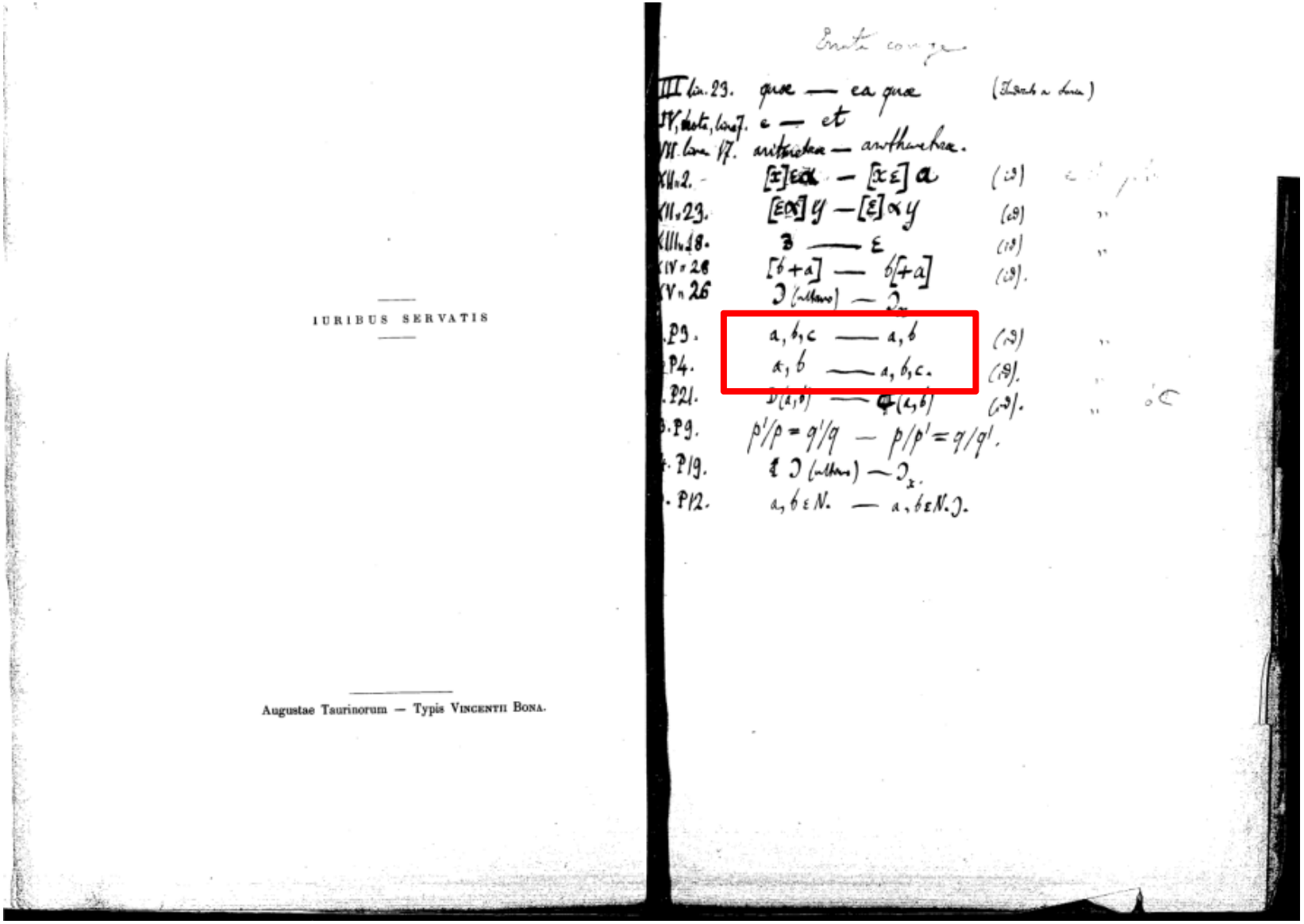
1. $1 \in \mathrm{N}$.
2. $a \in \mathrm{N} . \supset . a = a$.
3. $a, b, c \in \mathrm{N} . \supset : a = b . = . b = a$.
4. $a, b \in \mathrm{N} . \supset \therefore a = b . b = c : \supset . a = c$.
5. $a = b . b \in \mathrm{N} : \supset . a \in \mathrm{N}$.
6. $a \in \mathrm{N} . \supset . a + 1 \in \mathrm{N}$.
7. $a, b \in \mathrm{N} . \supset : a = b . = . a + 1 = b + 1$.
8. $a \in \mathrm{N} . \supset . a + 1 - = 1$.
9. $k \in \mathrm{K} \therefore 1 \in k \therefore x \in \mathrm{N} . x \in k : \supset_x . x + 1 \in k :: \supset . \mathrm{N} \supset k$.

← qui c'è un errore di stampa!

Definitiones.

10. $2 = 1 + 1;\ 3 = 2 + 1;\ 4 = 3 + 1;$ etc.

… ma ovviamente Peano se n'era accorto!



IURIBUS SERVATIS

Augustae Taurinorum — Typis Vincentii Bona.

III lin. 23. quae — ea quae (Zanichelli a linea)
IV, tabula, linea 7. e — et
VII. linea 17. antithesea — antithesea.
XII n. 2. [x]εcls — [xε]a (3)
XII n. 23. [εϰx]y — [ε]ϰy (8)
XIII n. 18. ℑ — ε (18) "
XIV n. 28 [b+a] — b[+a] (3).
XV n. 26 ℑ (alibi) — ℑ
P9. a, b, c — a, b (2) "
P4. a, b — a, b, c. (8).
P21. ϕ(a, b) — Φ(a, b) (3). "
P9. p'/p = q'/q — p/p' = q/q'.
P19. ι ℑ (alibi) — ℑ$_x$.
P12. a, b ε N. — a, b ε N. ℑ.

I numeri naturali di Peano.

I numeri di Peano sono i numeri dell'uomo preistorico:
un numero è un osso (o un bastone) su cui sono state incise delle tacche; si crea un nuovo numero incidendo una nuova tacca, e in linea di principio tale operazione può essere iterata indefinitamente.

La definizione di Peano è un esempio di definizione induttiva, che è un modo di definire un insieme infinito di elementi stabilendo:

- un insieme di enti-base appartenenti all'insieme;
- un insieme di operazioni (modi, regole) per costruire un nuovo elemento dell'insieme a partire da altri elementi.

Gli elementi dell'insieme così definito sono allora tutti e soli gli enti che si possono in tal modo costruire a partire dagli elementi-base e generando via via elementi più complessi.

I numeri naturali (preistorici).

- un osso o un bastone senza tacche è un numero naturale;
- se n è un numero naturale, incidendo su n una (altra) tacca ottengo un nuovo numero naturale.

è un numero naturale *(in realtà gli uomini preistorici , probabilmente cominciavano a contare da uno, così come del resto Peano; oggi si preferisce partire da 0)*

se  è un numero naturale, allora  è un numero naturale.

I numeri naturali preistorici in notazione moderna.

È la cosiddetta *notazione unaria*. Iniziando da uno:

a) | (oppure 1) è un numero naturale;

b) se n è un numero naturale, |n è un numero naturale (e nient'altro è un numero naturale).

Allora i numeri naturali sono: | || ||| |||| ||||| …

Se si vuole partire da zero, come richiesto nella matematica moderna, si possono usare anche nella notazione unaria due simboli (non confondere con la notazione binaria!):

0 s0 ss0 sss0 ssss0 … (la *s* vuol dire *successore*)

□ |□ ||□ |||□ ||||□ …

0 |0 ||0 |||0 ||||0 …

Peano coglie l'essenza dei numeri naturali, che sono prodotti dalla ripetizione del gesto primordiale di "aggiungere 1".

Le operazioni aritmetiche.

L'operazione di addizione è definita induttivamente per mezzo dello "aggiungere 1":

$$m + 0 = m$$
$$m + |n = |\,(m + n)$$

Il calcolo 2+3 = 5 è:

$$||0 + |||0 = |\,(||0 + ||0) = |\,|\,(||0 + |0) = |\,|\,|\,(||0 + 0) = |\,|\,|\,||0$$

La moltiplicazione è definita per mezzo dell'addizione:

$$m \times 0 = 0$$
$$m \times |n = m + m \times n$$

Modernamente scriviamo n invece di |n, e n-1 invece di n.
In Python, se la moltiplicazione non fosse già una primitiva:

```
def multiply(m, n):
  if n == 0: return 0
  else: return m + multiply(m, n-1)
```

Esercizio

Definire in Python l'operazione di elevamento di un numero x ad un esponente n intero non negativo.

```
def exp(x, n):
  if(n == 0): return 1
  else: return x*exp(x, n-1)
```

Definire in Python la funzione fattoriale:

```
n! = 1·2·3 … ·(n-1)n
```

```
def fact(n):
  if(n == 0): return 1
  else: return n*fact(n-1)
```

Nota a margine: ma i grandi interi esistono davvero?

Una corrente di matematici, detta ultrafinitismo, nega che i numeri naturali molto grandi esistano nello stesso senso in cui esistono i numeri che riusciamo a visualizzare.

Un numero come $10^{10^{10}}$, cioè 1 seguito da dieci miliardi di zeri, non potrà mai essere scritto nella notazione unaria cavernicola di Peano, perché è enormemente superiore al numero delle particelle elementari dell'universo (stimato di "solo" 10^{80}).

Per gli ultrafinitisti, i numeri tornano – in un certo senso – a coincidere con le notazioni, o almeno con classi di notazioni:

il numero $10^{10^{10}}$ è un numero di nuovo genere rispetto ai numeri di Peano, così come i numeri in notazione posizionale.

Il grande logico francese Jean-Yves Girard, in una conferenza nel 2000, ha detto, rispondendo a una domanda del pubblico: "il faudrait arriver à se débarrasser de ces foutus entiers"

La coerenza dell'aritmetica.

Non c'erano dubbi che l'aritmetica fosse coerente, per le stesse ragioni per cui non c'erano dubbi per la geometria. Peano, che nel 1903 si era inventato il Latino sine flexione, nel 1906 scriveva:

"Proba que systema de postulatos de Arithmetica, aut de Geometria, non involve contradictione, non es, me puta, necessario. Nam nos non crea postulatos ad arbitrio, sed nos sume ut postulatos propositiones simplicissimo, scripto in modo explicito aut implicito, in omni tractatu de Arithmetica, aut de Geometria. Nostro analysi de principios de ce scientias es reductione de affirmationes commune ad numero minimo, necessario et sufficiente"

"Proba de coexistentia de systema de postulatos pote es utile, si postulatos es hypothetico, et non respondentes ad factu reale".

Hilbert e la questione della coerenza.

- Nemmeno Hilbert riteneva che gli assiomi della geometria fossero arbitrari: anch'egli pensava che l'assiomatizzazione della geometria servisse alla "analisi logica della nostra nozione di spazio". Tuttavia non dava per garantito a priori che tale analisi fosse corretta.
- La teoria degli insiemi, sviluppata da Cantor, si era rivelata contradditoria, come aveva mostrato Russell col suo famoso paradosso.
- Lo sviluppo della cosiddetta algebra astratta aveva portato alla costruzione di nuovi enti matematici che non necessariamente avevano una corrispondenza nella realtà immediata.
- Hilbert ritenne allora che, per assicurare fondamenta sicure alla matematica in rapida espansione, occorresse non solo assiomatizzare le teorie, ma anche dimostrarne la coerenza.

Il programma di Hilbert e la coerenza dell'aritmetica.

- In particolare, data la centralità che era venuta assumendo l'aritmetica, per Hilbert era un compito cruciale dimostrare la coerenza dell'aritmetica. Era il secondo della lista dei 23 problemi presentata nel suo famoso discorso al Congresso dei Matematici a Parigi nel 1900.
- Essendo una (se non la) teoria matematica di base, una tale dimostrazione non poteva essere relativa (ad un'altra teoria), ma doveva essere assoluta.
- L'idea di Hilbert era di usare, per tale dimostrazione, solo una parte dell'aritmetica, che egli chiamò "matematica reale", i cui metodi dimostrativi, che chiamò finitisti, non facevano uso del concetto di infinito.
- Gli enti della matematica reale sono per Hilbert i segni (astratti, nello senso in cui è un segno astratto la lettera 'a').

Un altro problema proposto da Hilbert: lo Entscheidungsproblem (problema della decisione).

Nella formulazione di Alonzo Church:

"By the Entscheidungsproblem of a system of symbolic logic is here understood the problem to find an effective method by which, given any expression Q in the notation of the system, it can be determined whether or not Q is provable in the system"

Coerenza come criterio di esistenza.

Hilbert fu fra i principali iniziatori della filosofia "formalista" della matematica, rifiutando (almeno a parole) ogni pretesa ontologica per le teorie matematiche, che diventano delle libere costruzioni formali:

> "Se assiomi arbitrariamente stabiliti non sono fra loro in contraddizione, con tutte le loro conseguenze, allora essi sono veri, allora esistono gli enti definiti per mezzo di quegli assiomi.
> Questo è per me il criterio della verità e dell'esistenza."

Come si vede, si ha un completo capovolgimento di visione: fino ad allora gli assiomi di una teoria matematica erano giudicati non-contradditori perché veri.

Ora al contrario gli assiomi sono veri perché non contradditori (cioè se si dimostra che non sono contradditori).

Lo Entscheidungsproblem per una teoria.

Data una teoria assiomatica (del prim'ordine), esiste sempre un procedimento meccanico il quale, dato un qualunque enunciato nel linguaggio della teoria, permetta di stabilire se tale enunciato è un teorema della teoria, cioè se è derivabile dagli assiomi?

Procedimento meccanico è ciò che oggi chiamiamo *algoritmo* !

Una buona teoria assiomatica doveva inoltre essere completa(*): cioè, dato un qualunque enunciato P nel linguaggio della teoria, o P o la sua negazione devono essere dimostrabili nella teoria.

Per una teoria completa un procedimento meccanico del genere di quello richiesto esiste certamente, anche se estremamente inefficiente:

basta costruire una dopo l'altra tutte le possibili dimostrazioni in ordine crescente di lunghezza (della dimostrazione), finché si trova o la dimostrazione di P o la dimostrazione di non P.

(*) più precisamente si dice "sintatticamente completa".

Attenzione!

- Dimostrare non P è cosa molto diversa dal non dimostrare P!

Nel vecchio ordinamento giudiziario italiano l'imputato poteva venire assolto:

- "con formula piena", se si era "dimostrato" che non aveva commesso il reato;
- "per insufficienza di prove", se non si era dimostrato che aveva commesso il reato.

Esempio: si è dimostrato che (solo) uno dei due membri di una coppia ha commesso il reato, ma non si è potuto stabilire mediante dimostrazione chi dei due è stato; entrambi venivano assolti per insufficienza di prove.

La soluzione negativa dell'Entscheidungsproblem.

Negli anni '30 del Novecento si dimostrò che un siffatto procedimento di calcolo per una qualunque teoria non può esistere.

Ma per dimostrarlo fu necessario definire in modo preciso e rigoroso la nozione di "procedimento di calcolo", e quindi quella di "calcolabilità".

Scienziati diversi (Church, Gödel, Kleene, Turing) diedero definizioni fra loro completamente diverse di "procedimento di calcolo", e dimostrarono ciascuno che il procedimento richiesto dall'Entscheidungsproblem non poteva esistere in base alla propria definizione di procedimento di calcolo.

Gli stessi scienziati però dimostrarono anche che tali diverse definizioni erano tutte fra loro equivalenti, e che quindi esse verosimilmente catturavano la nozione intuitiva.

La soluzione negativa dell'Entscheidungsproblem e la definizione precisa del concetto intuitivo di calcolabilità.

Le diverse definizioni di calcolabilità corrispondevano a modi diversi, benché equivalenti, di concepire i procedimenti di calcolo.

Attraverso di esse si individuò, nella classe dei problemi algoritmici, cioè dei problemi formulabili in modo preciso, una netta distinzione fra i problemi risolubili e i problemi non risolubili, cioè i problemi per i quali è impossibile che esista un algoritmo risolvente.

Tutto ciò pose le basi teoriche e pratiche della scienza e della tecnica informatica.

Alcuni particolari della storia

Henry Poincaré, 1854-1912

Il grande matematico francese era molto critico nei confronti della logica matematica, dell'assiomatizzazione, e dei tentativi di dimostrare la coerenza dell'aritmetica con mezzi non più forti dell'aritmetica stessa.

Poincaré nel1905 ironizzava:

Ainsi c'est bien entendu, pour démontrer un théorème, il n'est pas nécessaire ni même utile de savoir ce qu'il veut dire. On pourrait remplacer le géomètre par le *piano à raisonner* imaginé par Stanley Jevons ; ou, si l'on aime mieux, on pourrait imaginer une machine où l'on introduirait les axiomes par un bout pendant qu'on recueillerait les théorèmes à l'autre bout, comme cette machine légendaire de Chicago où les porcs entrent vivants et d'où ils sortent transformés en jambons et en saucisses. Pas plus que ces machines, le mathématicien n'a besoin de comprendre ce qu'il fait.

Henri Poincaré, « Les mathématiques et la logique » in Revue de Métaphysique et de Morale, 1905, p. 815-835

traduzione:

Dunque è chiaro, per dimostrare un teorema non è necessario e nemmeno utile sapere che cosa vuol dire. Si potrebbe rimpiazzare il "geométra" (cioè il matematico studioso di geometria) con il "pianoforte logico" immaginato da Stanley Jevons; o, se si preferisce, si potrebbe immaginare una macchina dove da una parte si introdurrebbero gli assiomi, e dalla parte opposta si raccoglierebbero i teoremi: come in quella leggendaria macchina di Chicago dove i maiali entrano vivi ed escono trasformati in prosciutti e salsicce.

Il matematico non ha bisogno di sapere quello che sta facendo più di quanto ne abbiano queste macchine.

Soluzione finale ?

Nello spirito di Leibniz:

una volta individuato un insieme sufficientemente potente di assiomi per una teoria, tutti i problemi della teoria avrebbero potuto essere risolti in linea di principio meccanicamente: una "soluzione finale" ai problemi della matematica ...

... espressione che il logico francese Jean-Yves Girard oggi considera essere stata "tipica dello scientismo tedesco".

Ancora Girard: dimostrare la coerenza dell'aritmetica all'interno dell'aritmetica sarebbe stato come il parlamento francese (o italiano, diremmo noi) che vota la propria immunità, ...

"On ne revisse pas ses lunettes en les gardant sur le nez"

("non ci si può riavvitare gli occhiali tenendoli sul naso").

David Hilbert, Königsberg, 8 settembre 1930

Discorso di apertura al congresso annuale dell'Associazione degli scienziati e medici tedeschi:

Für uns gibt es kein *Ignorabimus*, und meiner Meinung nach auch für die Naturwissenschaft überhaupt nicht. Statt des törichten *Ignorabimus* heiße im Gegenteil unsere Losung:

wir müssen wissen,
wir werden wissen.

traduzione:
Per noi non ci sono *ignorabimus*, e a mio parere anche nella scienza non ce n'è assolutamente nessuno. Invece dello sciocco *ignorabimus*, sia al contrario il nostro motto:

dobbiamo sapere,
e sapremo.

Ma proprio due giorni prima, in quella stessa Königsberg, ad un altro congresso, un giovane matematico austriaco …

Kurt Gödel, Königsberg, 6 settembre 1930

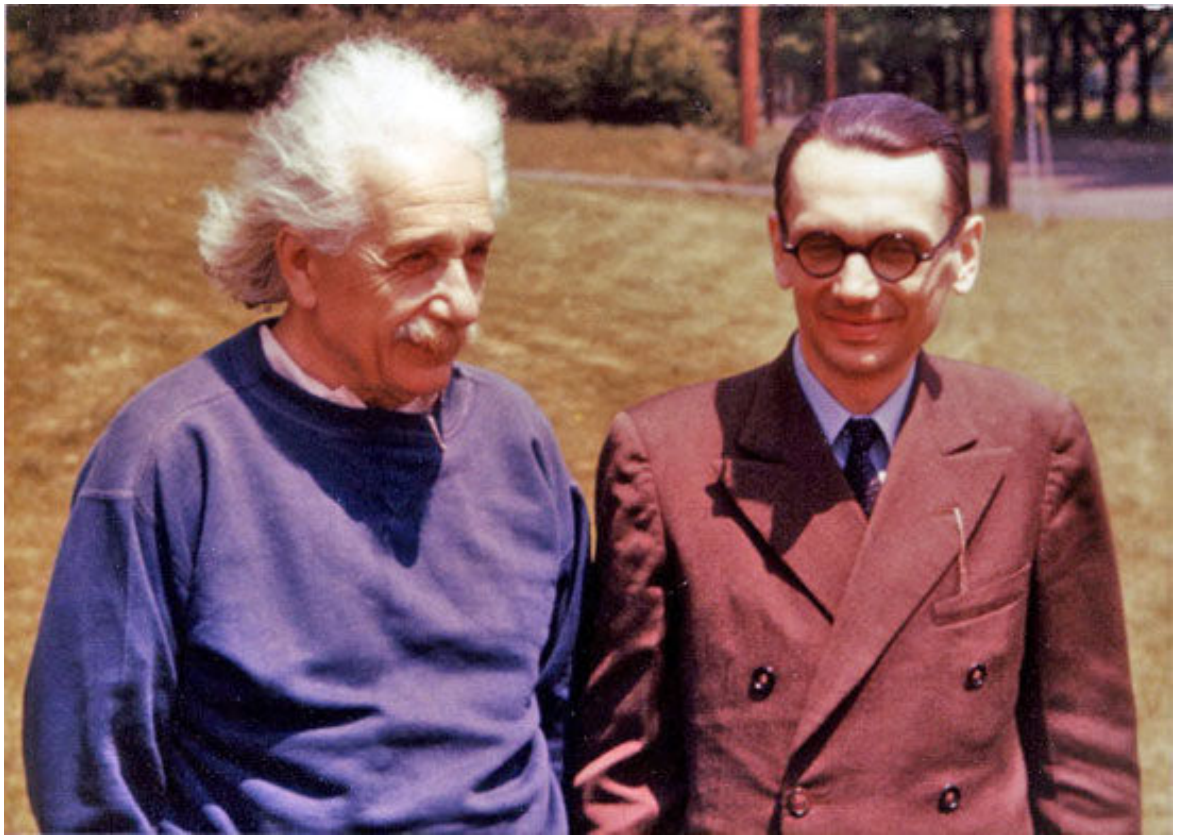
Ad una tavola rotonda del Congresso sull'epistemologia delle scienze esatte dava per la prima volta notizia di un suo nuovo risultato: la dimostrazione che qualunque sistema assiomatico dell'aritmetica che sia "abbastanza forte", se è coerente è incompleto.

È il famoso primo teorema di incompletezza, poi seguito dal secondo: se l'aritmetica è coerente, non è possibile dimostrarne la coerenza all'interno dell'aritmetica stessa.

Si può farlo usando una teoria più forte, come la teoria degli insiemi, la cui coerenza è quindi ancora più dubbia, e che a sua volta non può essere dimostrata all'interno della teoria stessa, …

Ciò verrà considerato dai più come la fine del programma di Hilbert in senso stretto.

Kurt Gödel in compagnia di un suo amico …



Incompletezza e problema della decisione.

Come abbiamo visto, se un sistema assiomatico è completo allora c'è un (sia pur inefficiente) algoritmo il quale, dato un qualunque enunciato, è in grado di stabilire se esso è un teorema in quel sistema (Entscheidungsproblem).

Se il sistema è incompleto, rimaneva comunque il dubbio che potesse comunque esistere un qualche altro algoritmo in grado di risolvere il problema.

Lo Entscheidungsproblem restava quindi aperto.

Intermezzo geografico-storico.

Dove si trova esattamente

Königsberg

la patria di Kant, di Hilbert, di Goldbach, di Kathe Kollwitz?

Era la capitale della

PRUSSIA ORIENTALE

Intermezzo geografico-storico.

Ora si chiama
Kaliningrad

e si trova in

RUSSIA (OCCIDENTALE)

Intermezzo geografico-storico.

Ora si chiama
Kaliningrad
Калинингра́д

RUSSIA

Le funzioni ricorsive primitive.

Gödel, per dimostrare i suoi teoremi di incompletezza, introduce e utilizza una ben definita classe di funzioni, che saranno poi chiamate "ricorsive primitive", che sono certamente calcolabili.

Sono le funzioni f in cui il valore per n, cioè f(n), si ottiene dal valore per n-1, cioè f(n-1), come negli esempi precedenti del fattoriale e dell'esponenziale.

La ricorsione primitiva garantisce che il calcolo termina, perché f(n) si ottiene da f(n-1) che a sua volta si ottiene da f(n-2) … e così via fino a f(0), di cui è definito il valore.

Si vede facilmente che la ricorsione primitiva equivale ad un ciclo for, in cui a partire da f(0), che è dato, si calcola il valore di f(1), da questo il valore di f(2), e così via.

Programmazione con il ciclo for

Fattoriale:

```
def factit(n):
  ris = 1
  for i in range(1,n+1):
    ris = ris*i
  return ris
```

Esponenziale:

```
def expit(x,n):
  ris = 1
  for i in range(1,n+1):
    ris = ris*x
  return ris
```

Le funzioni ricorsive generali

Nel 1934 Gödel, riprendendo un'idea del francese Herbrand, definì una forma generale di ricorsione, in cui il nome della funzione da definire può comparire nei secondi membri delle definizioni in modo arbitrario, e chiamò ricorsive generali le funzioni così definite.

Una definizione ricorsiva generale può condurre, per alcuni valori degli argomenti, a calcoli che non terminano.

Ad esempio la funzione che richiama se stessa con lo stesso argomento addirittura non termina per nessun argomento:

```
def f(n): return f(n)
```

(in realtà, su un computer reale il calcolo esaurisce lo spazio di memoria disponibile e genera un errore di "stack overflow").

Le funzione ricorsive generali coprivano un'amplissima classe di funzioni intuitivamente calcolabili, ma chi poteva garantire che fossero tutte?

Programmazione in Python: introduzione.

Disegnare un quadrato di lato L con la grafica della tartaruga:

ripeti 4 volte:
  avanti di L;
  ruota di 90°

In Python si dice:

```
def quadrato(L):
  for i in range(4):
    forward(lato)
    right(90)
```

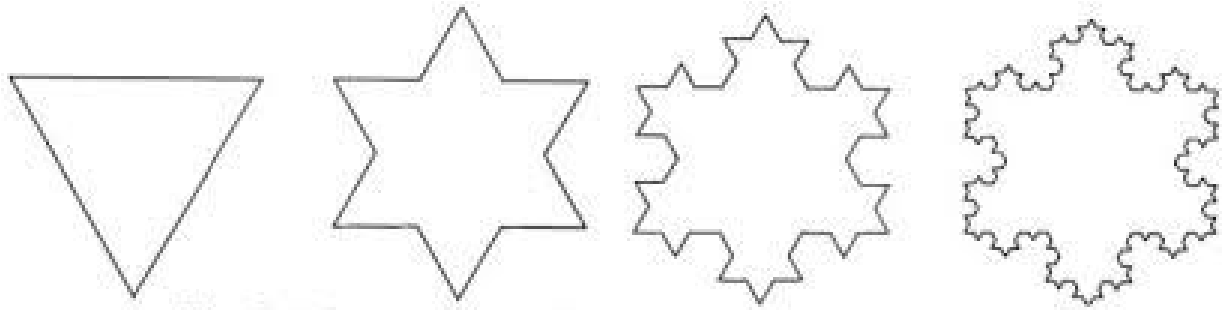
Esercizi. Definire le funzioni

```
def equitriangolo(L):
def esagonoRegolare(L):
def poligonoRegolare(L, n):
```

…

La programmazione ricorsiva: esercizi.

Disegnare il "fiocco di neve" di von Koch con la grafica della tartaruga:



The first four stages in the construction of the Koch snowflake

Bisogna ripetere tre volte il disegno del lato:

```
def fiocco(n, lato):
  for i in range(3):
    disegnaLato(n,lato)
    right( … )        di quanto bisogna ruotare?
```

Il vero problema è disegnare il lato.

Esercizio: la procedura ricorsiva disegnaLato

```
def disegnaLato(n, lato):
  n==0 or lato < 3:   scrivere la base della ricorsione
  else:
   … scrivere le istruzioni con cui
     la procedura richiama ricorsivamente se stessa
```

L'intero programma

```
from turtle import *

def disegnaLato(n, lato):
   ...
def fiocco(n, lato):
   ...

# programma principale
n = int(input("livello di ricorsione: "))
up()              # alza la penna
goto(-200,100) # spostati nel punto (-200, 100)
down()            # abbassa la penna
fiocco(n, 400) # disegna un fiocco di lato 400
exitonclick()  # se clicca, esci
```

Esercizio: la procedura ricorsiva *disegnaLato:* suggerimento per la soluzione.

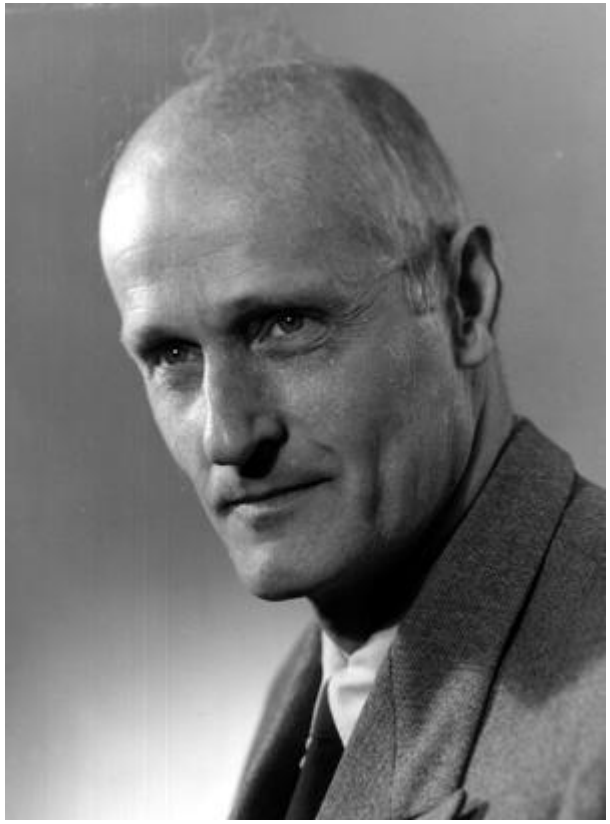
```
def disegnaLato(n, lato):
  if n==0 or lato < 3: forward(lato)
  else:
    disegnaLato(n-1, lato//3)
```

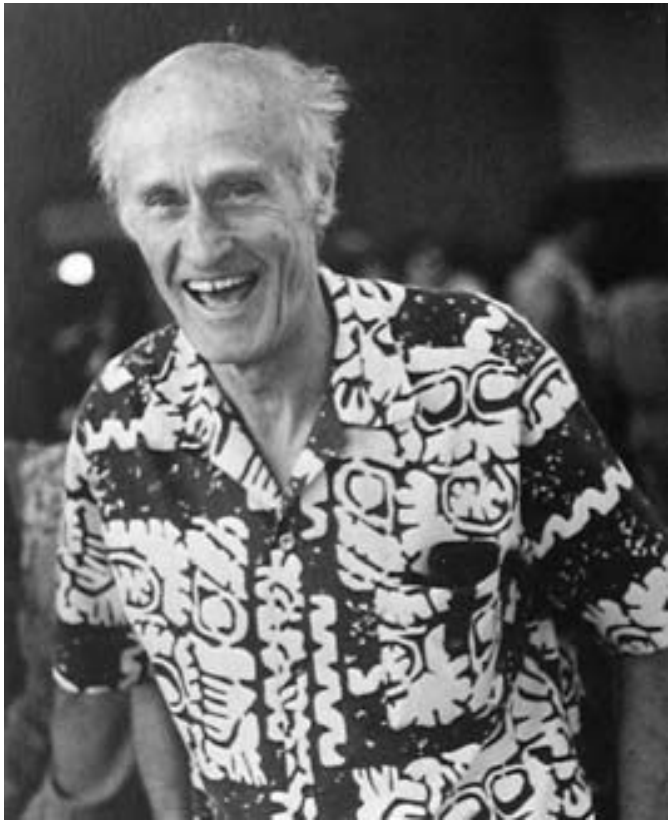
Un'altra definizione.

Nel 1936 il matematico americano Kleene diede un'altra definizione di calcolabilità, le cosiddette funzioni μ-ricorsive, che noi oggi riconosciamo analoghe alle funzioni definibili tramite cicli while, e dimostrò che tale definizione è equivalente a quella di Gödel, cioè che la classe delle funzioni μ-ricorsive coincide con la classe delle funzioni generali ricorsive di Gödel -Herbrand.

Stephen Kleene

Connecticut, 1909 – Wisconsin, 1994





Alonzo Church e il λ-calcolo

Matematico americano, nel 1932 pubblica un articolo in cui si propone di fondare la logica (e quindi la matematica) sul concetto di funzione invece che su quello di insieme:

A Set of Postulates for the Foundation of Logic

In esso faceva la sua comparsa un particolare formalismo con associato un calcolo: il λ-calcolo.

λ-termini: funzioni e niente altro.

Il linguaggio e l'associato calcolo sono di una stupefacente semplicità.

Un'espressione del linguaggio è un λ-termine, che può essere solo di tre forme:

- una variabile, come x, y, z, ecc.;
- un'applicazione di un termine ad un altro termine, che scriviamo M_1M_2, dove M_1 ed M_2 sono due termini;
- l'astrazione di un termine rispetto a una variabile: $\lambda x.M$

In forma concisa, indicando con M, M_1, M_2 dei generici termini e con x una qualunque variabile:

$$M ::= x \mid M_1M_2 \mid \lambda x.M_1$$

Non ci sono costanti, simboli numerici, nomi di funzioni, …: solo termini che rappresentano funzioni, e nient'altro!

Esempi

λx.x : è la funzione identità: prende un qualunque oggetto x e dà come risultato quell'oggetto stesso.

λx λy.x : è una funzione che prende un qualunque oggetto x e dà come risultato la funzione λy.x che prende un qualunque oggetto y, non lo utilizza ("lo butta via"), e dà come risultato l'oggetto x.

λx λy.y : è una funzione che prende un qualunque oggetto x, non lo utilizza, e dà come risultato la funzione identità λy.y.

λx λy.x(x y) : è una funzione che prende un qualunque oggetto x e dà come risultato una funzione che prende un oggetto y e gli applica due volte l'oggetto x (tutti gli oggetti sono funzioni)

Come si vede, le funzioni del λ-calcolo sono "funzioni di ordine superiore", cioè possono prendere come argomenti altre funzioni e dare come risultato delle funzioni.

Regola di calcolo

Il calcolo è costituito da una sola regola, la β-regola:

$$(\lambda x.M)N \rightarrow M[x := N]$$

dove la scrittura $M[x := N]$ denota il termine che si ottiene da M sostituendo in esso la variable x, in tutti i punti in cui compare, con il termine N.

Esempio:

$(\lambda x\ \lambda y.x(x\ y))v\ z \rightarrow (\lambda y.v(v\ y))z \rightarrow v(v\ z)$.

Ci sono termini che hanno uno strano comportamento, ad esempio il termine $(\lambda x.xx)\ (\lambda x.xx)$ riproduce indefinitamente se stesso, dando così origine a un calcolo che non termina …

Ma che senso ha un calcolo in cui ci sono solo funzioni ma non ci sono elementi base su cui calcolare, nemmeno i numeri?

Risposta: anche i numeri sono funzioni!

Se si assume come fondamento della matematica la teoria degli insiemi, come si fa di solito, un numero naturale è un insieme (in particolare, 0 è l'insieme vuoto, 1 è un insieme che ha un elemento, e così via).

Se si assume come fondamento della matematica il concetto di funzione considerato come concetto primitivo, un numero naturale può essere definito come una funzione che itera l'applicazione di una funzione ad un argomento:

$0 = \lambda f.\ \lambda x.x$

$1 = \lambda f.\ \lambda x.(fx)$

$2 = \lambda f.\ \lambda x.f(fx)$

$3 = \lambda f.\ \lambda x.f(f(fx))$

…

$n = \lambda f.\ \lambda x.f(f(f \ldots (fx) \ldots))$ cioè, in forma abbreviata $\lambda f.\ \lambda x.f^n\, x$

Wittgenstein e i numeri di Church.

La definizione dei numeri Church non è altro che la versione formale di quanto scriveva Wittgenstein nel 1922 nel suo Tractatus logico-philosophicus (proposizione 6.021):

"un numero naturale è l'esponente di un'operazione"
intendendo con ciò che un numero è l'applicazione ripetuta di una stessa operazione a un oggetto di partenza;

La notazione unaria ss …s0 può essere vista come schema prototipale di tale ripetizione, dove:

- l'oggetto di partenza si chiama 0;
- l'operazione è quella di successore.

Aritmetica

Sui numeri di Church si possono agevolmente definire le operazioni aritmetiche, per esempio il + è la funzione

$$\lambda m.\ \lambda n.\ \lambda f.\ \lambda x.\ (mf)(nfx)$$

perché applicando n volte la funzione f a x, e poi applicando al risultato la funzione f stessa m volte , si applica m+n volte la funzione f a x.

Esempio:

$$2 + 3 = (\lambda m.\ \lambda n.\ \lambda f.\ \lambda x.\ (mf)(nfx))2\ 3 \rightarrow$$

$$\lambda f.\ \lambda x.\ (2f)(3fx) \rightarrow$$

$$\lambda f.\ \lambda x.\ (2f)(f(f(fx))) \rightarrow$$

$$\lambda f.\ \lambda x.\ (f(f(f(f(fx))))) = 5$$

Non è peggio dell'analogo calcolo fatto con i numeri nella notazione unaria di Peano.

Un risultato fondamentale.

Nel 1935-36 Church e Kleene dimostrarono che le funzioni sui numeri naturali definibili nel λ-calcolo sono tutte e sole le funzioni definibili nel sistema delle funzioni generali ricorsive di Gödel-Herbrand o in quello equivalente delle funzioni μ-ricorsive di Kleene.

Quindi il λ-calcolo, il sistema Gödel-Herbrand e quello di Kleene definiscono la stessa classe di funzioni.

Nota.

Poiché qualunque tipo di dato è codificabile, come oggi ben sappiamo, in un numero naturale, i tre tipi di definizione individuano, in tre modi completamente diversi, una stessa ampia classe di problemi, non direttamente numerici, risolubili mediante procedimenti meccanici di calcolo.

Trovata la nozione giusta?

Tale convergenza di risultati portava naturalmente a chiedersi se tale classe di funzioni non fosse proprio la classe di tutte e sole le funzioni "intuitivamente" considerate calcolabili. Cioè se non si fosse trovata la nozione matematica precisa corrispondente alla nozione intuitiva di "calcolabile", così come nell'800 si era trovata la nozione matematica precisa corrispondente alla nozione intuitiva di "*funzione il cui grafico è tracciabile senza sollevare la matita dal foglio*" (funzione continua).

Lo scetticismo di Gödel

Il logico più autorevole del tempo, Gödel, creatore, come abbiamo visto, di una delle suddette forme di definizione, non era affatto convinto che esse corrispondessero esattamente alla nozione intuitiva di "calcolabilità": pensava che non si potesse escludere l'esistenza di altri modi di "calcolare" che non erano catturati dai sistemi esistenti (funzioni ricorsive, λ-calcolo , ecc.).

Anzi, pensava che del concetto intuitivo di "calcolabilità" non si potesse dare una definizione precisa e rigorosa, ma servisse solo come "principio euristico".

Church e l'Entscheidungsproblem.

Church dimostrò che, secondo la sua nozione di calcolabilità, l'Entscheidungsproblem per una teoria assiomatica "abbastanza forte" (come quelle per le quali Gödel aveva dimostrato i suoi teoremi di incompletezza) è insolubile: cioè che non può esistere un algoritmo (esprimibile nel λ-calcolo) il quale, dato un qualunque enunciato ad es. nell'aritmetica di Peano (del primo ordine), sia sempre in grado di stabilire se tale enunciato è un teorema della teoria oppure no.

Alan Turing, 1912-1954

On Computable Numbers, with an Application
to the Entscheidungsproblem.
1936

L'analisi di Turing

Turing dà della nozione di calcolabilità una definizione radicalmente diversa da quelle di Gödel, Kleene e Church, partendo da un'accurata analisi delle azioni elementari di un uomo nell'atto di calcolare (un numero reale),

``a man in the process of computing'':

Computing is normally done by writing certain symbols on paper. We may suppose this paper is divided into squares like a child's arithmetic book. In elementary arithmetic the two-dimensional character of the paper is sometimes used. But such a use is always avoidable, and I think that it will be agreed that the two-dimensional character of paper is no essential of computation. I assume then that the computation is carried out on one-dimensional paper, i.e. on a tape divided into squares. I shall also suppose that the number of symbols which may be printed is finite. [...]

Il computer di Turing

Nel 1936, quando Turing scriveva il suo articolo, i calcolatori elettronici non esistevano ancora, e la parola "computer" indicava un essere umano (di solito donna) addetto ai calcoli (ad es. delle tavole di tiro dei cannoni per l'esercito).

The behaviour of the computer at any moment is determined by the symbols which he is observing, and his ``state of mind'' at that moment.

We may suppose that there is a bound B to the number of symbols or squares which the computer can observe at one moment. If he wishes to observe more, he must use successive observations.

We will also suppose that the number of states of mind which need be taken into account is finite. [...] If we admitted an infinity of states of mind, some of them will be ``arbitrarily close'' and will be confused.

L'analisi di Turing (continuazione).

L'azione del calcolatore (umano) è dunque determinata dalla coppia ⟨stato della mente, simbolo letto sul nastro⟩, e poiché sia l'insieme dei simboli che l'insieme degli stati sono finiti, anche l'insieme delle possibili azioni specificate è finito.

La quantità di carta a disposizione del calcolatore umano è invece illimitata, cioè il nastro quadrettato è potenzialmente infinito, benché ad ogni dato istante solo un numero finito di quadretti sia non bianco.

Turing conclude la sua analisi osservando che un'azione elementare può essere la scrittura di un simbolo nel quadratino esaminato (se questo è bianco), oppure la cancellazione del simbolo contenuto nel quadratino, oppure lo spostamento dello ``sguardo'' su un quadratino adiacente; ognuna di tali azioni può essere accompagnata da un passaggio della mente da uno stato a un altro.

Macchine di Turing

Il calcolatore umano (computer) può pertanto essere simulato da una macchina:

"we may now construct a machine
to do the work of this computer"

Una *Macchina di Turing (abbrev. MdT)* è dunque definita da un alfabeto finito di simboli $s_1, s_2, \ldots, s_n$ (fra i quali uno indicante la casella bianca, ad es. "_"), da un insieme finito di stati $q_1, q_2, \ldots, q_m$, fra i quali ve ne è uno particolare designato come *stato iniziale*, e da una tabella di transizione le cui righe definiscono il comportamento della macchina nei vari casi possibili. Ad es.:

stato	simbolo letto	azione	nuovo stato
q_1	s_1	scrivi s_3	q_2
q_1	s_2	vai a destra	q_1
...	...	...	...

Macchine di Turing

Una macchina può quindi essere pensata come costituita da una testina mobile di lettura/scrittura e di un nastro infinito (o, alternativamente, come una testina fissa su un nastro mobile).

La testina è ad ogni istante posizionata sopra una casella del nastro e possiede uno stato interno q_i.

In una formulazione equivalente più conveniente, le colonne della tabella sono cinque invece di quattro, ad esempio:

stato	simbolo letto	scrittura	spostamento	nuovo stato
q_1	s_1	s_3	→	q_2
q_1	s_2	s_2	←	q_3
q_2	s_1	s_2	•	q_1
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...

Macchine di Turing

Nella colonna dello spostamento ci può essere:

←: spostamento della testina di una casella a sinistra;

→: spostamento della testina di una casella a destra;

•: la testina non si sposta (rimane Immobile).

A seconda dello stato in cui si trova e del simbolo letto, cioè dei contenuti delle prime due colonne, la macchina:

- sovrascrive la casella con il simbolo indicato in terza colonna;
- effettua lo spostamento indicato nella quarta colonna;
- si pone nello stato indicato dalla quinta colonna.

In modo più astratto.

Una "tabella" è un insieme di quintuple della forma:

$$q_i, s_k \Rightarrow s_r, \mathcal{M}, q_j$$

Se la testina, trovandosi la macchina nello stato interno q_i, legge sulla casella sottostante il simbolo s_k, allora:

- sovrascrive nella casella il simbolo s_k con il simbolo s_r;
- si muove di $\mathcal{M}$ (o, in una visione alternativa, muove il nastro di $\mathcal{M}$), dove $\mathcal{M}$ può essere $\leftarrow$, $\rightarrow$, o $\bullet$;
- passa dallo stato q_i allo stato q_r.

La macchina si ferma quando raggiunge una coppia

$$\langle \text{stato} , \text{simbolo} \rangle$$

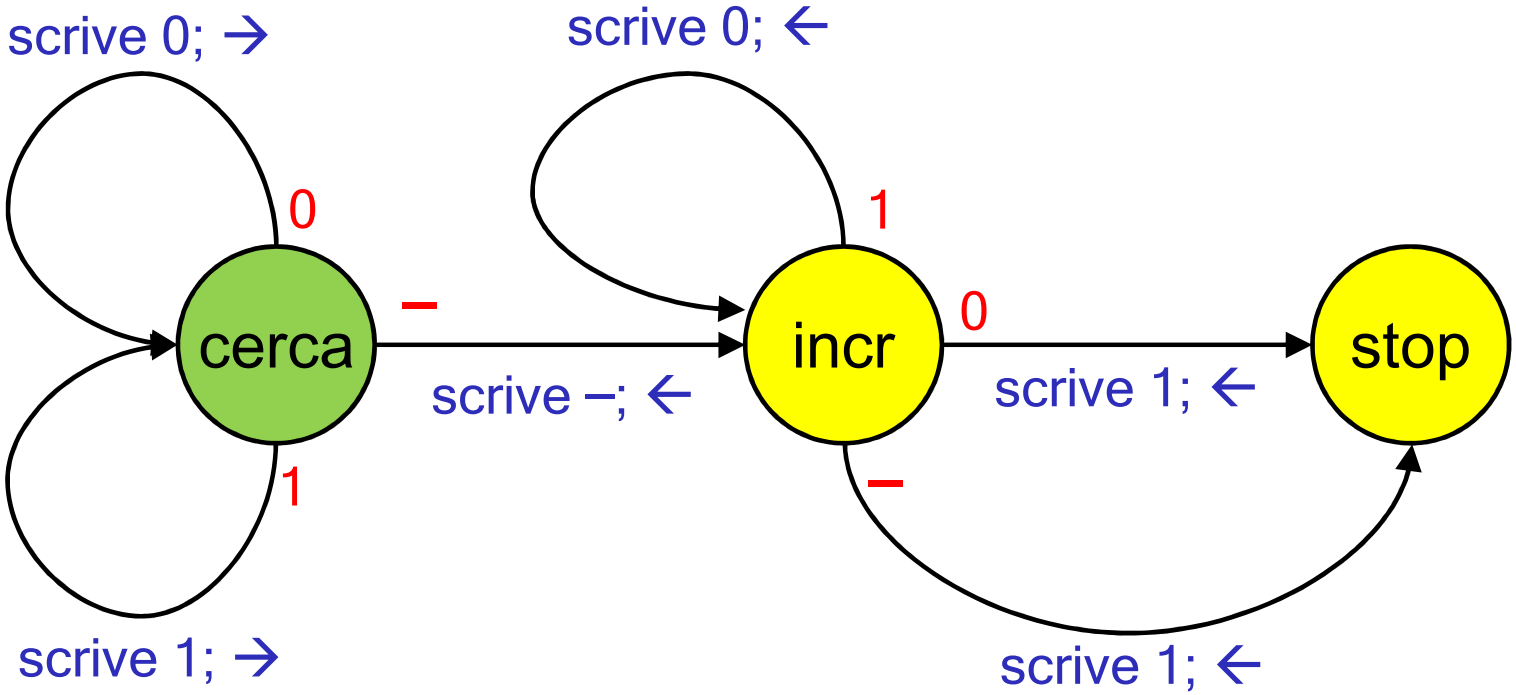
per cui non è specificata alcuna azione, oppure quando raggiunge uno stato designato come finale.

Esempio: una macchina che aggiunge 1 a un numero in notazione binaria.

La macchina lavora con un alfabeto di tre simboli: 0, 1, bianco, e ha tre stati interni: cercaBitDestro, incrementa, e stop.. La sua tabella è la seguente:

stato	simbolo letto	scrive	si sposta	nuovo stato
cercaBitDestro	bianco	bianco	←	incrementa
cercaBitDestro	0	0	→	cercaBitDestro
cercaBitDestro	1	1	→	cercaBitDestro
incrementa	bianco	1	←	stop
incrementa	0	1	←	stop
incrementa	1	0	←	incrementa

Una rappresentazione grafica.



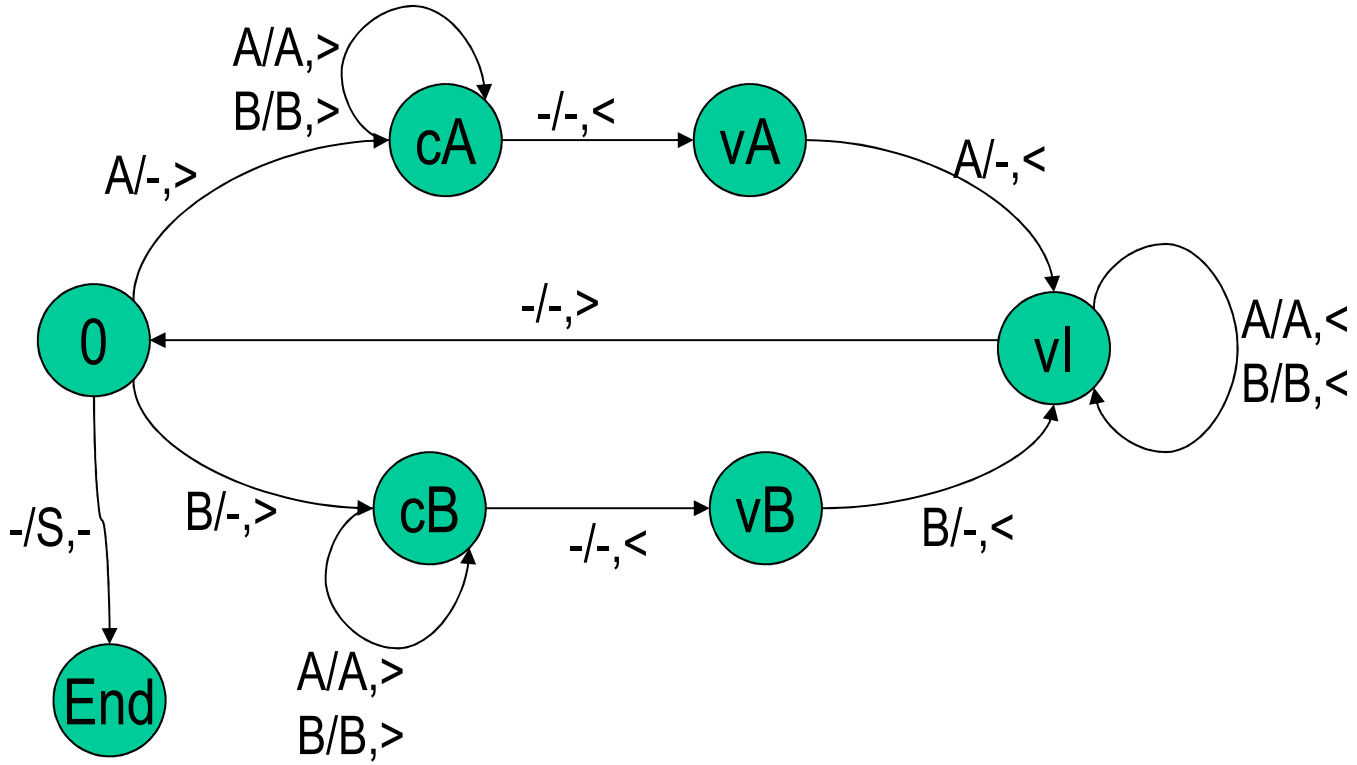
dove: lo stato iniziale è "cerca",

– = bianco, ← = "si sposta a sinistra", → = "si sposta a destra"

Varietà delle macchine.

- Macchine di Turing diverse, cioè con insiemi di regole diversi, effettuano processi di calcolo diversi.
- Ciascuna particolare macchina realizza quindi un particolare algoritmo, e viceversa un particolare algoritmo può essere realizzato da (almeno) una macchina.

Altro esempio: una macchina trova-palindrome.



Equivalenza fra la definizione di Turing e le altre.

Nel 1937 Turing dimostrò che la classe delle funzioni calcolabili dalle sue macchine coincideva anch'essa con la classe delle funzioni calcolabili secondo le definizioni precedenti (Herbrand-Gödel, Kleene, Church).

La definizione di Turing era però molto più persuasiva delle altre, e convinse anche Gödel che la nozione rigorosa di calcolabilità individuata da tutte queste definizioni equivalenti corrispondeva davvero alla nozione intuitiva:

"[...]The most satisfactory way, in my opinion, is that of reducing the concept of finite procedure to that of a machine with a finite number of parts, as has been done by the British mathematician Turing."

"I was completely convinced only by Turing's paper".

Turing come Michelangelo?

Un informatico americano, Robert Soare, in vena di paragoni straordinari, ha paragonato la differenza fra le definizioni di computabilità di Church e di Turing a quella fra i David di Donatello e di Michelangelo:

"Michelangelo and Turing both completely transcended conventional approaches. First ,they both created something completely new from their own visions, something which went far beyond the achievements of their contemporaries.

Second, both emphasized the human form.

Michelangelo brought out the human form in his statues and in the Sistine ceiling with magnificent human figures often shown in contraposto. Turing left behind the formal systems of lambda-definable or recursive functions. Turing searched into how a human being actually computes. ..."

La macchina di Turing universale.

Turing dimostrò che si può definire una MdT universale, cioè una MdT la quale, presi come input sul nastro:

- una descrizione, in un opportuno linguaggio di codifica, di una qualunque MdT,
- e un input per tale macchina,

è in grado di simulare la computazione di quella macchina a partire da quell'input.

Una MdT universale è perciò in grado di eseguire qualunque possibile algoritmo, ed è dunque il modello matematico del computer:

- il nastro della MdT universale rappresenta sia i dispositivi di input-output che la memoria (potenzialmente infinita);
- il linguaggio in cui devono essere descritte le MdT particolari è il linguaggio-macchina della MdT universale.

Macchina di Turing e calcolatori elettronici.

- La macchina di Turing universale non fu tuttavia mai intesa come descrizione della struttura concreta di una macchina reale.
- La definizione della struttura concreta del computer fu opera di molti scienziati e ingegneri, fra i quali John von Neumann, genio universale della matematica, della fisica e della nascente informatica: l'architettura costituita da:
  - unità logico aritmetica (ALU), che effettua i calcoli;
  - unità di controllo, che decide quali calcoli effettuare;
  - memoria di lavoro;
  - dispositivi di input/output

  è ancora oggi nota come "architettura di von Neumann".
- Ogni computer è equivalente a una MdT universale, cioè è in grado di eseguire qualunque possibile algoritmo.

John von Neumann (nato Neumann János)

Budapest, 1903 – Washington, 1957

Turing e il problema della fermata.

Turing dimostrò che esistono problemi algoritmici ben definiti per i quali non può esistere alcun algoritmo risolvente. Turing formulò i suoi teoremi in termini della sua macchina, ma oggi noi possiamo formularlo parlano semplicemente di computer e di programmi.

Il teorema fondamentale di Turing si può esprimere allora così:

Non può esistere un programma il quale:

- presi in input un qualunque programma P e un qualunque input Inp per quel programma,
- stabilisce se l'esecuzione di P con l'input Inp termina oppure no.

Il problema della fermata

Nota Bene. Per stabilire che un dato programma con un dato input termina, non ci si può limitare a lanciarne l'esecuzione:

se il programma termina lo scopriamo,
ma se non termina non lo scopriremo mai.

Per stabilire che non termina bisogna dare, esaminando il programma, una dimostrazione del fatto che non termina!

Esempi.

Quando un programma "si congela" (cioè "non risponde"), che cosa facciamo? Continuiamo ad aspettare perché magari fra un po' risponde, o facciamo un riavvio?

Quando aspettiamo l'autobus alla fermata e non arriva, continuiamo ad aspettarlo perché magari arriverà in ritardo, oppure andiamo a piedi perché forse l'autobus è stato deviato, oppure c'è uno sciopero, ecc.?

Turing e l'Entcheidungsproblem.

Turing dimostrò che se l'Entscheidungsproblem fosse risolubile, allora sarebbe risolubile anche il problema della fermata.
Ma poiché il problema della fermata non è risolubile, allora non è risolubile nemmeno l'Entscheidungsproblem.

La dimostrazione si basa sul seguente ragionamento.

Si può costruire una teoria assiomatica dei programmi, in cui si possono scrivere degli enunciati equivalenti ad affermazioni del tipo "il programma P termina".

Per la definizione, se l'Entscheidungsproblem fosse risolubile, esiterebbe un algoritmo il quale, data una qualunque teoria assiomatica (del prim'ordine) e dato un qualunque enunciato nel linguaggio di quella teoria, stabilisce se tale enunciato è o non è un teorema della teoria.

Ma allora, applicando tale algoritmo alla suddetta teoria dei programmi, si risolverebbe il problema della fermata!

Turing e i teoremi di incompletezza di Gödel.

Il primo teorema di incompletezza diventa quindi, almeno in una sua forma particolare riferentesi alla teoria dei programmi, una conseguenza immediata dei risultati precedenti.

Se infatti fosse possibile costruire una teoria assiomatica dei programmi "sintatticamente completa", cioè tale che in essa per ogni enunciato E sia dimostrabile o E o la sua negazione, allora per sapere se un programma P con input I termina, basterebbe (!) generare una dopo l'altra tutte le possibili dimostrazioni (in ordine crescente di lunghezza), finché si trova o la dimostrazione dell'enunciato "P con input I termina", oppure quella dell'enunciato "P con input I non termina" !

Conclusione: l'importanza dei risultati negativi.

- Finché si trattava di mostrare che un dato problema poteva essere risolto da un certo algoritmo, bastava far vedere l'algoritmo; oggi diremmo, basta scrivere il programma in un qualche linguaggio di programmazione.
- Ma per stabilire che non può esistere alcun algoritmo che risolve un dato problema, bisognava preventivamente stabilire che cosa è un procedimento di calcolo in generale.
- Negli anni '30 del '900, come abbiamo visto, proprio le dimostrazioni dell'irresolubilità del problema della decisione, che segnavano il fallimento del "programma di Hilbert", portarono alla definizione precisa del concetto di calcolabilità, e quindi alla nascita dei fondamenti dell'informatica.