MACCHINE DI TURING
TESI DI CHURCH-TURING
IL PROBLEMA DELL'ARRESTO

# INFORMATICA

# IL DECIMO PROBLEMA DI HILBERT

- Nel 1900 David Hilbert pose 23 problemi al tempo insoluti che influenzarono profondamente la matematica del XX secolo
- La storia del decimo problema di Hilbert è strettamente legata alla formalizzazione del concetto di algoritmo, della nozione di computabilità e la nascita dell'informatica

David Hilbert, 1912

# IL DECIMO PROBLEMA DI HILBERT

Equazione algebrica in una o più incognite
a coefficienti interi delle quale si cercano
le soluzioni intere

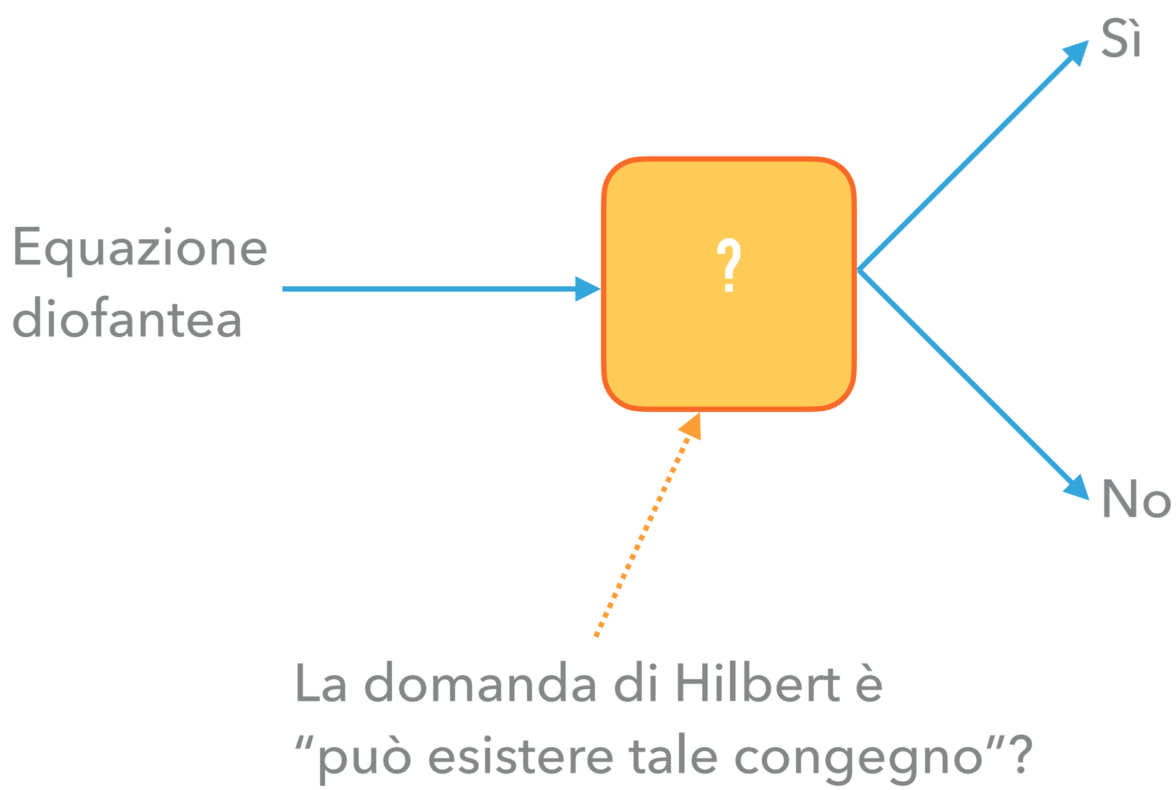
*Given a Diophantine equation with any number of unknown quantities and with rational integral coefficients: To devise a process according to which it can be determined in a finite number of operations whether the equation is solvable in rational integers.*

Traduzione dall'originale in tedesco

Qui è espresso il concetto di avere una procedura che
in un numero finito di passi decide se una data equazione
diofantea possieda o no soluzioni intere

Il problema fu risolto in maniera negativa (non esiste tale procedura) nel 1970 da Yuri Matijasevic, costruendo sul lavoro di Martin Davis, Hilary Putnam e Julia Robinson.

# IL DECIMO PROBLEMA DI HILBERT




- Primo, importante problema:
- Cosa è una procedura che in un numero finito di operazioni può arrivare ad una decisione (sì o no)?
- Alan Turing (1912-1954) lavorò sulla formalizzazione di questo concetto

# LA NASCITA DELLA MACCHINA DI TURING

- Si procede all'idealizzazione del lavoro di un matematico
- Gli "ingredienti" principali sono:

Matematico (x1)

O contabile (x1)

Matita di durata illimitata (x1)


Gomma di durata illimitata (x1)

Carta (x∞)

# LA NASCITA DELLA MACCHINA DI TURING

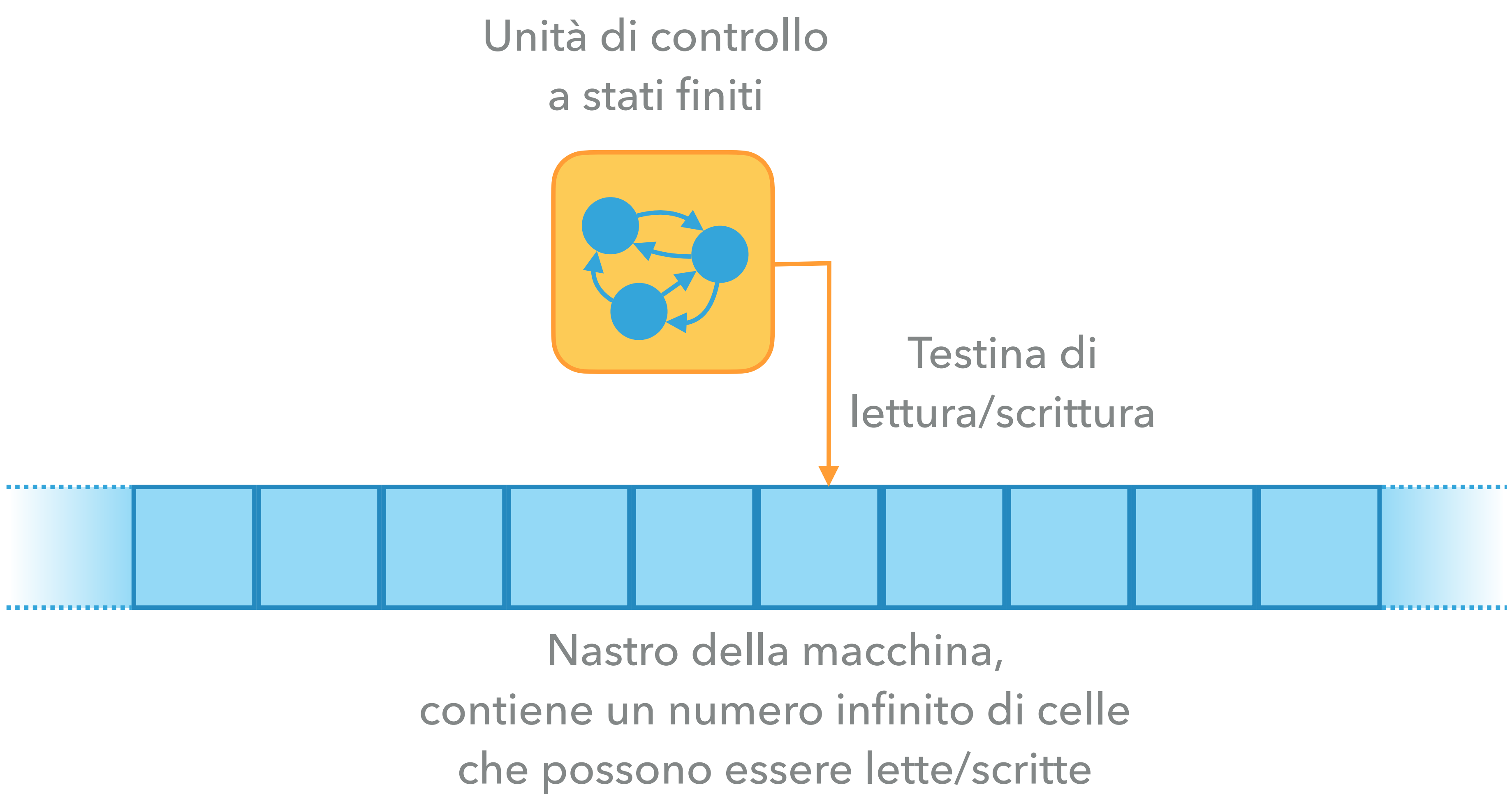
- Un matematico può tenere a mente un numero limitato di informazioni e prendere decisioni in base a quanto legge sul foglio che ha davanti a sé
- Per ovviare alla sua memoria finita può leggere e scrivere da una quantità illimitata di fogli, cancellando e riscrivendo ove necessario
- La quantità di simboli distinti che un matematico può scrivere in un foglio è finita o, almeno, ne può distinguere un numero finito

# LA NASCITA DELLA MACCHINA DI TURING

- Il matematico legge il contenuto del foglio
- Ragiona su come modificarlo cambiando il suo stato mentale (i.e., le informazioni che si ricorda)
- Eventualmente cancella il foglio e cambia quello che c'è scritto sopra
- Passa ad un altro foglio (quale foglio dipende dalle informazioni che si ricorda) e ricomincia la procedura
- A meno che non abbia trovato la risposta che cercava, in quel caso termina

# LA MACCHINA DI TURING

E siamo arrivati alla Macchina di Turing (Turing Machine o TM), che viene rappresenta come:



# DEFINIZIONE FORMALE DI UNA MACCHINA DI TURING

- Una macchina di Turing $M$ è una settupla $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ dove
- $Q$ è un insieme finito di stati
- $\Sigma$ è l'alfabeto di input della macchina (insieme finito di simboli)
- $\Gamma$ è l'alfabeto di lavoro della macchina, con $\Sigma \subseteq \Gamma$ e contenente un simbolo $\# \in \Gamma$ detto simbolo di blank
- $\delta\colon Q \times \Gamma \rightarrow Q \times \Gamma \times \{\leftarrow, \rightarrow\}$ è la funzione di transizione della macchina
- $q_0 \in Q$ è lo stato iniziale della macchina
- $q_{\text{accept}} \in Q$ è lo stato accettante della macchina
- $q_{\text{reject}} \in Q$ è lo stato rifiutante della macchina, con $q_{\text{accept}} \neq q_{\text{reject}}$

# SPIEGAZIONE INTUITIVA DELLA DEFINIZIONE FORMALE

- Una macchina di Turing $M$ è una settupla $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$
- $Q$ è l'insieme finito (per quanto grande) di stati mentali del matematico
- $\Sigma$ è l'insieme di simboli usati per specificare il problema che vogliamo risolvere
- $\Gamma$ è l'insieme di tutti i simboli che possiamo leggere e scrivere, questo include quelli di input e un simbolo "blank" che indica che non è scritto nulla

# SPIEGAZIONE INTUITIVA DELLA DEFINIZIONE FORMALE

- Una macchina di Turing $M$ è una settupla $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$
- $\delta\colon Q \times \Gamma \to Q \times \Gamma \times \{\leftarrow, \rightarrow\}$ ci dice come "ragiona" il matematico:
  - A partire dal suo stato mentale (un elemento di $Q$) e da quello che legge sul foglio che ha davanti (un elemento di $\Gamma$)
  - Decide che nuovo stato mentale assumere (un elemento di $Q$), cosa scrivere sul foglio che ha davanti (un elemento di $\Gamma$) e se passare al il foglio alla sua sinistra ($\leftarrow$) o alla sua destra ($\rightarrow$)

# SPIEGAZIONE INTUITIVA DELLA DEFINIZIONE FORMALE

La funzione di transizione è spesso vista come una tabella di transizione

| Stato | Simbolo | Nuovo stato | Nuovo simbolo | Movimento |
|---|---|---|---|---|
| $q_0$ | $A$ | $q_0$ | $B$ | $\rightarrow$ |
| $q_0$ | $B$ | $q_{\text{accept}}$ | $B$ | $\leftarrow$ |
| $q_1$ | $A$ | $q_{\text{reject}}$ | $A$ | $\leftarrow$ |
| $q_1$ | $B$ | $q_1$ | $A$ | $\rightarrow$ |

Condizione attuale

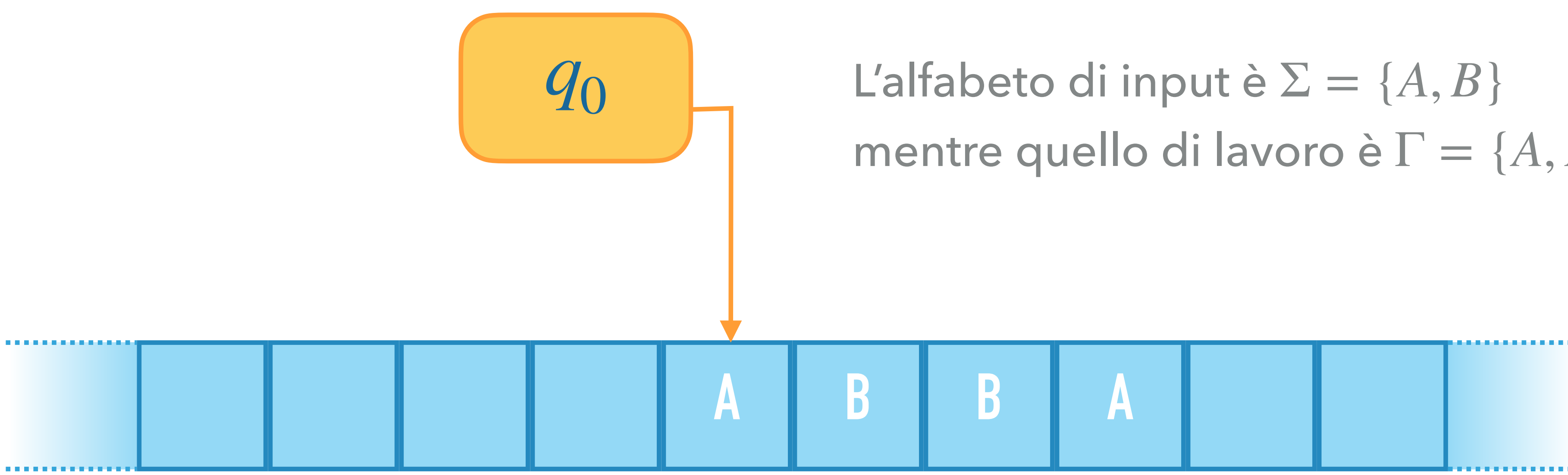
Azione da fare

# SPIEGAZIONE INTUITIVA DELLA DEFINIZIONE FORMALE

- Una macchina di Turing $M$ è una settupla $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$
- $q_0$ è lo stato mentale iniziale del matematico
- $q_{\text{accept}}$ è lo stato finale di "la risposta è positiva"
- $q_{\text{reject}}$ è lo stato finale di "la risposta è negativa"

# LA MACCHINA DI TURING: PASSI DI COMPUTAZIONE

Proviamo a definire una TM che accetta se trova scritto sul nastro "ABBA", non importa se seguito da altro, e rifiuta altrimenti

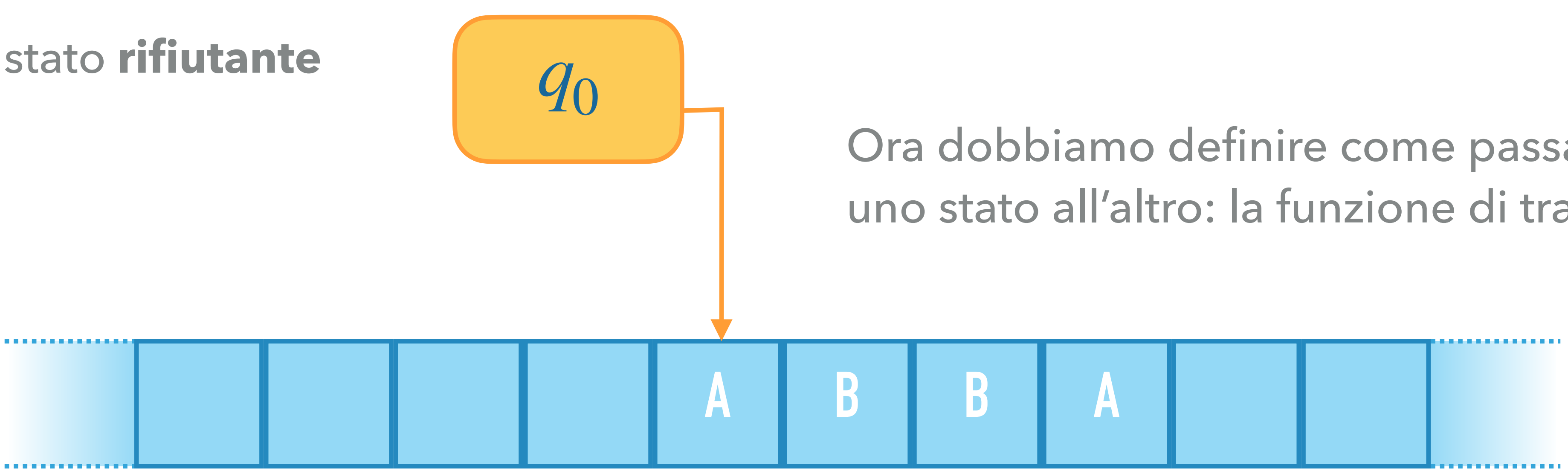



L'alfabeto di input è $\Sigma = \{A, B\}$
mentre quello di lavoro è $\Gamma = \{A, B, \#\}$

# LA MACCHINA DI TURING: PASSI DI COMPUTAZIONE

Stati della macchina:

- $q_0$ stato **iniziale**, significa "devo ancora iniziare a leggere"
- $q_A$ ha come semantica "ho letto la lettera A"
- $q_{AB}$ ha come semantica "ho letto le lettere AB"
- $q_{ABB}$ ha come semantica "ho letto le lettere ABB"
- $q_{ABBA}$ ha come semantica "ho letto le lettere ABBA", questo è lo stato **accettante**
- $q_{\text{reject}}$ è lo stato **rifiutante**

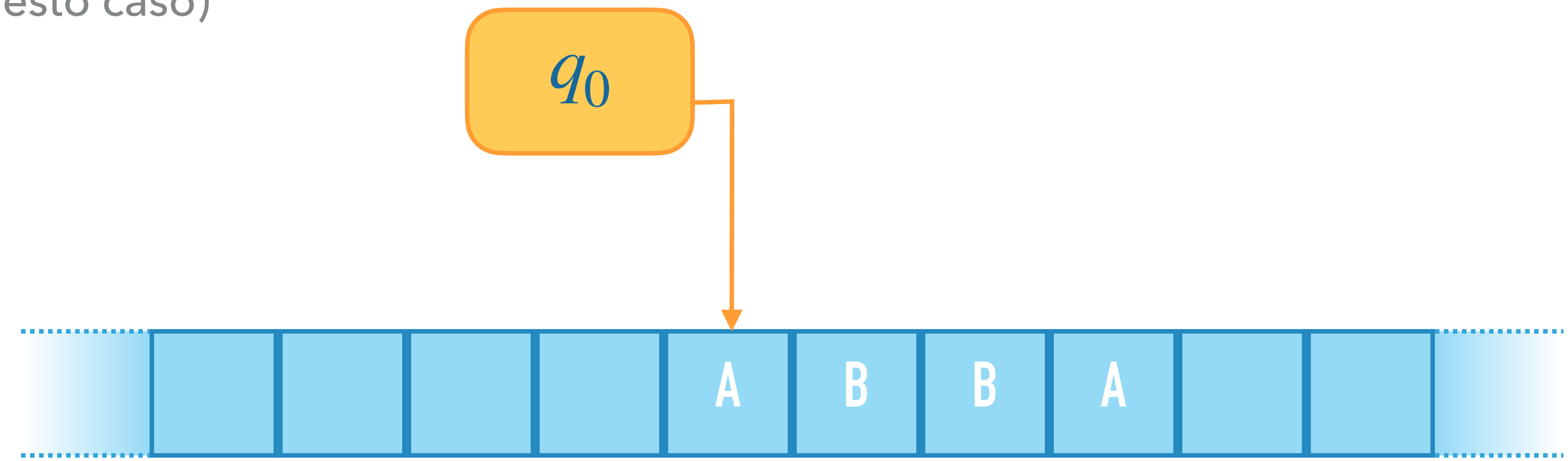



Ora dobbiamo definire come passare da uno stato all'altro: la funzione di transizione

# LA MACCHINA DI TURING: PASSI DI COMPUTAZIONE

Da $q_0$ abbiamo due possibilità:

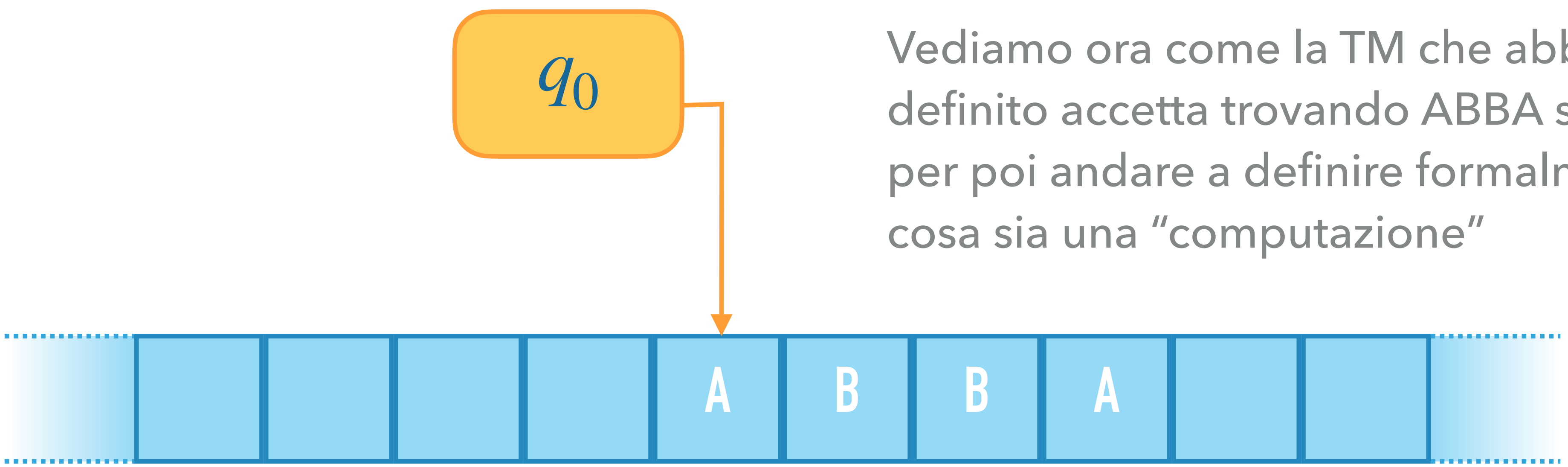
- Leggiamo "A" sotto la testina, quindi andiamo nello stato $q_A$ e andiamo a leggere la casella a destra
- Leggiamo qualcosa di diverso da "A" sotto la testina, non c'è scritto "ABBA", quindi rifiutiamo
- In termini di regola di transizione abbiamo:
  - $\delta(q_0, A) = (q_A, A, \rightarrow)$ (riscriviamo lo stesso simbolo che abbiamo letto, ma non è importante)
  - $\delta(q_0, B) = \delta(q_0, \#) = (q_{\text{reject}}, \#, \rightarrow)$ (anche se cosa scriviamo e dove ci spostiamo non è importante in questo caso)



# LA MACCHINA DI TURING: PASSI DI COMPUTAZIONE

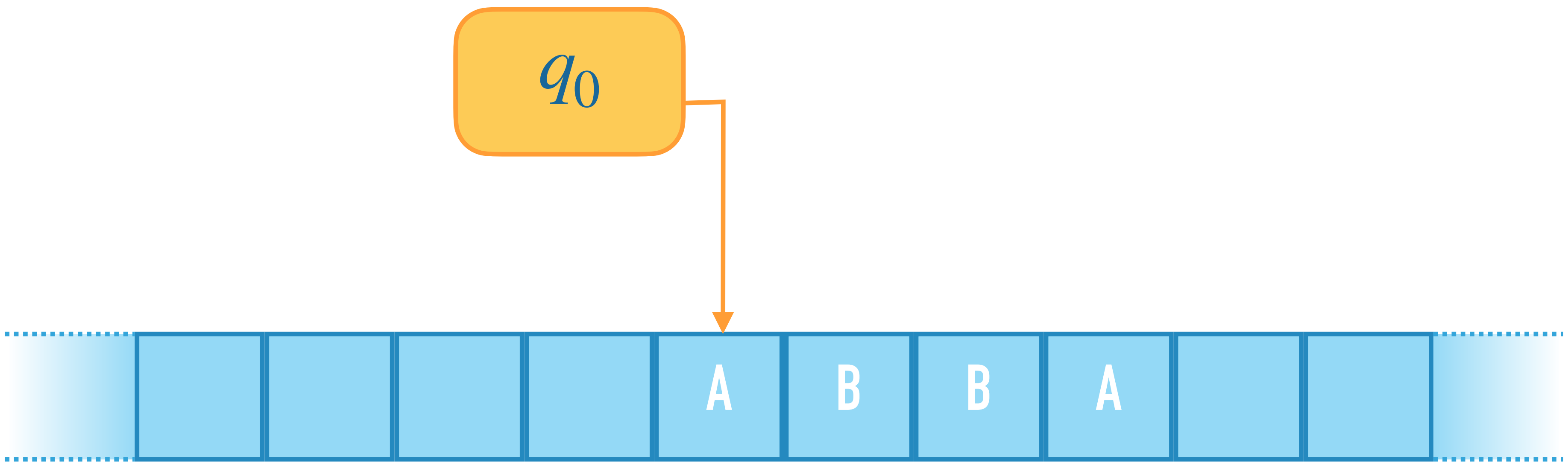
Seguendo lo stesso ragionamento possiamo definire le seguenti transizioni:

- $\delta(q_A, B) = (q_{A,B}, B, \rightarrow)$
- $\delta(q_{AB}, B) = (q_{ABB}, B, \rightarrow)$
- $\delta(q_{ABB}, A) = (q_{ABBA}, A, \rightarrow)$
- Tutte le altre transizioni portano ad uno stato rifiutante



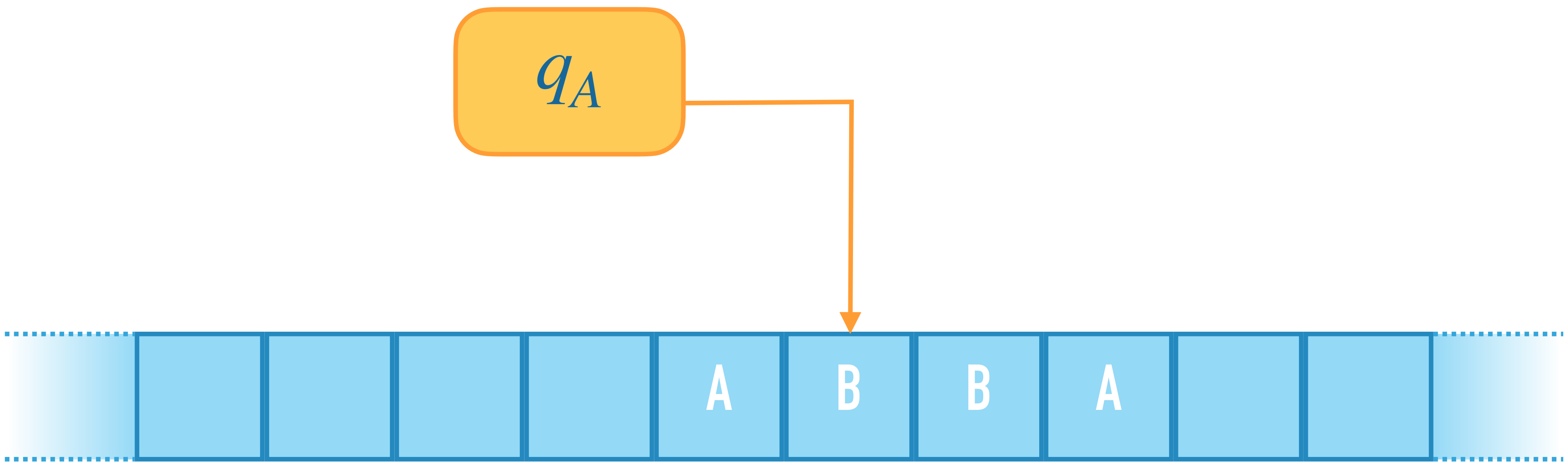

Vediamo ora come la TM che abbiamo definito accetta trovando ABBA sul nastro, per poi andare a definire formalmente cosa sia una "computazione"

# LA MACCHINA DI TURING: PASSI DI COMPUTAZIONE



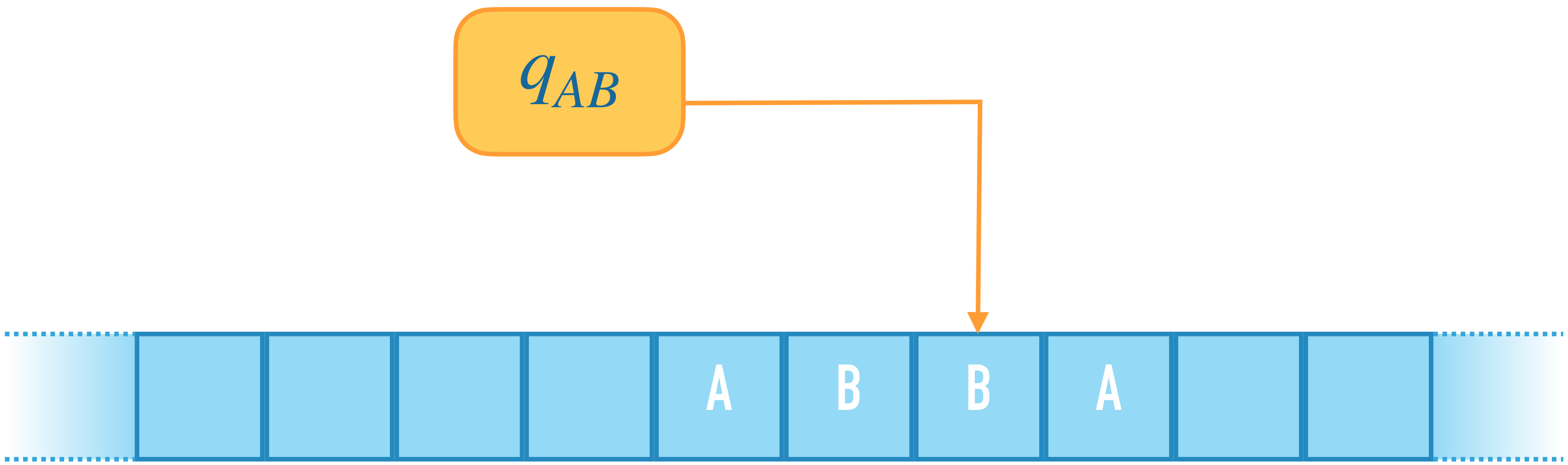

Transizione da eseguire: $\delta(q_0, A) = (q_A, A, \rightarrow)$

# LA MACCHINA DI TURING: PASSI DI COMPUTAZIONE



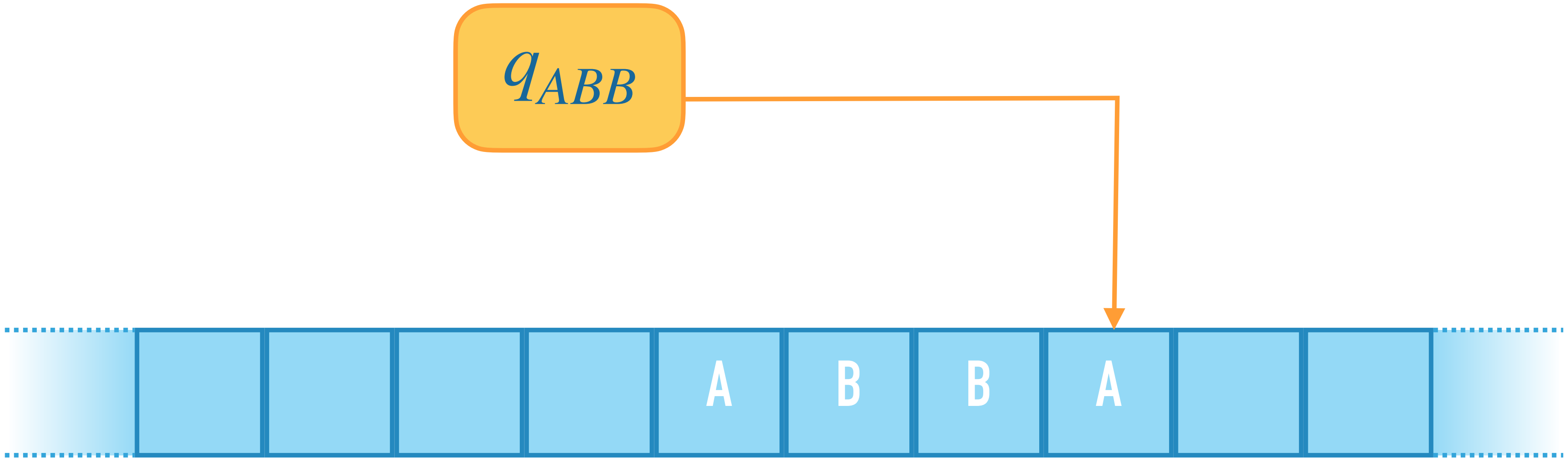

Transizione da eseguire: $\delta(q_A, B) = (q_{AB}, B, \rightarrow)$

# LA MACCHINA DI TURING: PASSI DI COMPUTAZIONE



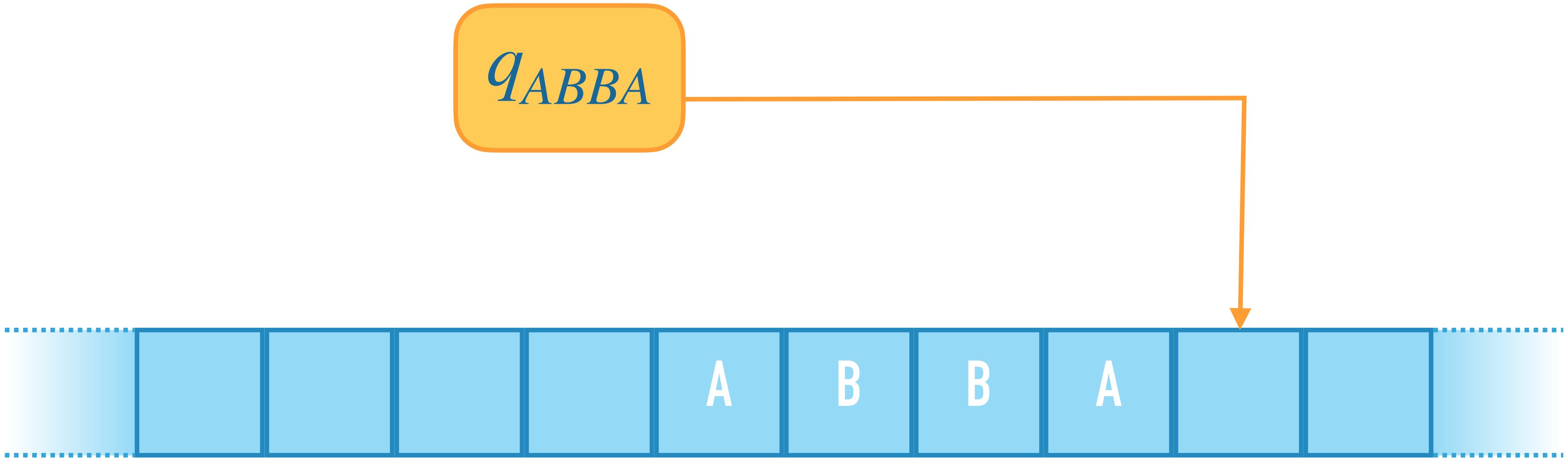

Transizione da eseguire: $\delta(q_{AB}, B) = (q_{ABB}, B, \rightarrow)$

# LA MACCHINA DI TURING: PASSI DI COMPUTAZIONE




Transizione da eseguire: $\delta(q_{ABB}, A) = (q_{ABBA}, A, \rightarrow)$

# LA MACCHINA DI TURING: PASSI DI COMPUTAZIONE




La macchina è ora nello stato accettante, quindi abbiamo accettato quello che c'era scritto inizialmente sul nastro, ovvero l'input

# ALFABETI, PAROLE E LINGUAGGI

- Un alfabeto $\Sigma$ è un insieme finito di simboli. Per esempio $\Sigma = \{a, b, c\}$
- Una parola $w$ è una sequenza di simboli da $\Sigma$, per esempio $w = abba$ è una parola sull'alfabeto $\Sigma = \{a, b, c\}$
- Indicheremo con $\varepsilon$ la stringa vuota, che non contiene alcun elemento
- La lunghezza di una parola $w$, denotata con $|w|$, è il numero di simboli che $w$ contiene. Per esempio per $w = abba$, $|w| = 4$

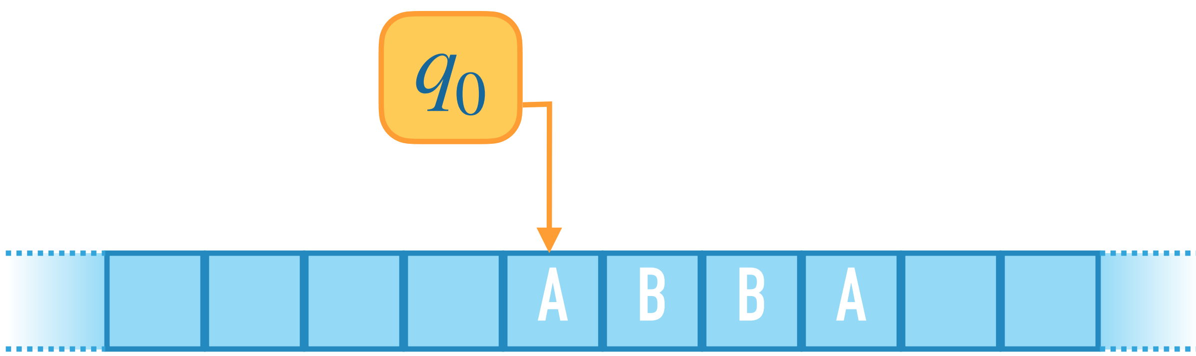
# ALFABETI, PAROLE E LINGUAGGI

- La concatenazione di due parole $x$ e $y$ è la parola $xy$ contenente tutti i simboli di $x$ seguiti da tutti i simboli di $y$. Per esempio, se $x = ab$ e $y = ba$ allora $xy = abba$
- Indichiamo con $\Sigma^\star$ (letto "sigma star"), l'insieme $\{x_1 \cdots x_k \mid k \geq 0 \text{ and each } x_i \in \Sigma\}$ di tutte le stringhe finite sull'alfabeto $\Sigma$. Per esempio se $\Sigma = \{a, b, c\}$, allora $\Sigma^\star = \{\varepsilon, a, b, c, aa, ab, ac, ba, \ldots\}$
- Dato un alfabeto $\Sigma$, un **linguaggio** $L$ su $\Sigma$ è un sottoinsieme di $\Sigma^\star$, i.e., $L \subseteq \Sigma^\star$

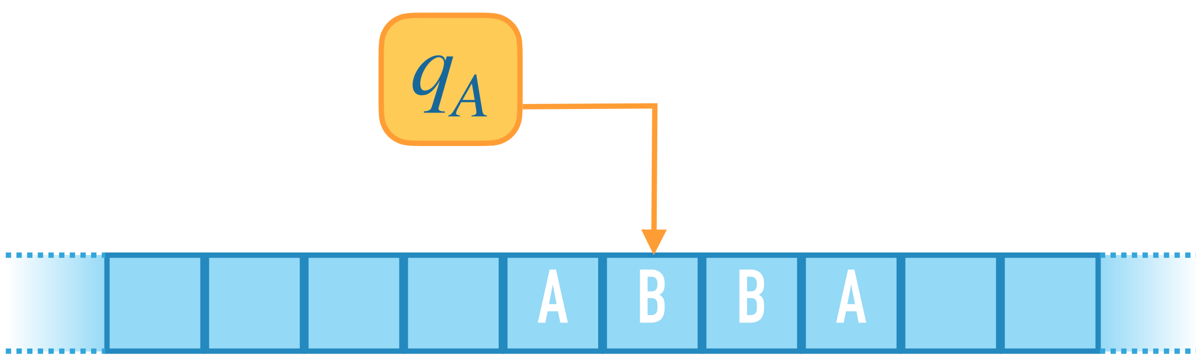
# CONFIGURAZIONE DI UNA MACCHINA DI TURING

- Vogliamo un modo di descrivere in che configurazione è una macchina di Turing:
  - Contenuto del nastro
  - Stato della macchina
  - Posizione della testina
- Tutto questo è descrivibile con una concatenazione di tre parole: $uqv$, dove:
  - $q \in Q$ è lo stato della TM
  - $u \in \Gamma^\star$ è il contenuto del nastro che precede la testina, con solo simboli di blank prima del primo simbolo di $u$
  - $v \in \Gamma^\star$ è il contenuto del nastro sotto la testina (che si trova sopra il primo simbolo di $v$) e che è successivo alla testina. Vi sono solo simboli di blank dopo l'ultimo simbolo di $v$

# LA MACCHINA DI TURING: CONFIGURAZIONE




Configurazione: $q_0ABBA$




Configurazione: $Aq_0BBA$

Ma con che configurazione inizia una macchina di Turing?

La configurazione iniziale di una TM con input $w \in \Sigma^\star$ è un nastro in cui è presente solo $w$ e simboli di blank su tutto il resto del nastro, la testina si trova sul primo simbolo di $w$ e la macchina si trova nel suo stato iniziale $q_0$. Quindi la configurazione iniziale è $q_0w$

# PASSI DI COMPUTAZIONE

- Data due configurazioni $C_1$ e $C_2$, diciamo che $C_1$ porta a $C_2$ se possiamo ottenere $C_2$ da $C_1$ in un singolo passo di computazione.
- Formalmente: se $C_1 = uaq_ibv$ con $u, v \in \Gamma^\star$, $a, b \in \Gamma$ e $q_i \in Q$ allora
  - $C_1$ porta a $C_2 = uq_jacv$ se esiste $\delta(q_i, b) = (q_j, c, \leftarrow)$
  - $C_1$ porta a $C_2 = uacq_jv$ se esiste $\delta(q_i, b) = (q_j, c, \rightarrow)$

# CONFIGURAZIONI ACCETTANTI E RIFIUTANTI

- Una configurazione che contiene lo stato $q_{\text{accept}}$ è detta accettante
- Una configurazione che contiene lo stato $q_{\text{reject}}$ è detta rifiutante
- Configurazione accettanti e rifiutanti sono dette configurazioni di arresto, dato che quando la macchina entra in uno stato di accettazione o di rifiuto assumiamo che si fermi

# COMPUTAZIONE

- Una computazione è una sequenza di configurazioni $C_1, C_2, \ldots$ tali per cui $C_i$ porta a $C_{i+1}$
- Una computazione $C_1, C_2, \ldots, C_k$ è detta accettante se la configurazione $C_k$ è accettante
- Una computazione $C_1, C_2, \ldots, C_k$ è detta rifiutante se la configurazione $C_k$ è rifiutante
- Una computazione si arresta se la sequenza $C_1, C_2, \ldots$ è finita e, quindi, raggiunge uno stato accettante o rifiutante $C_k$

# COSA PUÒ FARE QUINDI UNA MACCHINA DI TURING?

- Su un input $w \in \Sigma^\star$ una macchina di Turing può:
  - Accettare
  - Rifiutare
  - Non fermarsi. Questo può accadere se non raggiungiamo mai uno stato accettante o rifiutante.
- Possiamo usare le macchine di Turing per verificare se una parola appartiene o no ad un linguaggio

# LINGUAGGI RICORSIVAMENTE ENUMERABILI

- L'insieme delle parole di $\Sigma^{\star}$ che una macchina di Turing $M$ accetta è detto il **linguaggio di** $M$, o il **linguaggio riconosciuto da** $M$, denotato con $L(M)$
- Se un linguaggio $L$ è riconosciuto da una macchina di Turing allora diciamo che $L$ è un **linguaggio ricorsivamente enumerabile**. La classe dei linguaggi ricorsivamente enumerabili si denota con $RE$
- Notate come non sia richiesto che $M$ si fermi per $w \notin L$

# LINGUAGGI RICORSIVI

- Dato che spesso non possiamo riconoscere (vedremo meglio dopo) se una macchina si arresta o no, possiamo richiedere che la macchina si arresti sempre. Quindi $w \notin L(M)$ implica che $M$ si arresta rifiutando su input $w$
- In questo caso diciamo che il linguaggio è **deciso** da $M$
- Se un linguaggio $L$ è deciso da una macchina di Turing allora diciamo che $L$ è un **linguaggio ricorsivo**. La classe dei linguaggi ricorsivi si denota con $R$

# MA LA TM È UN "BUON MODELLO"?

- I linguaggi riconosciuti/decisi da TM sono gli stessi riconosciuti/decisi da tantissimi modelli di calcolo: macchine a registri, lambda calcolo, sistemi a membrane, funzioni ricorsive, etc.
- Praticamente ogni modello di calcolo vagamente realistico ideato dagli anni '30 ad oggi ha al più la stessa potenza della macchina di Turing
- Questo fa pensare che il modello di macchina di Turing sia una buona rappresentazione della nozione informale di algoritmo

# TESI DI CHURCH-TURING

La classe delle funzioni (intuitivamente) calcolabili coincide con la classe delle funzioni calcolabili da macchine di Turing

La tesi prende il nome da Alan Turing e Alonso Church, inventore del lambda calcolo

Non è un teorema, perché c'è la definizione di "funzioni (intuitivamente) calcolabili" non è una nozione formalizzata

# ESISTONO LINGUAGGI NON DECISI DA MACCHINE DI TURING?

- Ci stiamo chiedendo se esistono linguaggi al di fuori di $R$
- Mostriamo che esiste un linguaggio non ricorsivo
- Definiamo $L = \{\langle M, w \rangle \mid M$ si arresta su input $w\}$ come il linguaggio delle coppie formate da una macchina di Turing $M$ e da un input $w$ tali per cui $M$ si arresta su input $w$
- Dobbiamo prima assicurarci che ogni coppia $\langle M, w \rangle$ sia effettivamente rappresentabile come una stringa su un dato alfabeto
- Distinguiamo $M, w$ (la macchina $M$ e la parola $w$) da $\langle M, w \rangle$, la descrizione (una codifica) di $M$ e di $w$

# CODIFICA DELLE MACCHINE DI TURING

Mostriamo una possibile codifica (molte sono possibili):

Insieme degli interi da $0$ a $|Q| - 1$

Tabella di transizione scritta come sequenza di quintuple di numeri

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$$

Numero che rappresenta lo stato

Insieme degli interi da $|Q|$ a $|\Gamma| + |Q| - 1$

Insieme degli interi da $|Q|$ a $|\Sigma| + |Q| - 1$

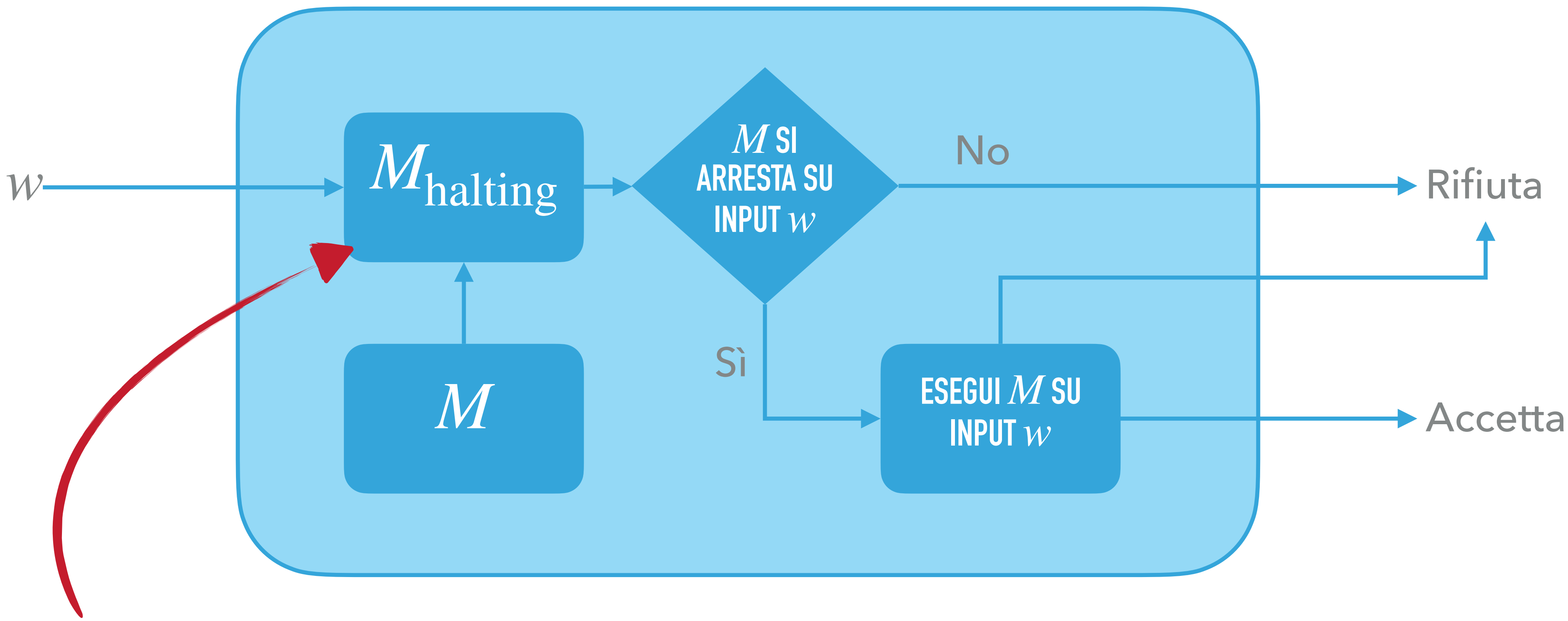
Abbiamo quindi codificato un TM come i numeri $|Q|, |\Sigma|, |\Gamma|,$ seguiti da una sequenza di quintuple di numeri, seguite da tre numeri: stiamo usando solo un insieme finito di simboli (le cifre usate per i numeri ed i segni di punteggiatura – se volessimo potremmo addirittura codificare tutto in binario e usare solo due simboli)

# IL PROBLEMA DELL'ARRESTO

- Se il linguaggio $L = \{\langle M, w\rangle \mid M \text{ si arresta su input } w\}$ fosse decidibile allora esisterebbe una macchina $M_{\text{halting}}$ che ci dice se una macchina $M$ si arresta o no su un dato input $w$
- Questo è il **problema dell'arresto**, decidere se una macchina si arresta o no
- Questo mostrerebbe che $R = RE$, dato che per ogni macchina $M$ che riconosce un linguaggio $L(M) \in RE$ verificare con $M_{\text{halting}}$ che $M$ termina su input $w$, in quel caso eseguiamo $M$ su input $w$, altrimenti rifiutiamo subito

# IL PROBLEMA DELL'ARRESTO

Come sarebbe possibile trasformare una macchina $M$ che riconosce un linguaggio in una macchina $M'$ che decide lo stesso linguaggio, arrestandosi sempre



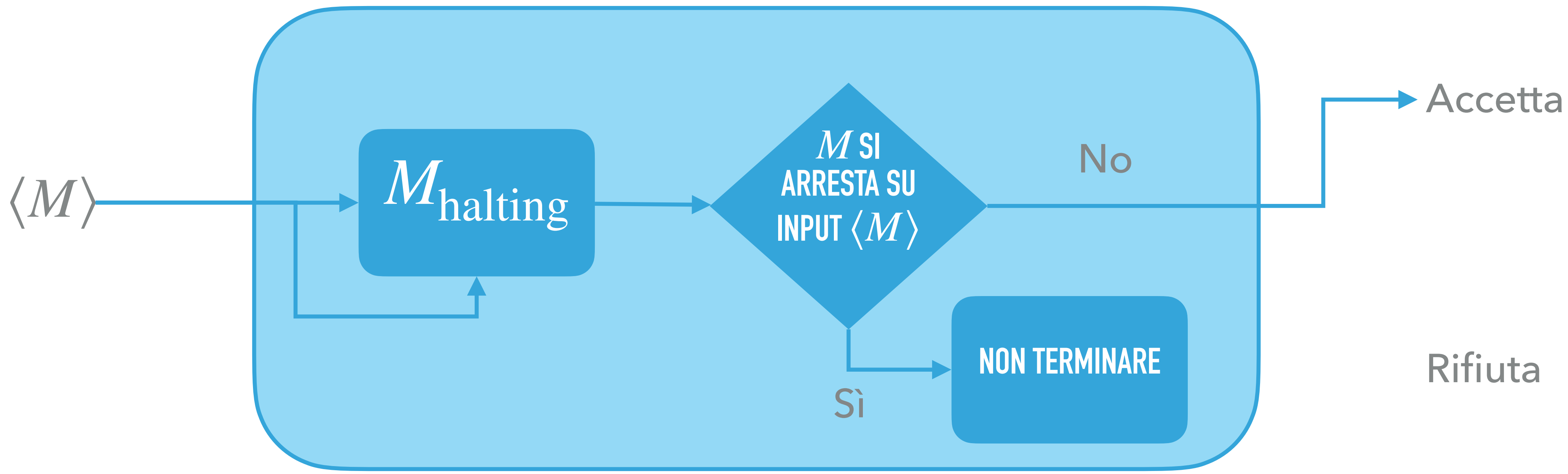

Mostriamo che $M_{\text{halting}}$ non esiste

# IL PROBLEMA DELL'ARRESTO

- Per mostrare che $L = \{\langle M, w\rangle \mid M \text{ si arresta su input } w\}$ non è decidibile, assumiamo esista la macchina $M_{\text{halting}}$ tale per cui $L(M_{\text{halting}}) = L$
- Allora possiamo costruire una macchina $D$ definita in modo tale che su input $\langle M\rangle$ (la descrizione di una TM):
  - Se $M_{\text{halting}}$ accetta su input $\langle M, \langle M\rangle\rangle$allora $D$ non termina
  - Se $M_{\text{halting}}$ rifiuta su input $\langle M, \langle M\rangle\rangle$ allora $D$ accetta
- Notate che stiamo fornendo in input a una macchina la sua stessa descrizione, che è perfettamente valido dato che la descrizione $\langle M\rangle$ di $M$ è una parola (potete pensarlo come una stringa binaria)

# IL PROBLEMA DELL'ARRESTO

Come costruire la macchina $D$ a partire dalla macchina $M_{\text{halting}}$




Questo mostra che se la macchina $M_{\text{halting}}$ esiste allora possiamo costruire la macchina $D$

Ma se la macchina $D$ non esiste allora neanche la macchina $M_{\text{halting}}$ può esistere

# IL PROBLEMA DELL'ARRESTO

- Ci chiediamo cosa succede quando diamo in input a $D$ la sua stessa codifica $\langle D \rangle$. Abbiamo solo due casi:

- Se $D$ su input $\langle D \rangle$ accetta allora $M_{\text{halting}}$ su input $\langle D, \langle D \rangle \rangle$ ha rifiutato, quindi $D$ su input $\langle D \rangle$ non si arresta. Una contraddizione

- Se $D$ su input $\langle D \rangle$ non si arresta allora $M_{\text{halting}}$ su input $\langle D, \langle D \rangle \rangle$ ha accettato, quindi $D$ su input $\langle D \rangle$ si arresta. Una contraddizione

# IL PROBLEMA DELL'ARRESTO

- La macchina $D$ non può esistere, ma quindi è la macchina $M_{\text{halting}}$ che non può esistere
- Esistono quindi linguaggi non ricorsivi, come $L$, che è un linguaggio **indecidibile** o **non decidibile**.
- Il linguaggio $L$ è invece ricorsivamente enumerabile, dato che è sufficiente simulare la macchina $M$ su input $w$ e accettare se la macchina accetta, rifiutare se rifiuta e continuare la simulazione di $M$ senza arrestarsi se $M$ non si arresta, ne segue che $R \subsetneq RE$

# IL PROBLEMA DELL'ARRESTO: IMPLICAZIONI

- Dato che i linguaggi di programmazione come Python sono Turing-completi (i.e., sono potenti come una macchina di Turing), il risultato negativo del problema dell'arresto si applica a loro
- Ne segue che non esiste nessun programma in grado di dirci con certezza se il nostro codice Python entra in un ciclo infinito oppure no (esistono euristiche, ma programmi del genere non possono essere infallibili)