Fondamenti di Informatica

Computabilità e Macchine di Turing

Prof. Franco Zambonelli
Gennaio 2011

Letture Consigliate:

Roger Penrose, La Mente Nuova dell'Imperatore, Sansoni Editrice.

Martin Davis, Il Calcolatore Universale, Adelphi.

Yuri Castelfranchi, Macchine Come Noi.

Il Sogno di Hilbert

Alla fine dell'800, grazie ai progressi nel campo della logica (Algebre di Boole e Frege), i matematici cominciarono a pensare che tutta la matematica potesse essere in qualche modo "automatizzata". O meglio:

- ogni sistema matematico (per esempio quello euclideo) si basa su un insieme finito di assiomi, le "verità" di base di quel sistema matematico
- ogni nuova verità si deriva da quelle esistenti tramite un insieme limitato di regole logiche
- ogni teorema è definibile in modo preciso attraverso la composizione opportuna di regole a partire dagli assiomi o da verità già dimostrate dagli assiomi

Di conseguenza, componendo le regole in tutti i modi possibili a partire dagli assiomi è possibile elencare e dimostrare tutti i teoremi possibili di un qualsiasi sistema matematico formale (la sfida di Hilbert, 1900).

Quindi, avendo a disposizione una "macchina" in grado di gestire le verità del mondo e di operare automaticamente su di esso attraverso degli operatori, tale macchina sarà in grado, se opportunamente costruita in modo da comporre opportunamente gli operatori, di dimostrare un qualsiasi teorema tra tutti quelli elencabili…

Il Sogno svanisce: Cantor

Gli Infiniti di Cantor

Cantor identificò per primo (1874) la presenza di tipi diversi di "infiniti":

- infiniti numerabili, cioè che possono essere messi in associazione univoca con l'insieme dei numeri naturali
- infiniti non numerabili, che non possono essere messi in associazione univoca con i numeri naturali, e che sono in qualche modo infiniti più grandi.

I numeri razionali sono numerabili:

	1	2	3	4	5
1	1 fratto 1	1 fratto 2	1 fratto 3	1 fratto 4	etc
2	2 fratto 1	2 fratto 2	2 fratto 5	etc	
3	3 fratto 1	3 fratto 2	etc		
4	4 fratto 1	etc			
5	etc				
6					

I numeri reali no.

Dimostrazione: il procedimento di diagonalizzazione!!!
Per numerare i numeri razionali, possiamo mettere in fila tutte i possibili numeri, e poi comporre un numero prendendo la prima cifra decimale diversa dal primo numero, la secondo diversa dal secondo numero, etc all'infinito. Questo numero sarà diverso da tutti i razionali, e non sarà numerabile.

Di conseguenza: chi ci dice che l'insieme dei teoremi matematici definiscono un insieme numerabile, e quindi "elencabile"??? Il dubbio c'è, e Hilbert lo sapeva…

Il Sogno svanisce: Russell

Bertand Russell applica agli insiemi il procedimento di diagonalizzazione, e arriva a ragionare sulla esistenza dell' "insieme di tutti gli insiemi", e si chiede: tale insieme appartiene all'insieme o no?. Da qui, arriva al seguente paradosso (1902) che manda in crisi Frege:

"l'insieme di tutti gli insiemi che non contengono sé stessi"
Contiene sé stesso oppure no?

Da cui derivano una serie di "versioni della strada": il barbiere che rade solo coloro che non radono sé stessi; "non vorrei fare parte di un club che mi accetta tra i suoi membri" (Groucho Marx).

Tale paradosso è "devastante" perché in un qualche modo tutta la logica era basata, più o meno direttamente, sui concetti di insieme e sulla verità logica come verità di appartenenza a un insieme…Qui Russell prova che ci sono verità sugli insiemi che non possono essere dimostrate tramite la teoria degli insiemi!!!!

E' un caso patologico della teoria degli insiemi?? No, in realtà…

Il Teorema di Godel

Kurt Godel, dimostra (1931) attraverso un sofisticato teorema matematico che:

"in un qualsiasi sistema matematico formale, esistono proposizioni **indecidibili** all'interno del sistema stesso"

In pratica, dimostra che il paradosso di Russell non era un caso, ma una proprietà generale di tutti i sistemi matematici basati su regole e assiomi. E quindi, afferma che non è possibile dimostrare in modo automatico tutti i teoremi del mondo….

E' il teorema fondamentale della matematica e della informatica moderna.

Ma in che modo questo influisce sull'informatica??
Ce lo dice Alan Turing…

La Macchina di Turing

Alan Turing si pose per primo il problema di costruire una macchina "concettuale" in grado di "dimostrare" teoremi, sulla base delle teorie di Hilbert.

La macchina completa consiste in una astrazione concettuale immaginata a partire da un'uomo che fa calcoli matematici su un foglio di carta:

- Di un nastro potenzialmente infinito, in grado di contenere "simboli" su caselle in fila sul nastro. → astrazione del foglio
- Un meccanismo per fare scorrere il nastro (o per muoversi sul nastro) → astrazione della penna che si muove sul foglio
- Un meccanismo per leggere e scrivere sul nastro. Nella versione originale, la macchina poteva solo marcare (p.e., bucare) il nastro e riconoscere i buchi. In generale, la cosa è indifferente dal punto di vista matematico, possiamo pensare di avere simboli binari o simboli qualsiasi sul nastro.
- Di un'insieme possibili di stati "interni" alla macchina. La macchina passa da uno stato all'altro a seconda delle azioni che ha compiuto, e il suo comportamento può essere diverso a seconda del suo stato interno (così come noi ci reagiamo diversamente a seconda dell'umore). Lo stato della macchina in qualche modo traccia le azioni passate della macchina. → astrazione del "tenere a mente le cose" mentre l'uomo fa i calcoli

Quindi, la macchina, ha integrate una serie di regole che stabiliscono

- Dato lo stato interno della macchina, dato il simbolo presente sulla posizione corrente del nastro
- Quali azioni deve compiere la macchina (spostarsi sul nastro e/o scrivere un simbolo sulla posizione corrente del nastro),
- Quale sarà lo stato successivo della macchina (transizione di stato)

Una regola "speciale" stabilisce lo "stop" della macchina (fine del compito)

Ora, se i simboli sul nastro rappresentano in qualche modo "dati" e "fatti" del mondo, e la macchina è in grado di leggerli, le regole della macchina di Turing sono in grado di analizzare una serie di fatti e su questa base produrre nuovi fatti → cioè è in grado di dimostrare teoremi unendo analisi aritmetico logica.

Esericizio: pensare a una trasformazione per gradi della macchina di Turing che la faccia diventare la moderna architettura di Von Neumann.

La Macchina di Turing (esempi)

Assumiamo una macchina:

- le caselle del nastro possono essere vuote (o zero, è lo stesso) o avere un 1

Possiamo costruire una macchina che ci dice se sul nastro ci sono almeno due 11 consecutivi?
Codifichiamo le regole in questo modo: ***Situazione → Azioni***

O, più in dettaglio: ***Stato Interno, Simbolo Nastro → Prossimo Stato, Azione***

Dove:
Stato Interno è un numero che identifica lo stato interno della macchina,
Simbolo Nastro identifica il simbolo presente sulla posizione corrente del nastro.
Prossimo Stato indica lo stato a cui passerà la macchina nello "step" seguente
Azione indica cosa deve fare la macchina. Essa può: spostarsi a destra (lo indichiamo con con >) o a sinistra (lo indichiamo con <) sul nastro, oppure può cambiare il simbolo sul nastro.

In pratica: le regole dicono cosa deve fare la macchina quando si trova in un certo stato e legge un certo simbolo.

Una regola che fa passare la macchina in uno stato per cui non ci sono regole definite provoca lo "stop" della macchina. Chiaramente bisogna che la macchina si arresti perché si possa leggere il risultato stabile sul nastro, altrimenti è tutto inutile.

Ecco allora le regole di una macchina per riconoscere la presenza di due 1 di seguito:
0 0 0 > (non trovo 1, mi sposto a sinistra)
0 1 1 > (sono in stato 0 e trovo un 1, mi porto in stato 1 e mi sposto a sinistra)
1 0 0 > (sono in stato 1 e trovo uno 0, torno in stato 0 e mi sposto)
1 1 2 > (sono in stato 1 e trovo un 1, mi porto in stato 2 e mi sposto)

Oppure le regole per riconoscere una configurazione "101"
0 0 0 >
0 1 1 >
1 0 2 >
1 1 0 >
2 1 3 >
2 0 0 >

Esercizio: scrivere le regole per trasformare una configurazione "10101" nella configurazione "10001". Sperimentare sulla applet.

La Macchina di Turing Universale

Chiaramente una macchina di turing con un certo insieme di regole è in grado di risolvere un solo specifico problema.

Il vero contributo di turing non è stato inventare il concetto di macchina di turing, ma scoprire che

- poteva esistere una macchina di turing universale, cioè una macchina in grado di risolvere qualsiasi problema risolvibile da qualsiasi altra macchine di turing
- Scoprire che esistono problemi che le macchine di Turino non possono risolvere
- Evidenziare che se una macchina di Turing non può risolvere un certo problema, nessuna altra macchina di nessun tipo potrà risolverlo.

Consideriamo che:
Le possibili regole per le macchine di Turing sono in numero finito ma numerabile. (ricordiamo la numerabilità dei numeri razionali, come insieme NxN, e possiamo facilmente estendere a considerare le regole delle macchine di Turing come un insieme numerabile 2N*2N)

Possiamo quindi numerare tutte le macchine di Turino che possiamo concepire:

T_1, T_2, T_3, T_n

Consideriamo quindi m come la configurazione di simboli sul nastro.

Ebbene, la n-esima macchina di Turing si potrà quindi considerare come una funzione:

$T_n(m)=q_n$

Cioè come una funzione ***Tn*** che applicata su ***m*** fornisce un output ***q***.

Ma allora è possibile identificare una macchina ***k-esima*** tale che:

$$T_k(n,m) = q_n \quad \textit{per ogni n, e per ogni m}$$

Il Teorema dell'Arresto

Esistono delle cose che una macchina di Turing non può fare? Dei problemi che non può risolvere?
Ebbene: non esiste una macchina di Turing che è in grado di determinare se un'altra generica macchina di Turing si fermerà mai o no.

Ragionando generalmente: una macchina per determinare se un'altra macchina si ferma, deve in qualche modo "emularne" il comportamento, ma se ne emula il comportamento allora se la macchina emulata non si dovesse arrestare neanche la macchina emulatrice si potrebbe arrestare…

Formalmente:
consideriamo che questa mitica macchina sia Tx, ed essa sia in grado sempre di dirci se la macchina qualsiasi Tn, con input qualsiasi m, si ferma o no.

$T_x(n, m) = 1$ *Se* $T_n(m)$ *si ferma*
$T_x(n, m) = 0$ *Se* $T_n(m)$ *non si ferma*

Chiaramente la macchina deve funzionare anche nella ipotesi restrittiva che n=m, per ogni m,n (attenzione; notiamo che porre n=m significa fare un procedimento di diagonalizzazione!!!)

Quindi

$T_x(n, n) = 1$ *Se* $T_n(n)$ *si ferma*
$T_x(n, n) = 0$ *Se* $T_n(n)$ *non si ferma*

Ma con questa ipotesi restrittiva, T_x diventa funzione di una singola variabile, per cui:

$T_x(n) = 0$ *Se* $T_n(n)$ *non si ferma*

Ma questo deve chiaramente valere anche per n=x

Per cui $T_x(x) = 0$ *se* $T_x(x)$ *non si ferma*. Il che è un assurdo.

NOTIAMO CHE: Siamo giunti a un assurdo equivalente al paradosso di Russel, ed equivalente al concetto di indecidibilità di Godel.

IMPLICAZIONI: esistono problemi non computabili, che un calcolatore non è in grado di risolvere. Non esistono programmi che possano controllare in modo automatico la correttezza di altri programmi.