STORIA DELLE CONOSCENZE SCIENTIFICHE SULL'UOMO E SULLA NATURA

A.A. 2016–2017

Prof. Roberto Giuntini, PhD.

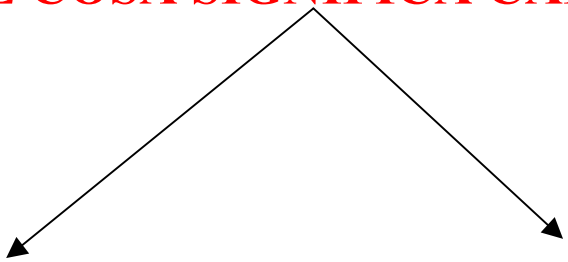
Introduzione alla storia dell'intelligenza artificiale e della robotica

Modulo II: Le macchine di Turing

3. LE MACCHINE DI TURING

Turing cercò di fornire una risposta *matematica* al problema

CHE COSA SIGNIFICA CALCOLARE?



Essere umano **Macchina**

Per calcolare ci si serve:

1) di un certo numero finito di simboli che possono essere sistemati in un certo modo, per esempio, su un foglio di carta. Sul foglio di carta si può **scrivere** o **cancellare** i simboli dell'alfabeto un **numero finito di volte**.

Si può inoltre:

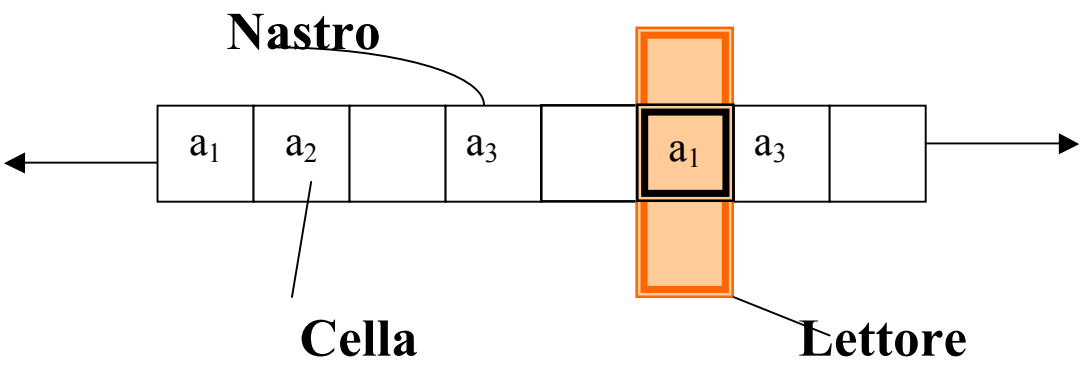
2) modificare il proprio campo visivo un **numero finito di volte**;
3) ricordando un **numero finito di istruzioni** e di atti già compiuti, si passa da una combinazione iniziale di simboli (che rappresentano gli argomenti iniziali della funzione da calcolare) a una combinazione finale di simboli (che rappresenta il risultato della funzione).

Intuitivamente: Una **MACCHINA DI TURING** (MT) è un modello meccanico dei processi (1)-(3); questo modello è ottenuto mediante opportune astrazioni, semplificazioni e idealizzazioni che però si possono considerare inessenziali.

Una MT è una macchina matematicamente idealizzata; è un **oggetto matematico**, non fisico.

Intuitivamente: Una MT dispone di:

1) **alfabeto finito**, cioè un insieme finito di simboli;
2) **nastro** potenzialmente infinito nei due sensi, diviso in **celle**; ciascuna cella può contenere al massimo un simbolo;
3) **lettore** ("occhio") capace di osservare solo una cella alla volta;
4) **memoria** capace solo di un certo numero finito di **stati**, detti **stati interni**.

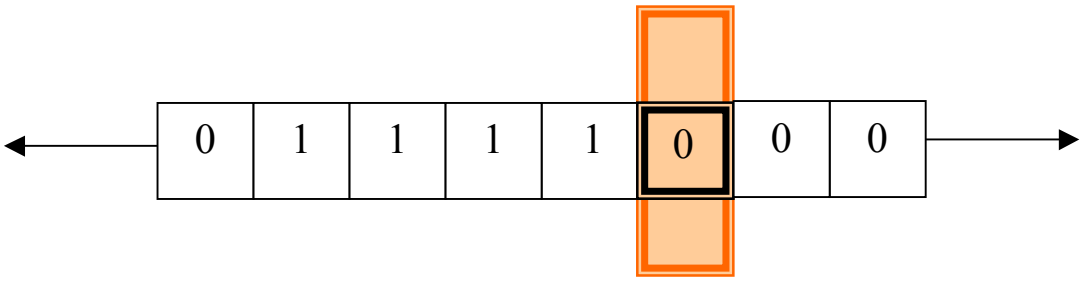


Si assume che la macchina funzioni per **passi elementari successivi**. Più precisamente, in un passo, la macchina può:

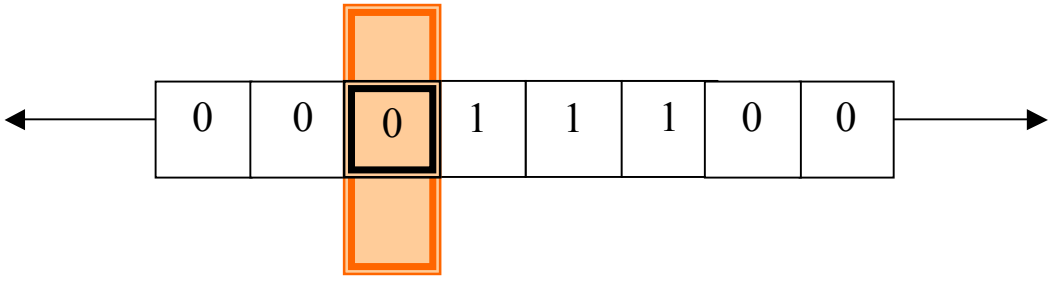
a) stampare o cancellare un solo simbolo, e passare eventualmente da un certo stato interno a un altro stato interno;
b) spostarsi di un passo a sinistra o a destra, e passare eventualmente da un certo stato interno a un altro stato interno;
c) fermarsi.

Ulteriori semplificazioni e convenzioni:

a) L'alfabeto contiene due soli simboli: 0 e 1;
b) Il lettore rimane fermo, mentre il nastro si muove;
c) Una cella si considera vuota quando contiene il simbolo 0;
d) Il nastro contiene sempre un numero finito di simboli 1;
e) **Prima dell'avvio**, il lettore della macchina si trova sempre su una cella vuota che alla sua destra ha solo celle vuote, e immediatamente alla sua sinistra c'è una cella che contiene l'"ultimo" simbolo 1 dell'**input**.



d) Alla fine (supposto che la macchina si fermi!), il lettore si trova sulla cella vuota immediatamente a sinistra del primo simbolo 1 dell'**output**.



Caratteristiche di una MT:

a) Il numero degli stati (interni) è finito;
b) il suo comportamento è completamente determinato dal suo stato interno e dall'input. Dato il suo **stato iniziale** e l'**input**, la macchina opera in modo completamente **deterministico.**

Possiamo descrivere il comportamento di una MT attraverso una **tavola** (**di transizione degli stati**). Identificheremo la TM proprio con la tavola che descrive il suo comportamento.

La tavola è costituita da **5 colonne**.

- Nella **prima colonna** (indicata con **S#**) indicheremo lo stato "attuale" della macchina;
- Nella **seconda colonna** (indicata con **R**) indicheremo ciò che la macchina **legge** (cioè 1 o 0);
- Nella **terza colonna** (indicata con **W**) indicheremo ciò che la macchina **scrive**;
- Nella **quarta colonna** (indicata con **M**) indicheremo il movimento che la macchina compie (cioè L, se si sposta a sinistra (di una cella); R, se si sposta a destra; H se si ferma);
- Nella **quinta colonna** (indicata con **N#**) lo stato in cui passa la macchina.

Convenzione: Indicheremo gli stati con numeri naturali > 0.

Quindi, una riga della tavola di una MT avrà, per esempio, la forma seguente:

S#	R	W	M	N#
2	1	0	R	3

Si legge:
Se ti trovi nello stato 2 e leggi il simbolo 1, allora stampa il simbolo 0 (cioè cancella), vai a destra [n.b.: è il nastro che si muove] e portati nello stato 3.

Esempio 1: **MT0**
Macchina che, qualsiasi cosa osservi sulla cella in cui è posizionata, cancella quello che trova, cioè stampa (o ristampa se c'è già) 0 e si ferma.

S#	R	W	M	N#
1	0	0	H	1
1	1	0	H	1

Esempio 2: **MT1**
Macchina che, qualsiasi cosa osservi sulla cella in cui è posizionata, stampa 1(o ristampa se c'è già) e si ferma.

S#	R	W	M	N#
1	0	1	H	1
1	1	1	H	1

Esempio 3: **MTR**
Macchina che, qualsiasi cosa osservi sulla cella in cui è posizionata, si sposta a destra di una cella e si ferma

S#	R	W	M	N#
1	0	0	R	2
1	1	1	R	2
2	0	0	H	2
2	1	1	H	2

Esempio 4: **MTR1**

Macchina che ricerca, per fermarsi, la prima cella a sinistra che contenga 1.

S#	R	W	M	N#
1	0	0	R	2
1	1	1	R	2
2	0	0	R	2
2	1	1	H	2

MTR1 può non fermarsi mai.

Vediamo alcune MT scritte nel linguaggio di "Turing Machine Simulator".

Esempio 5: **MTCOPIA**

```
{copia una stringa di simboli 1 a destra della cella di partenza}

{S#=State  R=Read  W=Write  M=Move  N#=New State }


{ S# R  :  W  M  N# }

  1  0  :  0  R  2
  1  1  :  1  R  2
  2  0  :  0  L  3
  2  1  :  1  R  2
  3  1  :  0  L  4
  3  0  :  0  H  3
  4  0  :  0  L  5
  4  1  :  1  L  4
  5  0  :  1  R  6
  5  1  :  1  L  5
  6  0  :  0  R  7
  6  1  :  1  R  6
  7  0  :  1  L  3
  7  1  :  1  R  7
```

Come rappresentiamo i **numeri naturali** con le macchine di Turing?

Il numero n viene rappresentato con n+1 simboli 1

Così:

0 ⟶ 1
1 ⟶ 11
2 ⟶ 111

Dato $n \in \mathbb{N}$, indicheremo la rappresentazione di n con $\overline{n}$

Nel caso di funzioni binarie, l'input viene rappresentato da due stringhe di simboli 1, separate da 0.

Nel caso di funzione n-arie, , l'input viene rappresentato da n stringhe di simboli 1, separate da 0.

Esempio 5: **MTEUCLIDE**

Implementazione per il calcolo del **MCD** di due numeri.

```
{ S#  R   :   W   M   N#}

  1   0   :   0   R   1
  1   1   :   0   R   2
  2   0   :   0   R   12
  2   1   :   1   R   3
  3   0   :   0   R   4
  3   1   :   1   R   3
  4   0   :   0   L   5
  4   1   :   1   R   4
  5   0   :   1   L   9
  5   1   :   0   L   6
  6   0   :   0   L   7
  6   1   :   1   L   6
  7   0   :   1   R   8
  7   1   :   1   L   7
  8   0   :   1   L   10
  8   1   :   0   R   3
  9   0   :   1   L   10
  9   1   :   1   L   9
 10   0   :   0   H   10
 10   1   :   0   L   11
 11   0   :   1   L   1
 11   1   :   1   L   11
 12   0   :   0   H   12
 12   1   :   1   R   12
```

Esempio 6: **MTSOMMA**

Calcola la somma di due numeri (naturali)

```
{ S#  R   :   W   M   N# }

  1   0   :   0   R   1
  1   1   :   0   R   2
  2   0   :   1   R   3
  2   1   :   1   R   2
  3   0   :   0   L   4
  3   1   :   1   R   3
  4   0   :   0   H   4
  4   1   :   0   H   4
```

Intuitivamente: l'esecuzione di un computo da parte di una MT è una successione di configurazioni a partire dalla **configurazione iniziale** (data dalla posizione in cui la macchina si trova all'inizio, l'iscrizione che si trova sul nastro nel momento in cui la macchina parte, e uno stato iniziale) fino a una
configurazione finale (data dalla posizione in cui si ferma la macchina, dall'iscrizione del nastro in quel momento e dallo stato finale).

Possiamo dare la definizione generale di funzione *Turing-computabile*.

Definizione Una funzione numerica k-aria f^k: $N^k \rightarrow N$ si dice **Turing-computabile** se e solo se esiste una macchina di Turing M tale che per qualsiasi k-pla di numeri naturali $(n_1,...,n_k)$ esiste una computazione di M che soddisfa le seguenti condizioni:

a) la sua **condizione iniziale** è data dalla successione di cifre $\overline{n_1},...,\overline{n_k}$, intervallate dal simbolo vuoto (0) [input].

b) La sua **condizione finale** è data da $\overline{f^k(n_1,...,n_k)}$ [output].

Intuitivamente: Una funzione numerica f è Turing-computabile quando esiste una macchina di Turing che calcola il valore di f per ogni scelta possibile di argomenti.

3a. LA MACCHINA DI TURING UNIVERSALE

La caratteristica fondamentale delle MT è che ne esiste una che può "simulare'' il comportamento di tutte le altre [Turing 1936].

TEOREMA:(*esistenza della TM universale*)[Turing 1936]
Esiste una macchina di Turing *U* (detta *macchina di Turing universale*) tale che per ogni macchina di Turing *M*, la MT *U* può simulare il computo di *M* con input *n*, una volta che il nastro contenga come input una successione codificata (nell'alfabeto {0,1}) di istruzioni di *M* e di *n*. L'output di *U* è identico a quello di *M*.

3b. CONSEGUENZE DELL'ESISTENZA DELLA MTU

Dire che una MT Universale (**MTU**) è una MT che può simulare (incorporare) ogni MT è come dire che:

- C'è un **diagramma di flusso universale** *n* tale che, dati *m* diagrammi di flusso (scritti con un certo simbolismo) che descrivono l'esecuzione di *m* compiti specifici, *n* può riprodurre ciascuno degli *m* diagrammi specifici
- UMT è uno strumento **programmabile** di computazione "general purpose", che fornisce le basi logiche per la costruzione dei moderni computer. La sua universalità è garantita dalla distinzione fra le operazioni elementari, eseguite dall'**hardware**, e le istruzioni specificate da un dato programma, contenuto nel **software**.

A. Una MTU può essere **fisicamente implementata** su diversi tipi di hardware.

B. Ogni computer convenzionale è **logicamente** (non fisicamente) **equivalente** a una MTU.

Turing non ha dimostrato che una MTU può calcolare una qualsiasi funzione calcolabile da un computer con architettura. Turing ha dimostrato che

Una MTU può calcolare una qualsiasi funzione che una MT può calcolare.

3c. LA TESI DI TURING-CHURCH

Le macchine di Turing [“macchine logiche di calcolo” nell’originale articolo di Turing del 1936] possono fare tutto ciò che può essere descritto come “puramente meccanico”. In altri termini: se esiste una procedura effettiva per ottenere il valore di una funzione matematica, allora questa funzione può essere calcolata da una MT.

Le MT furono introdotte da Turing per analizzare l'*Entscheidungsproblem*. Esse quindi vogliono simulare un **essere umano impegnato in un calcolo**.

La tesi di Turing è plausibile?

Nel 1936 A. **Church** propose un'altra nozione formale (molto diversa rispetto a quella di Turing) di metodo effettivo. In seguito altre nozioni formali sono state proposte.

Risultato:

Una funzione è Turing-computabile sse è Church-computabile sse…..

Le nozioni di Turing-computabilità e di Church-computabilità individuano la stessa classe di funzioni.

Possiamo parlare allora di **TESI DI TURING-CHURCH** (TTC)

Se accettiamo la validità di TTC, allora il problema dell'esistenza e della non-esistenza di metodi effettivi si riduce (in matematica e logica) al problema dell'esistenza e della non-esistenza di Macchine di Turing.

3d. MACCHINE DI TURING E PROBLEMI DI DECIDIBILITÀ

Problema della fermata (*Halting problem*)

Esiste una macchina di Turing che sia in grado di stabilire (per ogni $n,m \in \mathbf{N}$) se la n-esima macchina di Turing M_n che agisce sul numero m si fermi oppure no ?

Intuitivamente:
Le macchine di Turing hanno la capacità di *autodescriversi.* In particolare, nel linguaggio di una macchina di Turing M sarà possibile esprimere il problema: la macchina M si ferma o non si ferma quando le viene chiesto di calcolare il valore di una funzione per certi argomenti?
Potremo allora porre alla stessa macchina la seguente domanda:

"Ti fermerai qualora ti venga chiesto se non ti fermerai?"

Risolvere positivamente il problema della fermata implicherebbe stabilire che ogni problema matematico è decidibile, perché esisterebbe un algoritmo che fornisce una risposta 1/0 (positiva/negativa) a un qualsiasi problema matematico.

Teorema di Turing (irrisolvibilità del problema della fermata)

Non esiste nessuna macchina di Turing che sia in grado di decidere se una macchina di Turing si fermerà.
In altri termini, non esiste nessuna macchina di Turing che, dato l'input (*n*,*m*), produca l'output 1 se la macchina di Turing di indice *n* e di input *m* si ferma; produca l'output 0 altrimenti.

Se accettiamo TTC, il Teorema di Turing si può riformulare così:

Non esiste nessun algoritmo universale per decidere i problemi matematici.

Turing ha dimostrato che l'*Entscheidungsproblem* di Hilbert non ha soluzione.

Per dimostrare che un problema matematico è *indecidibile* è sufficiente dimostrare che la sua soluzione è equivalente alla soluzione del problema della fermata.

Il Teorema di Turing ovviamente non dimostra l'impossibilità di decidere *tutti* i problemi matematici!

Il Teorema di Turing **non** dimostra che esistono alcuni problemi matematici indecidibili (a esclusione del problema della fermata!). Il Teorema di Turing non dice niente sulla (ir)risolvibilità di *specifici problemi matematici.*