# PENSIERO COMPUTAZIONALE
# e la Macchina di Turing per il problem solving

Paolo Forte

## DA DOVE PARTIAMO?

## A COSA SERVE?

PROVEREMO A RISPONDERE
alle seguenti domande:

1. Esistono problemi matematici che non ammettono neppure in linea di principio di essere risolti mediante alcun algoritmo? (scopriremo che è strettamente legato allo studio delle proprietà dei sistemi formali della logica matematica).
2. La teoria della computabilità (come parte della Logica), in cui vengono indagati concetti quali quello di algoritmo e di funzione computabile in modo algoritmico, svolge il ruolo di «teoria dei fondamenti» dell'informatica?
3. Il logico matematico Turing con la sua Macchina risponde alle domande 1. e 2.?
4. Qual è il contributo di Turing all' Intelligenza artificiale?

## Cominciamo con la natura del pensiero computazionale

Il pensiero computazionale fornisce gli per studiare l'informatica. La sua applicazione, però, va ben al di là dell'informatica: è il processo del riconoscimento degli aspetti della computazione nel mondo che ci circonda, e nell'applicazione di strumenti e tecniche informatiche per capire e ragionare su sistemi e processi naturali, sociali e artificiali.

La prof.ssa Jeannette Wing (2006) ha definito per prima il pensiero computazionale come: «(...) I processi mentali coinvolti nel formulare problemi e le loro soluzioni in modo che le soluzioni possano essere rappresentate in una forma che può essere efficacemente eseguita da un agente di elaborazione dell'informazione. La soluzione può essere eseguita da un essere umano o da una macchina, o, più in generale, da combinazioni di uomini e macchine."

# Concetti del pensiero computazionale

Il pensiero computazionale è un processo cognitivo che coinvolge il ragionamento logico. Il pensiero computazionale comprende la capacità di pensare in termini di:

- algoritmi;
- scomposizione;
- generalizzazioni, individuando e facendo uso di schemi ricorrenti;
- astrazioni, scegliendo rappresentazioni appropriate;

# Ragionamento logico

Le teorie della complessità (valutazione della difficoltà di qualsiasi problema) e della calcolabilità (valutazione della possibilità di calcolare un problema qualsiasi) sono basate sulla Logica ed essendo in bilico fra matematica, informatica, ingegneria  e filosofia ...non è facile.

La genesi risale ai tempi di Aristotele con il tentativo di ridurre il ragionamento logico ad una serie di regole formali.

Calcolo e ragionamento logico sono facce della stessa medaglia, intuizione che dobbiamo ricordare non solo nel programmare i calcolatori, ma anche per progettarli e costruirli.

Durante questo processo, si usa l'astrazione, la valutazione e il pensiero algoritmico

# Astrazione

L'abilità nell’astrazione sta nello scegliere il dettaglio da nascondere in modo che il problema diventi più facile, senza perdere niente di ciò che è importante.

Un aspetto fondamentale di questo processo è la scelta di una buona rappresentazione del sistema.

Rappresentazioni diverse rendono più semplice operare su cose diverse.

Per esempio, un programma per computer che gioca a scacchi è un'astrazione. Si tratta di un insieme finito e preciso di regole da seguire ogni volta che è il turno del computer di muovere.

È lontano dai processi mentali analogici, emozionali, parziali e distratti tipici di un giocatore umano di scacchi.

È un'astrazione perché i dettagli inutili di tali processi sono rimossi.

# Pensiero algoritmico
# (Algoritmo formale)

Il pensiero algoritmico è la capacità di pensare in termini di sequenze
Algoritmo ->una successione finita di passi(steps) che definiscono le operazioni da eseguire sui dati (input)per ottenere i risultati(output).

- I passi che costituiscono lo schema devono essere “elementari”, ovvero non ulteriormente scomponibili (atomicità);
- I passi devono essere interpretabili in modo diretto e univoco dall’esecutore, sia esso umano o artificiale (non ambiguità);
- L’algoritmo deve essere finito, ossia composto da un numero definito di passi legati ad una quantità definita di dati in ingresso (finitezza);
- L’esecuzione deve avvenire entro un tempo finito (terminazione);
- L’esecuzione dello schema algoritmico deve condurre ad un unico risultato (DETERMINISMO).

# Macchina di Turing  per formulare un algoritmo

La formulazione di un algoritmo dipende dal modello di calcolo utilizzato: un “programma” per un modello matematico astratto, come una Macchina di Turing oppure il linguaggio C per un PC (2 differenti modelli di calcolo).

Qualsiasi modello si scelga, gli algoritmi devono esservi descritti, ossia rappresentati da sequenze finite di caratteri di un alfabeto finito .

⇒ gli algoritmi sono possibilmente infiniti, ma numerabili cioè possono essere ‘elencati’ (messi in corrispondenza biunivoca con l’insieme dei numeri naturali).

I problemi computazionali (funzioni matematiche che associano ad ogni insieme di dati il corrispondente risultato) non sono numerabili

# Logica

«La logica, che può dare soltanto la certezza, è lo strumento della dimostrazione; l'intuizione, lo strumento dell'invenzione.» (Henri Poincaré - Il valore della scienza)

Il primo importante studio sulla logica lo si deve ad Aristotele (384-322 a.C) e celebri sono rimaste le sue forme di sillogismo(deduzione -logica del primo ordine ). Ma «si può avere un ragionamento logicamente corretto e rigoroso, i cui contenuti siano falsi» !

Fu soprattutto Gottfried Leibniz (1646-1716) a rinnovare l'interesse per la formalizzazione e meccanizzazione del ragionamento in termini logici attraverso gli strumenti formali della matematica.

Nell'Ottocento, le due figure più rappresentative che per prime seppero realizzare concretamente il sogno della “meccanizzazione” della logica furono, prima, l‟inglese George Boole e, poi, il tedesco Gottlob Frege.

# Logica del primo ordine

Linguaggio formale che serve per gestire meccanicamente enunciati e ragionamenti che coinvolgono i connettivi logici, le relazioni e i quantificatori "per ogni ..." (∀) ed "esiste..." (∃). L'espressione "del primo ordine" indica che c'è un insieme di riferimento e i quantificatori possano riguardare solo gli elementi di tale insieme e non i sottoinsiemi; ad esempio si può dire "per tutti gli x elementi dell'insieme vale P(x)" ma non si può dire "per tutti i sottoinsiemi di A vale P(A)" (le teorie in cui ci sono quantificatori che spaziano sui sottoinsiemi dell'insieme di riferimento sono dette invece del secondo ordine).

# ALGEBRA LOGICA di BOOLE(1840)

Connettivi logici :AND ,NOT, OR per proposizioni con valori di verità (VERO = 1,FALSO=0) che ottengono valori di verità

CORRISPONDENZA tra and(1,0) e prodotto (1x 0)=0

tra or (1,0) e somma (1+0)=1

tra not(1) e –( 1)=0

Boole analizzò per la prima volta un linguaggio che ammetteva tra le possibili interpretazioni il calcolo proposizionale, che rappresenta oggi un'importante parte della logica matematica.

# I problemi di Hilbert

**Hilbert all'inizio del 20-esimo secolo (1900)aveva steso una lista di ventitré "problemi aperti"** cui si sarebbe dovuto dare risposta per sanare la "crisi dei fondamenti" che i pensatori come lui attribuivano alla disciplina, e ancora oggi i giovani matematici che ambiscono a vincere la **Medaglia Fields** (il massimo riconoscimento nel campo) cercano di risolvere i quesiti rimasti irrisolti. **Il secondo dei Ventitré Problemi, che consisteva nel provare la *coerenza* dell'artimetica** (ossia che non è possibile dimostrare al suo interno un'affermazione e anche il suo contrario) **fu ciò a cui scelse di lavorare il giovanissimo Gödel, nel 1929, a soli ventitré anni**.

# Dal programma di Hilbert

Si tratta di dimostrare *metamatematicamente* (cioè con dimostrazioni matematiche che hanno come oggetto la matematica stessa) la coerenza della matematica con metodi *finitari* (cioè che non chiamino in campo il concetto di infinito), ossia, semplificando, che la matematica è in grado di parlare di se stessa e che in essa non esistono *ignorabimus*, cioè proposizioni di cui non si può stabilire la verità o la falsità, chiamate tecnicamente *indecidibili*.

**Gödel :** Un sistema matematico non può essere sia coerente(“non contraddittorio”) che completo (in grado di “parlare di tutto”)

Dimostra che esistono proposizioni per le quali non è possibile dimostrare né la verità né la falsità, ma che qualunque altro sistema formale sarebbe risultato incompleto a causa di un’incompletezza di fondo caratterizzante qualunque sistema formale logico (sufficientemente espressivo).

Gödel riuscì a mostrare anche che nel sistema dell’aritmetica non è possibile formulare un enunciato logico che esprima la correttezza del sistema, conclusione che costituisce il secondo teorema di Gödel:

Nessun sistema coerente può essere utilizzato per dimostrare la sua stessa coerenza.

# Gödel

Dalla matematica greca :
Il paradosso del mentitore (Epimenide):
“Questa frase è falsa”.
Gödel modificò questo paradosso nella forma
“Questo teorema non è dimostrabile” e mostrò che era possibile codificare opportunamente questa espressione in una formula di un sistema logico formale (purché sufficientemente espressivo) dando luogo all‟incompletezza del sistema stesso.

# Da Godel alla Calcolabilita’-> Macchina di Turing

Una funzione si dice calcolabile in modo algoritmico (o calcolabile in modo effettivo, o effettivamente calcolabile) se esiste un algoritmo che consente di calcolarne i valori per tutti gli argomenti.
Turing affrontò il problema di fornire un equivalente rigoroso del concetto intuitivo di algoritmo definendo un modello dell'attività di un essere umano che stia eseguendo un calcolo di tipo algoritmico.
Le MT sono macchine astratte nel senso che, nel caratterizzarle, non vengono presi in considerazione quei vincoli che sono fondamentali se si intende progettare una macchina calcolatrice reale (ad esempio, le dimensioni della memoria, i tempi del calcolo, e così via), e soprattutto nel senso che esse sono definite a prescindere dalla loro realizzazione fisica (cioè, dal tipo di hardware utilizzato). Vale a dire, che cosa sia una MT dipende esclusivamente dalle relazioni funzionali che sussistono tra le sue parti, e non dal fatto di poter essere costruita con particolari dispositivi materiali.

# TURING

**PROBLEMA della DECIDIBILITA'**

Trovare un algoritmo ("procedura meccanica") che sia in grado di decidere per una proposizione logica del primo ordine se questa è sempre vera (a prescindere da qualunque interpretazione semantica). A metà degli anni '30, lo statunitense Alonzo Church e l'inglese Alan Turing affrontarono in modo indipendente il secondo problema di Hilbert (l'Entscheidungsproblem). Nel 1936, la conclusione a cui giunsero entrambi quasi contemporaneamente (Church precedette Turing di pochi mesi) fu che era impossibile realizzare un algoritmo capace di decidere se una qualunque proposizione dell'aritmetica era vera o falsa.

# Le Macchine di Turing
# (viste dalla …tastiera)

COMPONENTI della Macchina:

- Un nastro infinito, suddiviso in celle. Ogni cella può contenere un solo simbolo, tratto dall'alfabeto esterno (->insieme tasti dei caratteri);
- Una testina TLS capace di leggere un simbolo da una cella, scrivere un simbolo in una cella e muoversi di una posizione sul nastro, in entrambe le direzioni (D ,S,-);
- Un insieme finito Q di stati, tali che la macchina si trova esattamente in uno di essi in ciascun istante (insieme dei tasti per cambiare i caratteri);
- Un programma, che specifica esattamente cosa fare per risolvere un problema (Regole applcate per un numero finito di passi);

Possiamo vedere il programma eseguito da una Macchina di Turing come una funzione f : S xQ ->S x Px Q dove P:=insieme (S, D,F).

# MDT UNIVERSALE

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

Il contributo di Turing non è solo 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 macchina di Turing;
- Scoprire che esistono problemi che le macchine di Turing non possono risolvere;
- Evidenziare che se una macchina di Turing non può risolvere un certo problema, nessuna altra macchina di nessun tipo potrà risolverlo.

# Turing: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.

Tx(n, m) = 1 Se Tn(m) si ferma

Tx(n, m) = 0 Se Tn(m) non si ferma

# Dimostrazione del Teorema dell'arresto

M a se n=m (con metodo della diagonalizzazione)
Tx(n, n) = 1 Se Tn(n) si ferma
Tx(n, n) = 0 Se Tn(n) non si ferma

Ma con questa ipotesi restrittiva, Tx diventa funzione di una singola variabile, per cui: Tx(n) = 0 Se Tn(n) non si ferma Ma questo deve chiaramente valere anche per n=x Per cui Tx(x) = 0 se Tx(x) non si ferma. Il che è un assurdo. NOTIAMO CHE:

Siamo giunti a un assurdo equivalente al concetto di indecidibilità di Godel.

IMPLICAZIONI:

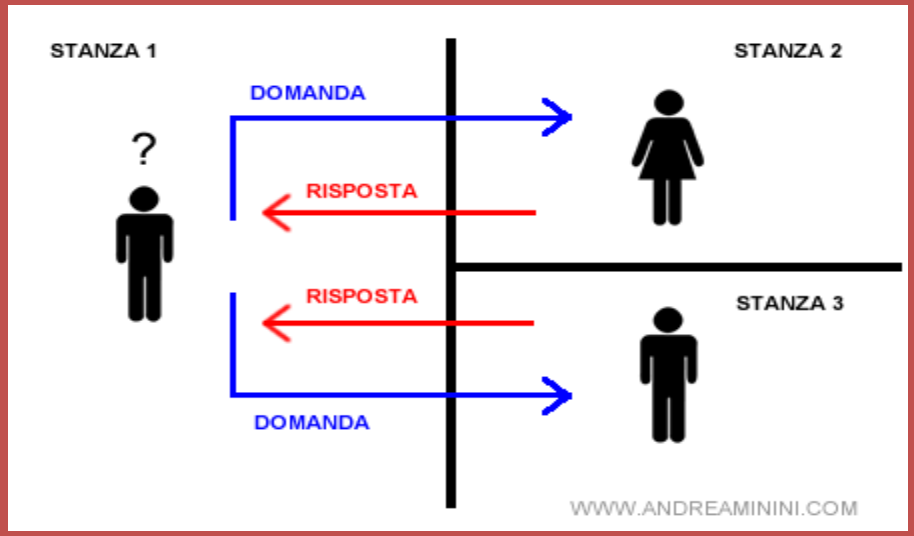
- esistono problemi non computabili, che un calcolatore non è in grado di risolvere.
- Non esistono programmi che possano controllare in modo automatico la correttezza di altri programmi.

Il problema dell'arresto è INDECIDIBILE

# Test di Turing-> IMITATION GAME

"Credo che entro cinquant'anni sarà possibile programmare computer in grado di partecipare al Gioco dell'imitazione così bene che un interrogante medio non avrà più del 70% di possibilità di operare la corretta identificazione dopo cinque minuti di domande".

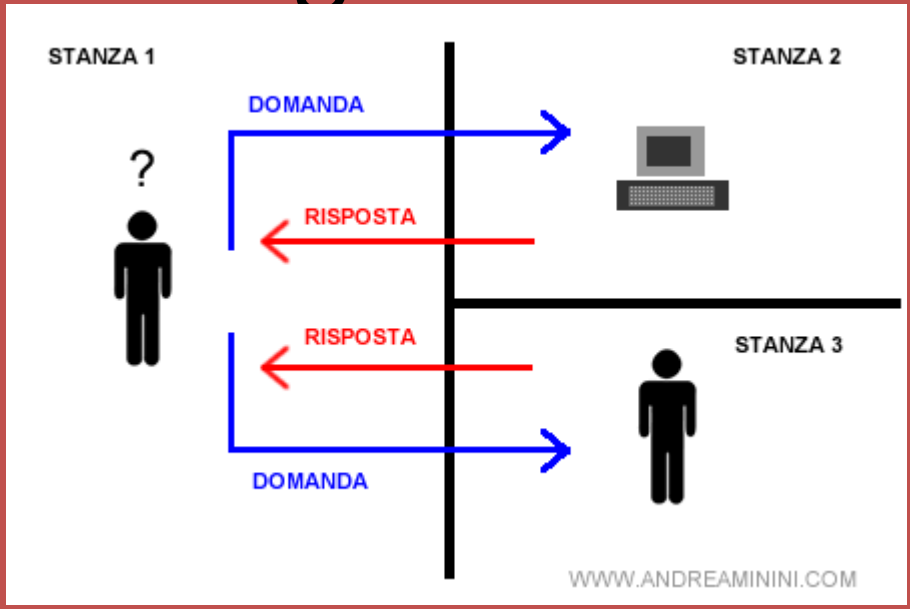
# Test di Turing per l’Intelligenza artificiale




TEST DI TURING:PRIMA PARTE

1) Lo scopo delle domande è di capire il sesso delle altre due persone e capire quale sia l'uomo e quale sia una donna. Esempio. Una domanda tipica può essere "hai capelli lunghi o corti?", "porti la gonna o i pantaloni?". Ripetendo il gioco N volte, l'intervistatore sbaglia il sesso dei partecipanti per X volte. Il suo tasso di errore è pari a X/N.

# Test di Turing2




TEST DI TURING(SECONDA PARTE)

Nella seconda fase del test sostituiamo uno dei due partecipanti con un computer.

Ora l'intervistatore deve capire se a rispondere è un uomo oppure una macchina. la sessione del test di Turing con il computer Il processo è una macchina. Alla fine del gioco dovrà identificare i partecipanti basandosi esclusivamente sulle loro risposte scritte. Il gioco si ripete N volte e l'intervistatore sbaglia l'identificazione dei partecipanti per Z volte, ottenendo un tasso di errore Z/N.

# Superamento del test di Turing?

Eugene Goostman” è (nella seconda parte del Test ) il primo software ad avere passato il cosiddetto “test di Turing”, convincendo un’arbitro su tre di essere un ragazzino di 13 anni di origini ucraine in grado di parlare un inglese scolastico. Eugene ( che utilizza chat box testuali), come qualsiasi altro software fino a ora sottoposto al test, è comunque ancora molto distante dal superare i requisiti e i limiti posti da Turing, che immaginava la realizzazione di un’intelligenza artificiale vera e propria, in grado di pensare autonomamente e non solo di gestire, con una serie di algoritmi, una chat.