

# Informatica Teorica



Anno Accademico 2022/2023  
Fabio Zanasi  
<https://www.unibo.it/sitoweb/fabio.zanasi>  
Seconda lezione

# Dove siamo

## **Cos'è un computer?**

In questa lezione introduciamo un primo modello astratto di computazione: la **macchina di Turing** (TM).

## **Ci sono limiti a ciò che i computer possono calcolare?**

# Alan Turing (1912-1954)

Conosciuto al pubblico generalista per la decrittazione del codice di *Enigma* durante la seconda guerra mondiale.

Prima della guerra (1936) pubblicò un articolo:

On Computable Numbers,  
with an Application to  
the *Entscheidungsproblem*



Questo lavoro è l'atto fondato dell'informatica.

# La domanda di Turing

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO  
THE ENTSCHEIDUNGSPROBLEM

*By A. M. TURING.*

[Received 28 May, 1936.—Read 12 November, 1936.]

The “computable” numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions

Che cos’è una **computazione**?

Che cosa significa per un problema essere **calcolabile**?

# Il metodo di Turing

Osserviamo come un essere umano esegue una computazione.



- Segue un insieme di **regole**.
- Vengono letti e scritti dei **simboli** (su un supporto, ad esempio un foglio di carta).
- L'azione eseguita dalla persona dipende dal simbolo che viene esaminato.

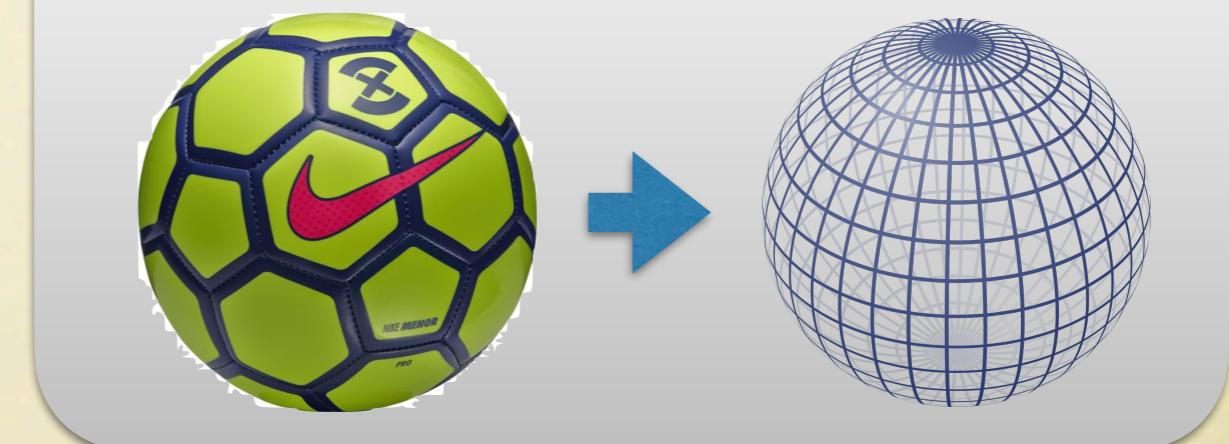
# Il metodo di Turing

Il metodo di Turing ha precedenti illustri, e più  
essere riassunto come il tentativo di identificare le  
proprietà **essenziali** del processo di computazione,  
a discapito di ciò che invece è **irrilevante**.

Reminder: Aristotele  
Essenziale  
Vs  
Accidentale



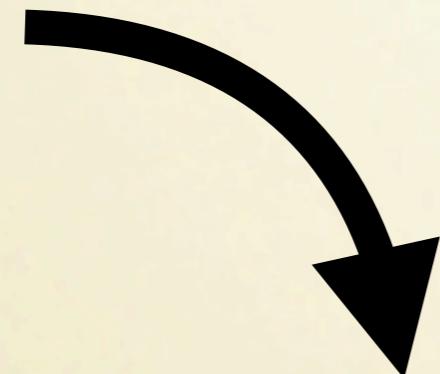
Reminder: la geometria



# Cos'è essenziale in una computazione?

## Astraiamo i dati

$$\begin{array}{r} 26 \\ \times 32 \\ \hline 52 \\ 780 \\ \hline 832 \end{array}$$



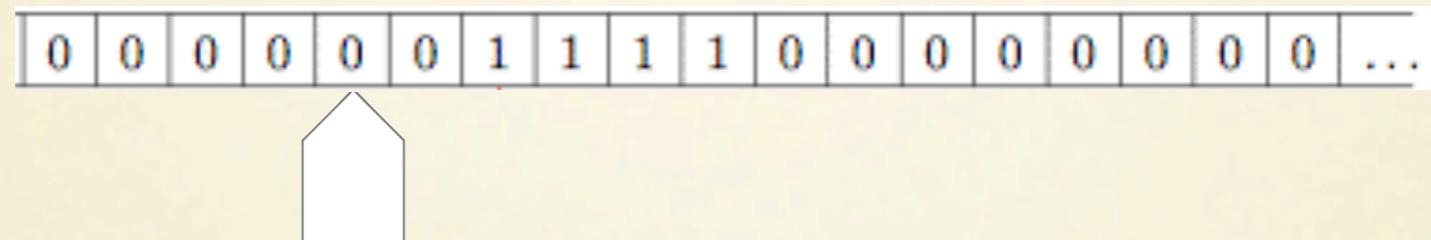
2	6	$\times$	3	2	=	5	2	$+$	7	8	0	=	8	3	2
---	---	----------	---	---	---	---	---	-----	---	---	---	---	---	---	---



0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	...
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	-----

# Cos'è essenziale in una computazione?

## Astraiamo le azioni

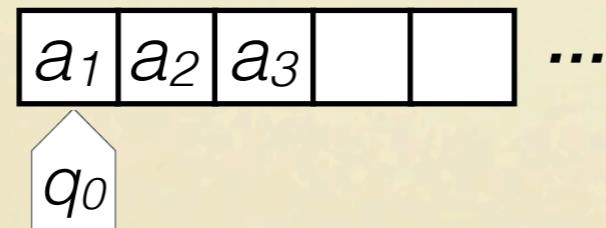


- Scrivi il simbolo 0
- Scrivi il simbolo 1
- Spostati una cella verso destra
- Spostati una cella verso sinistra
- Osserva il simbolo corrente e scegli il prossimo passo di conseguenza.
- Fermati.

# La Macchina di Turing

# La macchina di Turing: definizione informale

- Una macchina di Turing ha un **nastro**, infinito sul lato destro. Il nastro è diviso in **celle**.
- Ogni cella del nastro può contenere un simbolo o essere vuota.
- Una macchina di Turing ha una **testina** che si muove sul nastro. La testina è sempre in un certo **stato**.
- Una **configurazione** è una `istantanea' di un passo di computazione della macchina.
- La **configurazione iniziale** vede la testina posizionata nella prima cella di sinistra, nello stato iniziale  $q_0$ . Il nastro è vuoto, ad eccezione di una stringa di input  $a = a_1 a_2 \dots a_m$  che va ad occupare le prime  $m$  celle.



# La macchina di Turing: definizione informale

- Supponiamo che la macchina si trovi in una configurazione data. Come decide il prossimo passo di computazione?
- La testina legge il contenuto della cella, e sulla base della lettura la macchina può compiere una delle seguenti azioni:
  - **Fermarsi.**
  - **Cambiare stato, scrivere** un nuovo simbolo nella cella corrente (o lasciarla vuota), e **spostarsi** nella cella immediatamente a sinistra o a destra dell'attuale.
- Il tipo di azione da prendere è dettato dal **programma** della macchina.



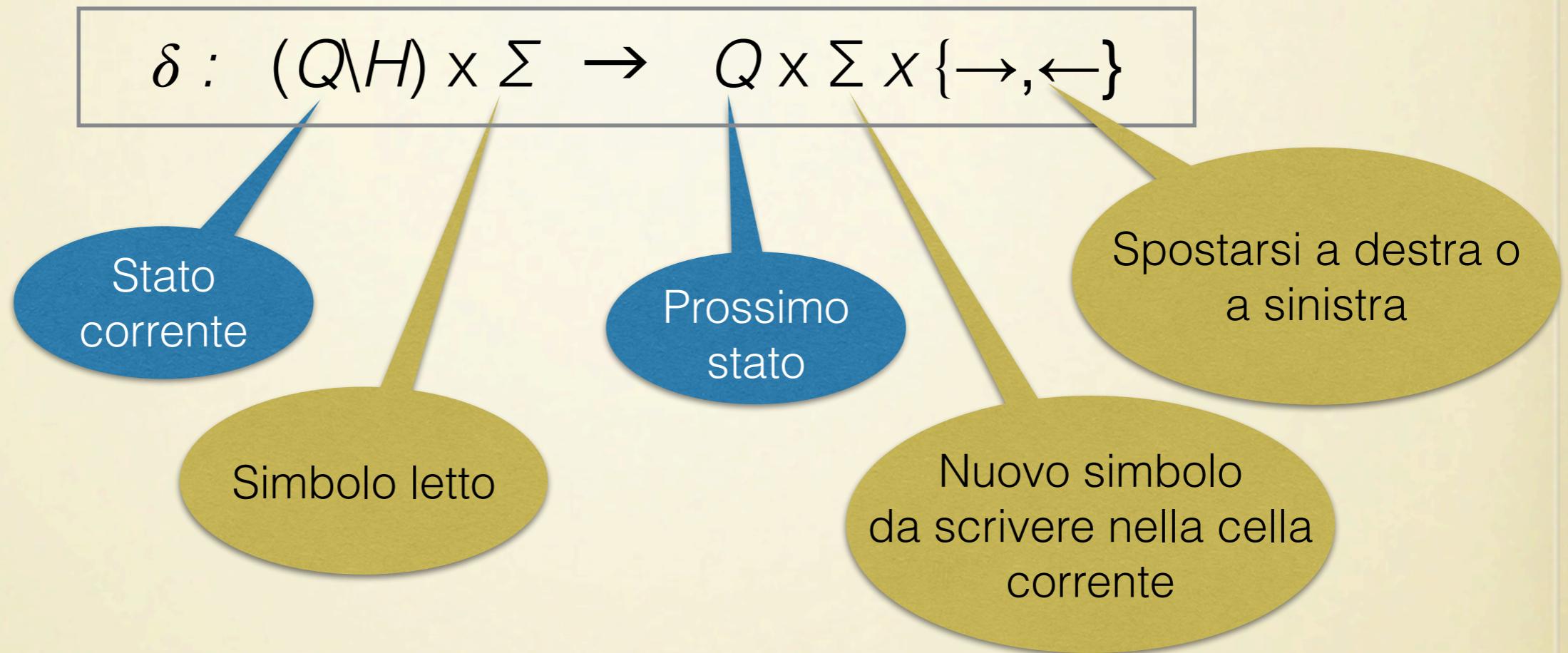
# La macchina di Turing: definizione formale

Una macchina di Turing è una tupla  $\langle \Sigma, Q, q_0, H, \delta \rangle$

- $\Sigma$  é un **alfabeto** finito di simboli, che include un simbolo speciale  $\sqcup$  che indica una cella vuota.
- $Q$  é un insieme finito di **stati**.
- $q_0 \in Q$  é lo **stato iniziale**.
- $H \subseteq Q$  é l'insieme degli **stati accettanti (o finali)**.
- $\delta$  é la **funzione di transizione**

$$\delta : (Q \setminus H) \times \Sigma \rightarrow Q \times \Sigma \times \{\rightarrow, \leftarrow\}$$

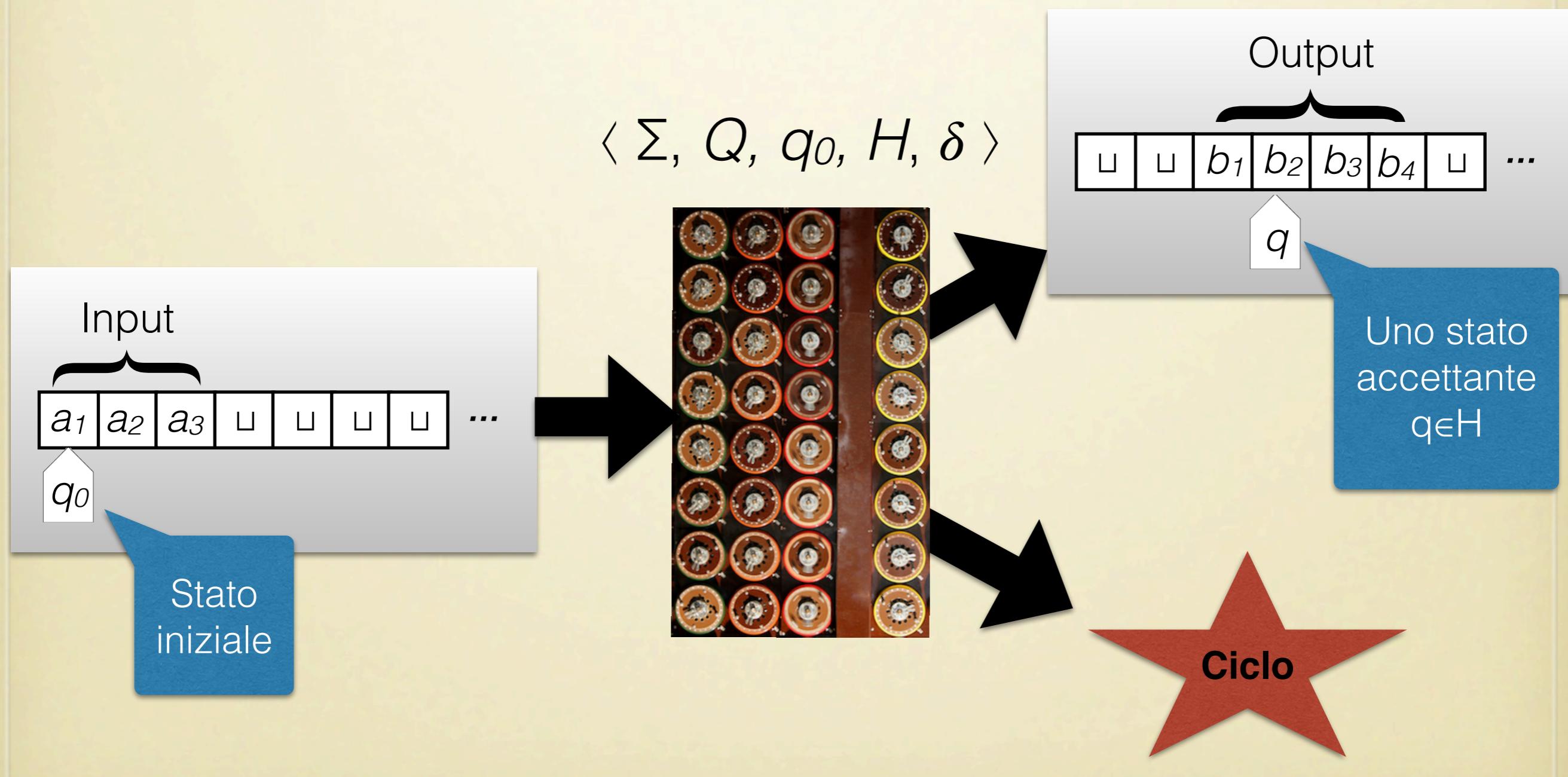
# La funzione di transizione



- $\delta$  esprime il programma che governa il funzionamento della TM.
- $\delta$  è una funzione *totale*.
- Possiamo scrivere la definizione di  $\delta$  come un insieme di quintuple:  
 $\{ \langle q_i, a, q_j, \sqcup, \rightarrow \rangle, \langle q_k, b, q_l, b, \rightarrow \rangle, \langle q_i, b, q_k, a, \leftarrow \rangle, \dots \}$

# Che cosa calcola una TM?

L'alfabeto  $\Sigma_I$  di simboli di *input/output* é un sottoinsieme di  $\Sigma$  non contenente  $\sqcup$ .



# Un esempio semplice

$\langle \Sigma, Q, q_0, H, \delta \rangle$

$$\Sigma = \{1, \sqcup\}$$

$$\Sigma_I = \{1\}$$

$$Q = \{q_0, q_1, h\}$$

$$H = \{h\}$$

$$\delta(q_0, \sqcup) = (h, 1, \leftarrow)$$

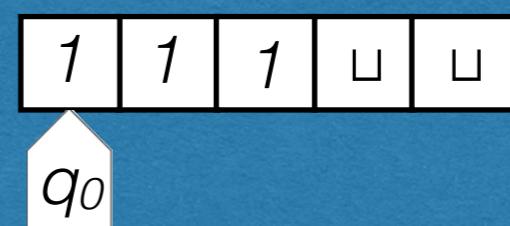
$$\delta(q_1, \sqcup) = (h, 1, \leftarrow)$$

$$\delta(q_0, 1) = (q_1, 1, \rightarrow)$$

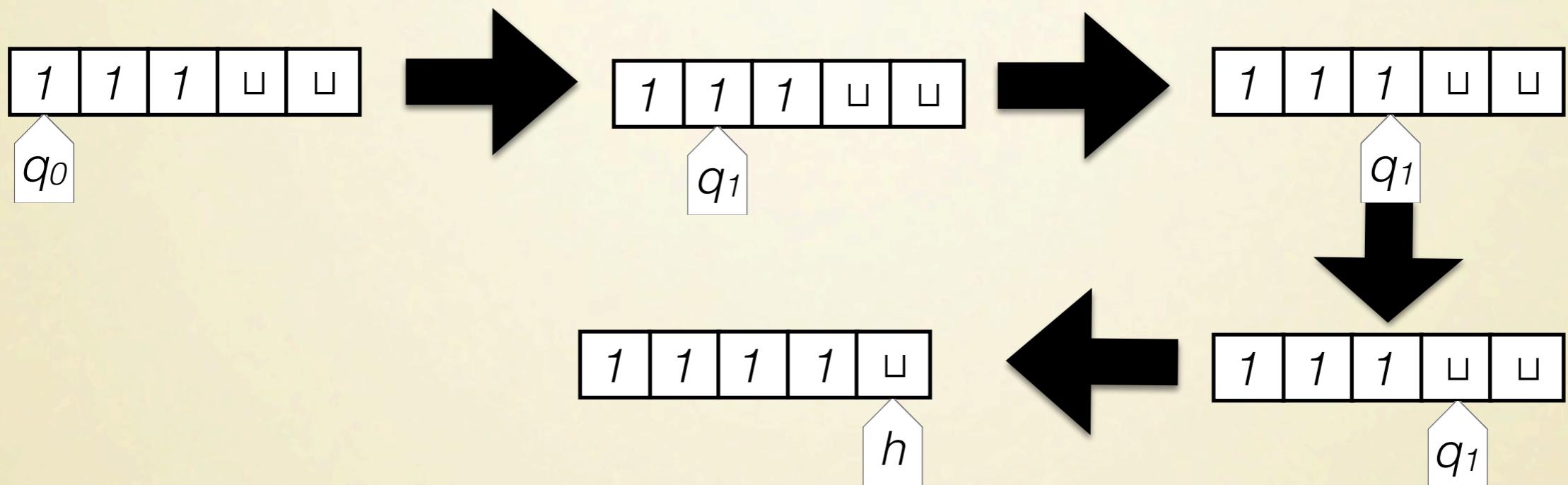
$$\delta(q_1, 1) = (q_1, 1, \rightarrow)$$

## Esercizio

Che cosa calcola?  
Provala sull'input:



# Un esempio semplice



$$\delta(q_0, \square) = (h, 1, \rightarrow) \quad \delta(q_0, 1) = (q_1, 1, \rightarrow)$$

$$\delta(q_1, \square) = (h, 1, \rightarrow) \quad \delta(q_1, 1) = (q_1, 1, \rightarrow)$$

# Rappresentazione a diagrammi

La TM che “aggiunge 1”

$$\Sigma = \{1, \sqcup\}$$

$$\Sigma_l = \{1\}$$

$$Q = \{q_0, q_1, h\}$$

$$H = \{h\}$$

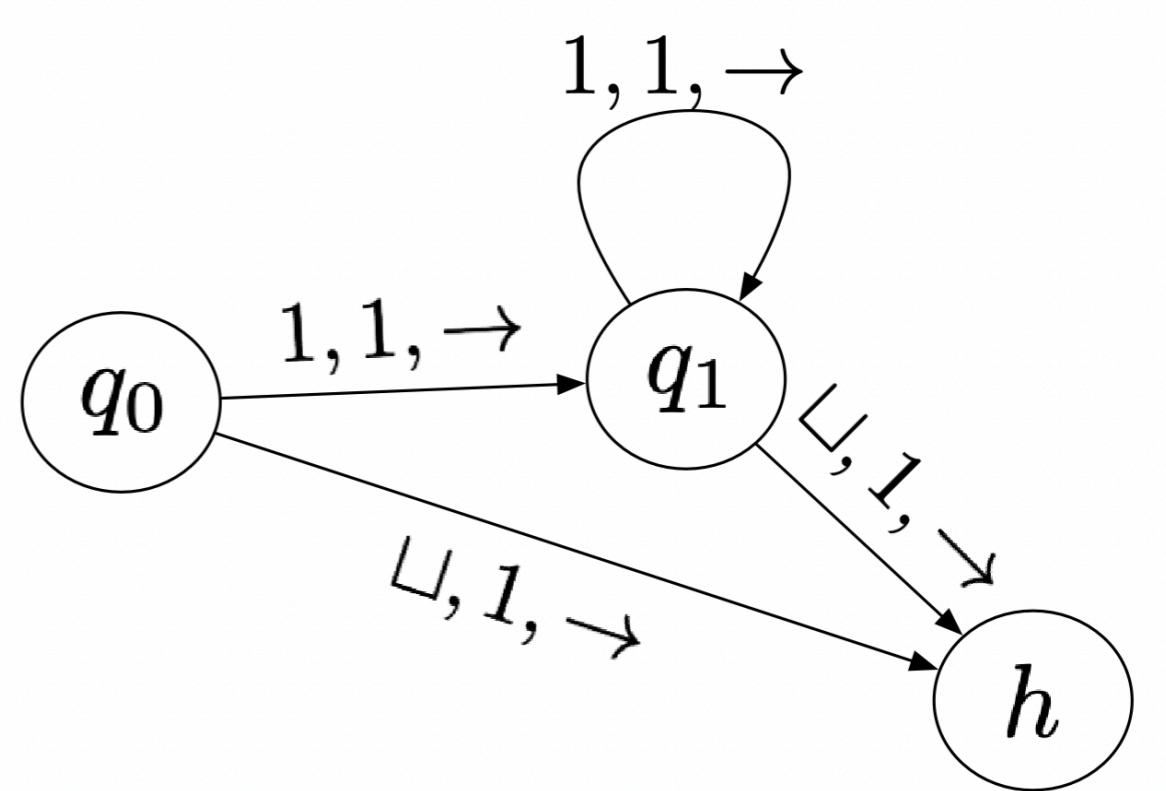
$$\delta(q_0, \sqcup) = (h, 1, \rightarrow)$$

$$\delta(q_1, \sqcup) = (h, 1, \rightarrow)$$

$$\delta(q_0, 1) = (q_1, 1, \rightarrow)$$

$$\delta(q_1, 1) = (q_1, 1, \rightarrow)$$

Disegniamo  
solo configurazioni  
**raggiungibili**

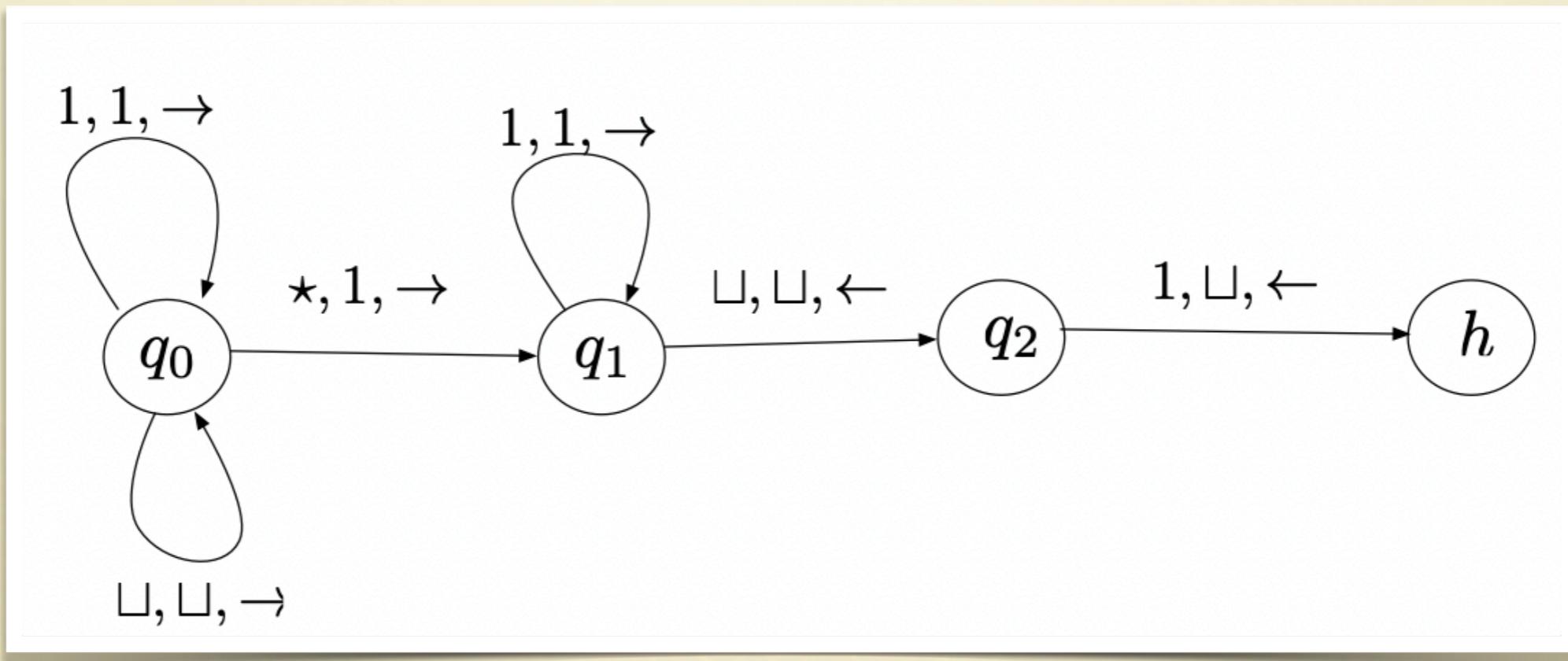


# Esempio: addizione unaria

$$\Sigma = \{1, \star, \sqcup\} \quad \Sigma_l = \{1, \star\}$$

$$Q = \{q_0, q_1, q_2, h\}$$

$$H = \{h\}$$



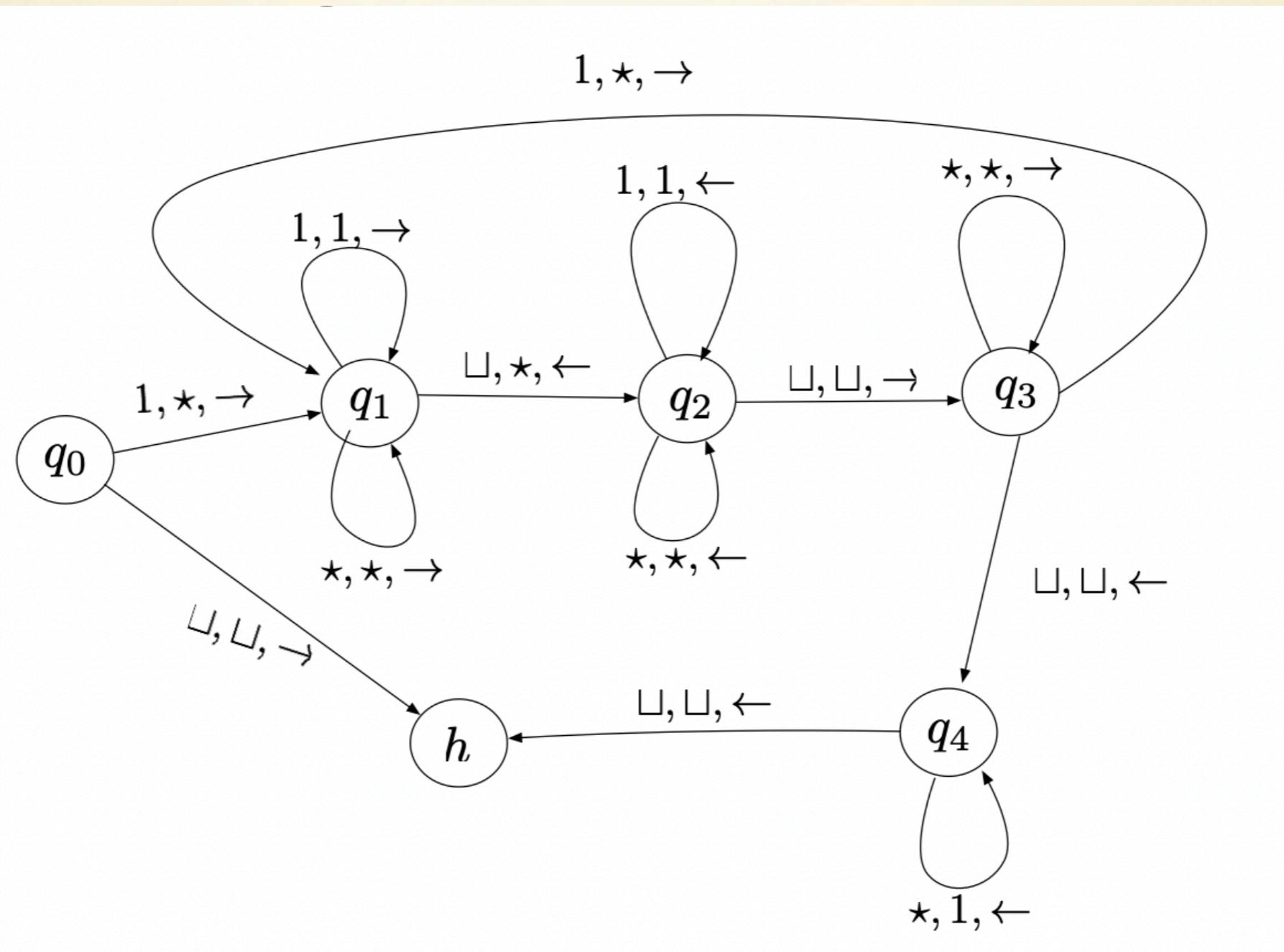
# Esempio: moltiplicazione unaria per due

$$\Sigma = \{1, \star, \sqcup\}$$

$$\Sigma_l = \{1\}$$

$$Q = \{q_0, q_1, q_2, q_3, q_4, h\}$$

$$H = \{h\}$$



- Finora abbiamo visto macchine di Turing che computano funzioni da  $\mathbb{N}^k$  a  $\mathbb{N}$ .
- Come è possibile usare una macchina di Turing per calcolare problemi più complessi, o semplicemente di tipo diverso?

# Espressività delle macchine di Turing

# Problemi di decisione

Molto spesso un problema che vogliamo risolvere usando uno strumento di calcolo può essere espresso come un **problema di decisione**.

- Dato un grafo, è strettamente connesso?
- Dato un insieme di città e una descrizione delle distanze tra di esse, è possibile visitarle a turno in modo tale che la lunghezza del percorso è minore di una data costante  $d$ ?
- Dato un insieme di equazioni, ha una soluzione?
- Date alcune forme da tagliare da un foglio di alluminio, possiamo ricavarle tutte da un singolo rettangolo di taglia  $n \times m$ , dove  $n$  e  $m$  sono dati?

# Problemi di decisione

In astratto, ciascun problema ha due componenti.

## Il problema del commesso viaggiatore

- Dati: un insieme di città e una descrizione delle distanze tra loro.
- Domanda si/no: Esiste un percorso di distanza meno di  $d$ ?

## Il problema delle equazioni

- Data: un sistema di equazioni
- Domanda si/no: il sistema di equazioni ha soluzione?

Instanze **positive**: dati per i quali la risposta è si.  
Instanze **negative**: dati per i quali la risposta è no.

# Problemi di decisione

Abbiamo visto che per calcolare la risposta ad un problema di decisione abbiamo bisogno di un programma che riceve in input dati del tipo corretto e da una risposta SI o NO come output.

## Come fare tutto ciò con le macchine di Turing?

- Abbiamo visto che le TM possono calcolare funzioni di tipo  $\mathbb{N}^k \rightarrow \mathbb{N}$
- Il passaggio chiave è **codificare** un problema di decisione come *la funzione caratteristica di un linguaggio formale*.

# Linguaggi formali: ripasso

- Dato un alfabeto  $\Sigma$  di simboli, un **linguaggio formale** é (semplicemente) un sottoinsieme di  $\Sigma^*$ .

- Esempio:

- Alfabeto  $\Sigma = \{1\}$
- Linguaggio delle stringhe di lunghezza pari.

La stringa vuota

$$L = \{\varepsilon, 11, 1111, 111111, \dots\}$$

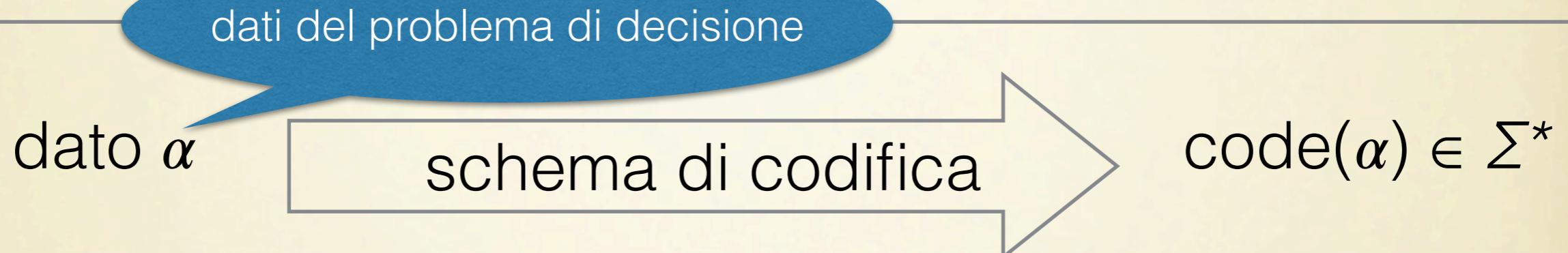
Insieme di *stringhe* di simboli in  $\Sigma$ .

- La **funzione caratteristica** di un linguaggio  $L$  é la funzione  $\chi_L : \Sigma^* \rightarrow \{0,1\}$  definita come:

$$\chi_L(x) = \begin{cases} 1 & \text{se } x \in L \\ 0 & \text{altrimenti} \end{cases}$$

# Domande?

# Dai problemi di decisione ai linguaggi formali



Il **linguaggio** che codifica il **problema di decisione** sarà:  
 $L = \{x \in \Sigma^* \mid x = \text{code}(\alpha) \text{ per qualche dato } \alpha, \text{ e } \alpha \text{ è un'istanza positiva del problema}\}$

## Proprietà che solitamente vogliamo per code(-)

- Se  $\alpha \neq \beta$  allora  $\text{code}(\alpha) \neq \text{code}(\beta)$
- Dovremmo poter verificare se  $x \in \Sigma^*$  è  $\text{code}(\alpha)$  per qualche  $\alpha$
- Dovremmo poter calcolare  $\alpha$  a partire da  $\text{code}(\alpha)$ .

(Quando si guarda alla complessità computazionale, altre proprietà diventano rilevanti: ad esempio quando la codifica è *ridondante* e calcolata in modo *efficiente*.)

# Linguaggi decidibili

**In conclusione, come facciamo a ragionare su un problema di decisione utilizzando una macchina di Turing?**

- Assumiamo una codifica del problema di decisione come un linguaggio  $L$  su un alfabeto  $\Sigma'$ .
- Vogliamo una TM  $\mathcal{M}$  con le seguenti proprietà:
  - l'alfabeto di input/output  $\Sigma_I$  è  $\Sigma'$ .
  - l'insieme degli stati finali è  $H = \{Y, N\}$ .
- Diciamo che  $\mathcal{M}$  **accetta** un input  $x \in \Sigma_I^*$  se la computazione ferma in stato Y.  $\mathcal{M}$  **rigetta**  $x$  se ferma in stato N.
- $\mathcal{M}$  **decide**  $L$  se:
  - Quando  $x \in L$ , allora  $\mathcal{M}$  accetta  $x$ .
  - Quando  $x \notin L$ , allora  $\mathcal{M}$  rigetta  $x$ .
- Un linguaggio (un problema di decisione) è **decidibile** se c'è una TM che lo decide.

# Linguaggi riconoscibili

C'è un modo ulteriore, più 'sottile', in cui una TM può essere associata ad un linguaggio (problema di decisione).

- Assumiamo una codifica del problema di decisione come un linguaggio  $L$  su un alfabeto  $\Sigma'$ .
- Vogliamo una TM con alfabeto di input/output  $\Sigma'$ .
- $\mathcal{M}$  **riconosce**  $L$  se:
  - Quando  $x \in L$ , allora  $\mathcal{M}$  si ferma (= raggiunge uno stato finale).
  - Quando  $x \notin L$ , allora  $\mathcal{M}$  non si ferma (= entra in un ciclo).
- Un linguaggio (un problema di decisione) è **riconoscibile** se c'è una TM che lo riconosce.

Decidibile → Riconoscibile

# Attenzione alla terminologia

Decidable = Recursive = Computable

Sipser

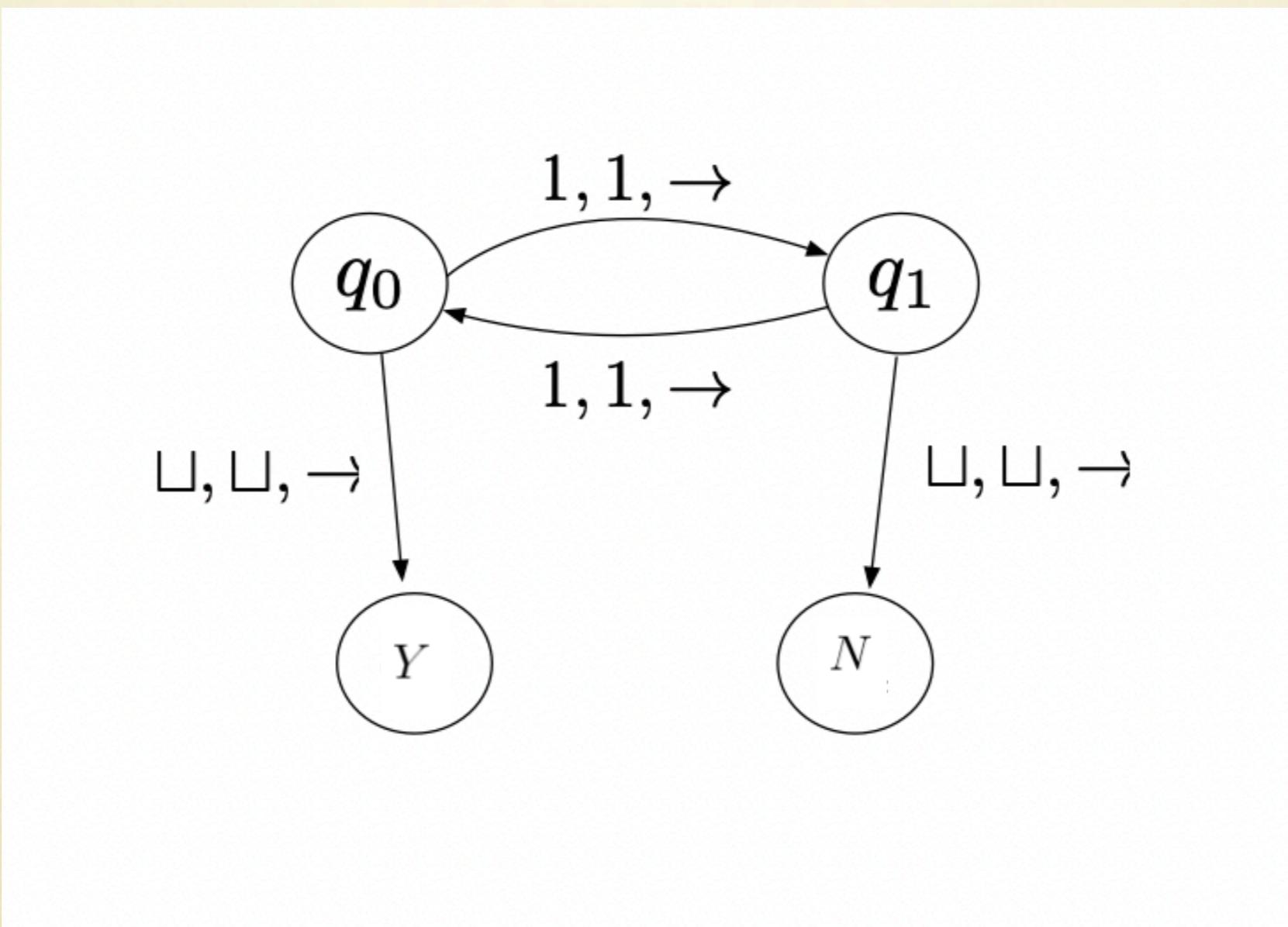
A. Church &  
S. Kleene ('30s)

C. R. Soare ('90s)

Recognisable = Recursively enumerable = Computably enumerable

# Esempio: stringhe di lunghezza pari

Il linguaggio delle stringhe di lunghezza pari sull'alfabeto  $\Sigma_1 = \{1\}$  è decidibile.



# Esempio: palindromi

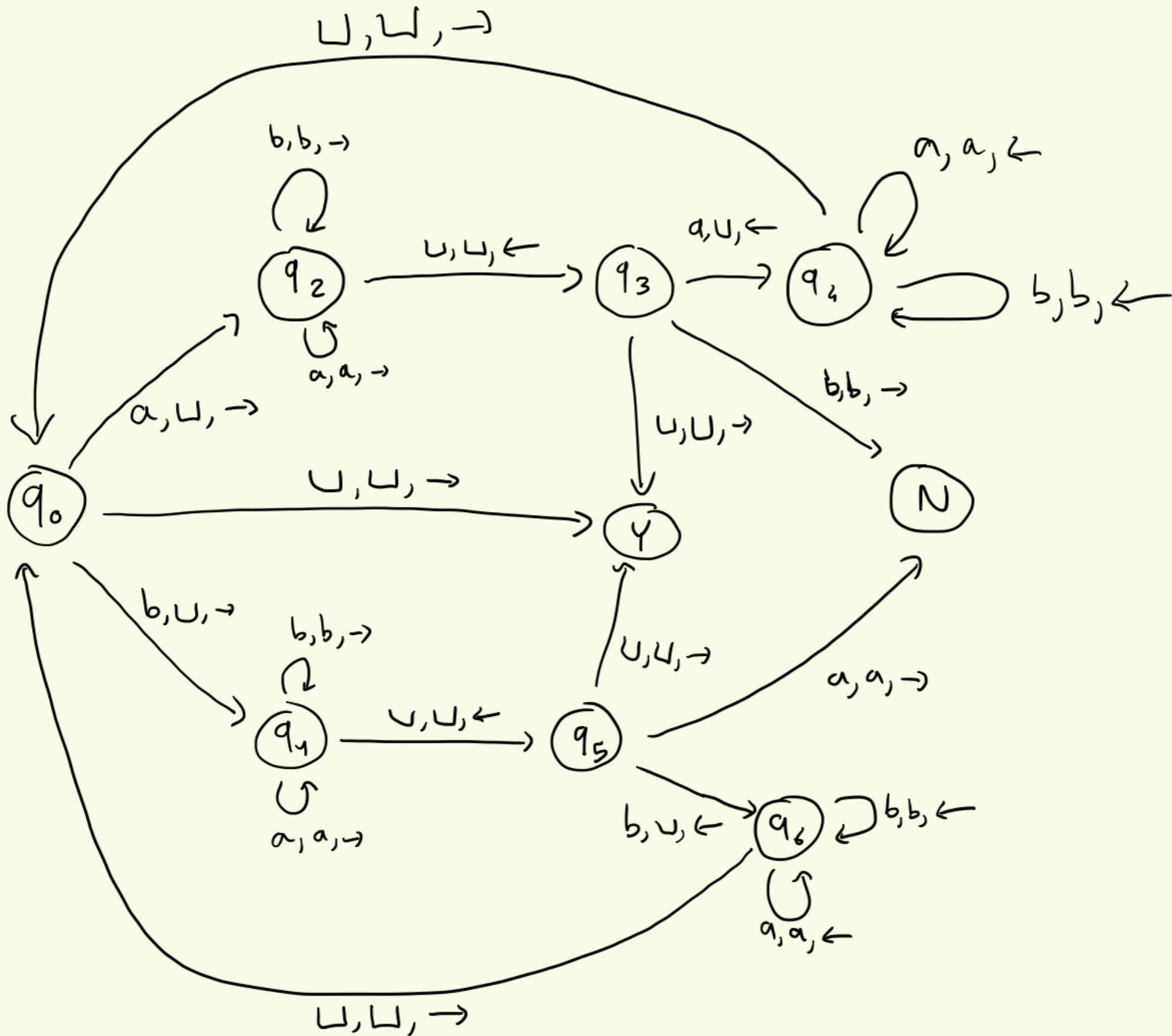
Il linguaggio dei palindromi su alfabeto  $\{a,b\}$  è decidibile.

aaabaaa

abababbababa

baaabbbabaaab

...



# E i linguaggi riconoscibili?

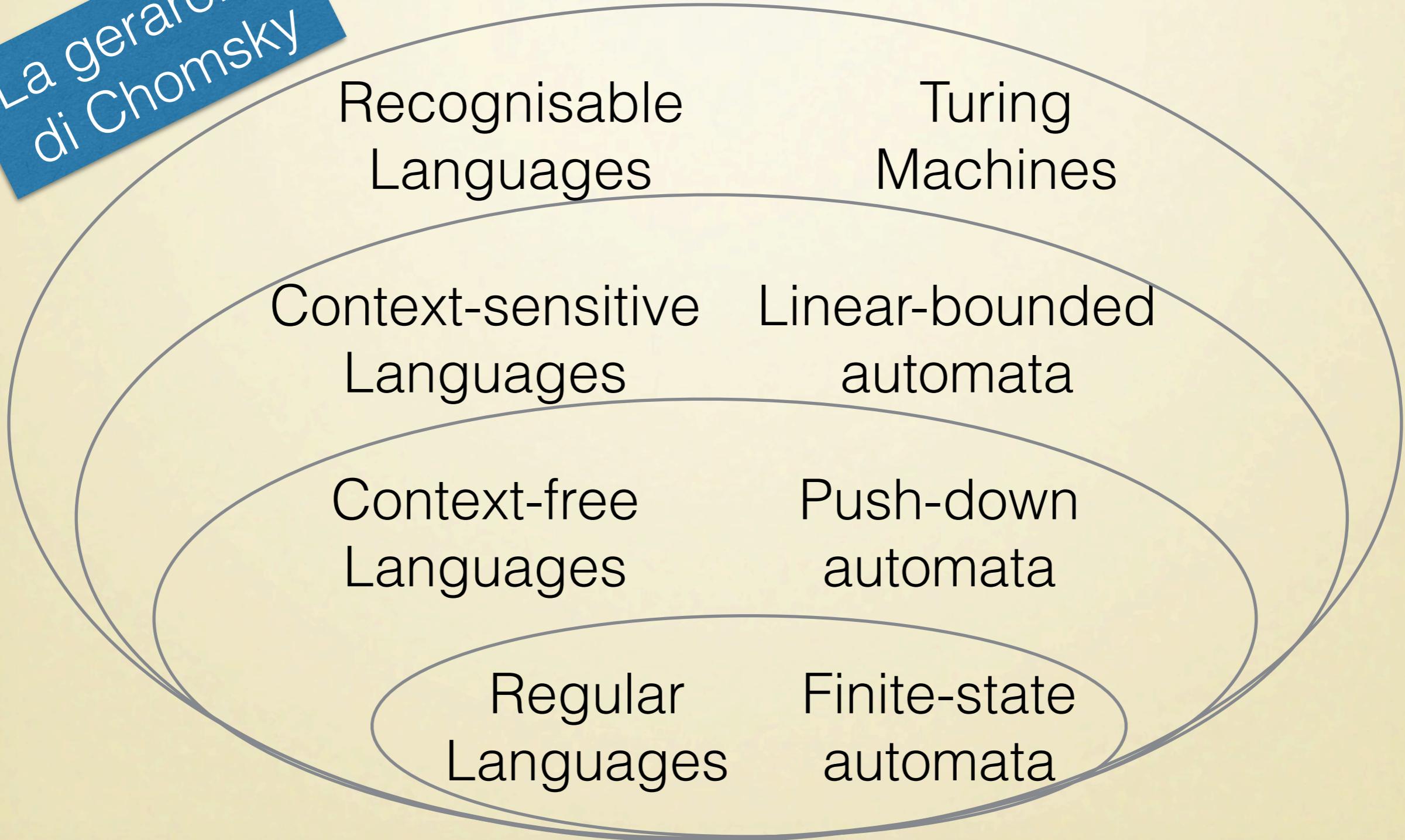
- Vedremo che molti problemi significativi non sono decidibili, ma solo riconoscibili.
- Altri non sono nemmeno riconoscibili!

Lo vedremo la prossima settimana.

# Macchine di Turing nel contesto della teoria dei linguaggi formali

# Macchine di Turing e automi

La gerarchia  
di Chomsky



# Hardware o software?

Una macchina di Turing è una macchina a stati.

Tuttavia avremmo potuto definirla come un linguaggio di programmazione.

Cinque tipi di istruzione:

- PRINT  $a$  per ogni  $a \in \Sigma$
- GO LEFT
- GO RIGHT
- GO TO STEP  $i$  IF  $a$  IS SCANNED per tutti i naturali  $n$
- STOP

**Non é importante l'implementazione fisica, ma l'espressività del modello astratto.**