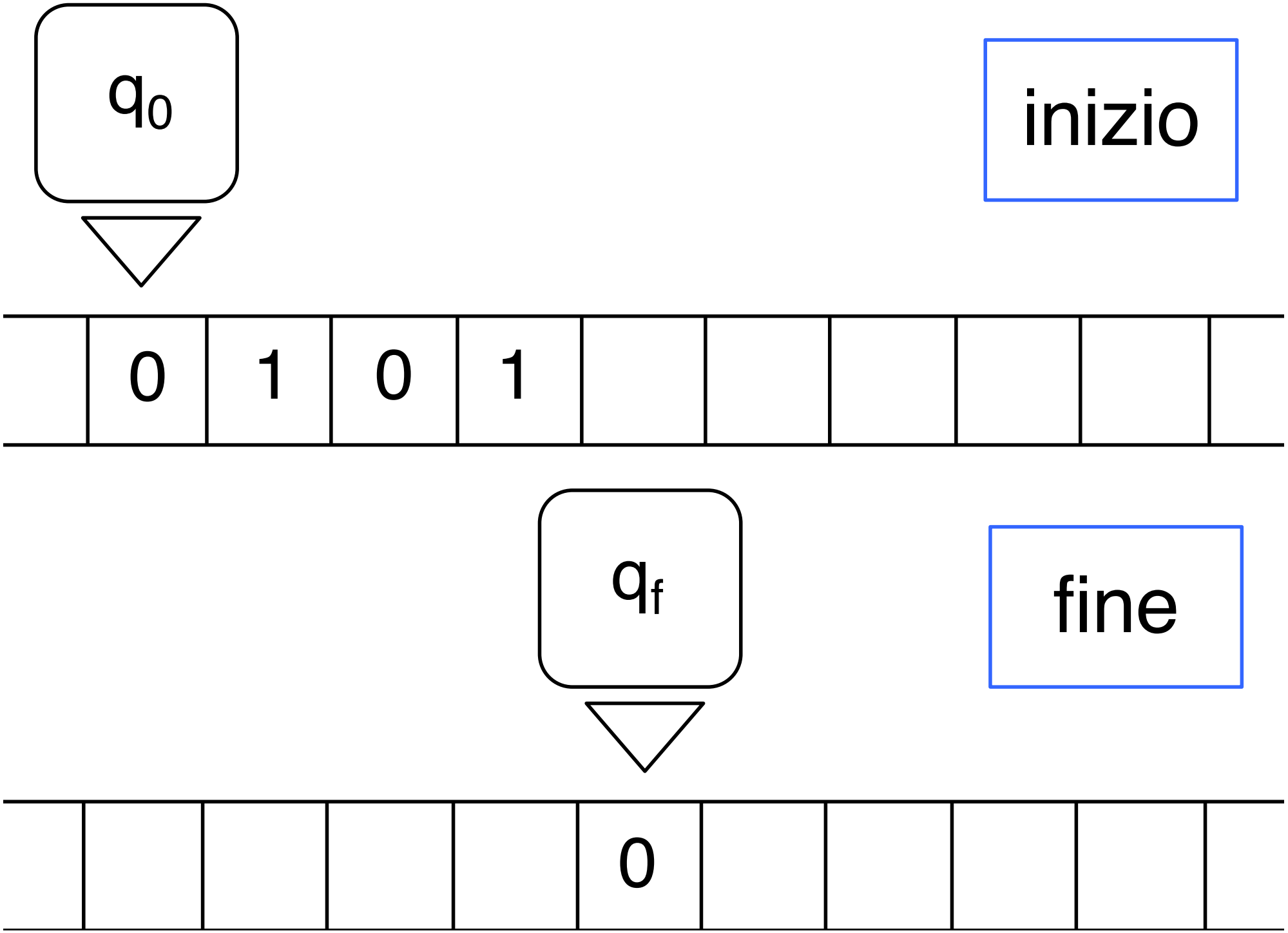
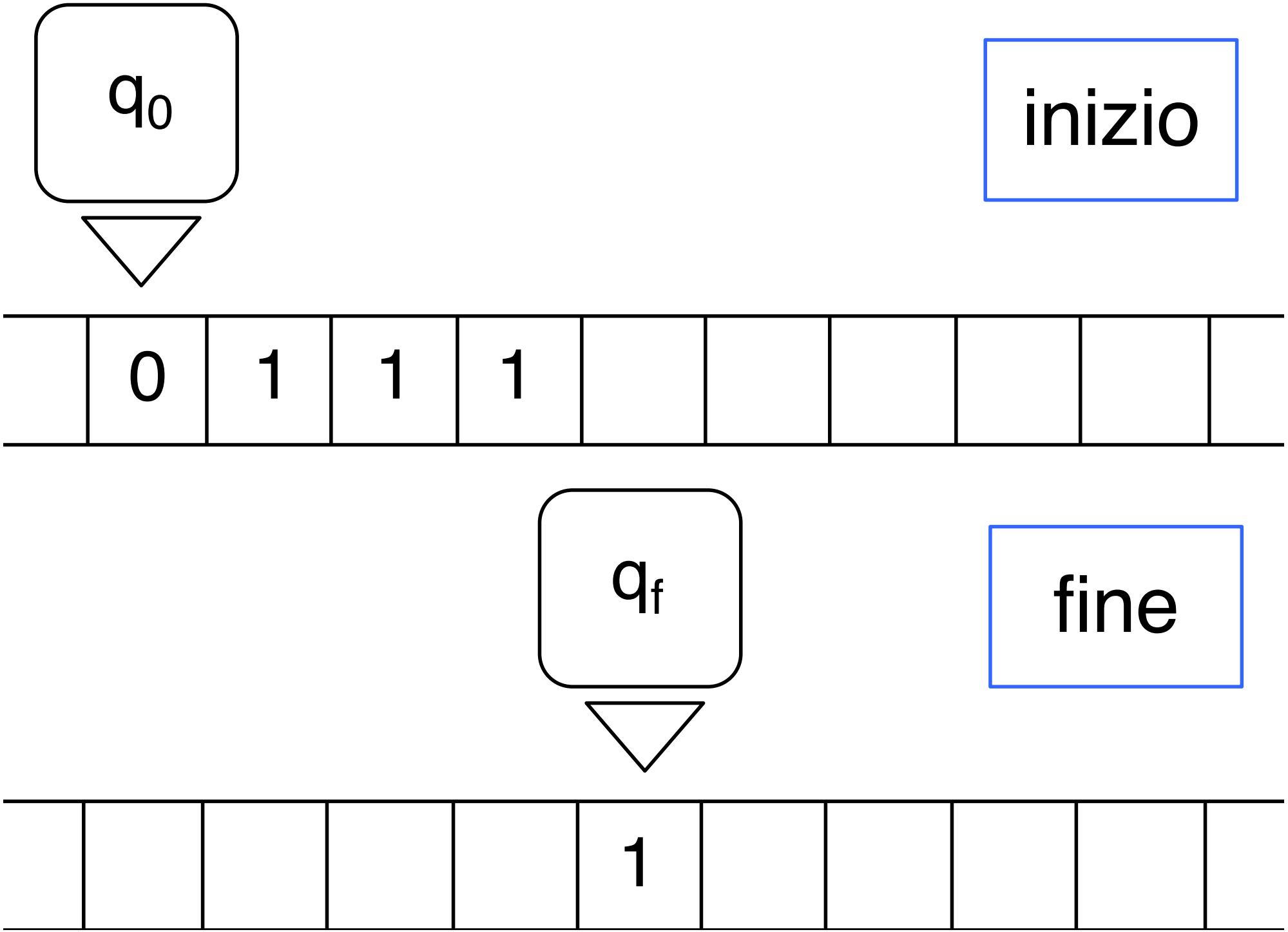
Computabilità
Appunti_08 (19/12/2022)

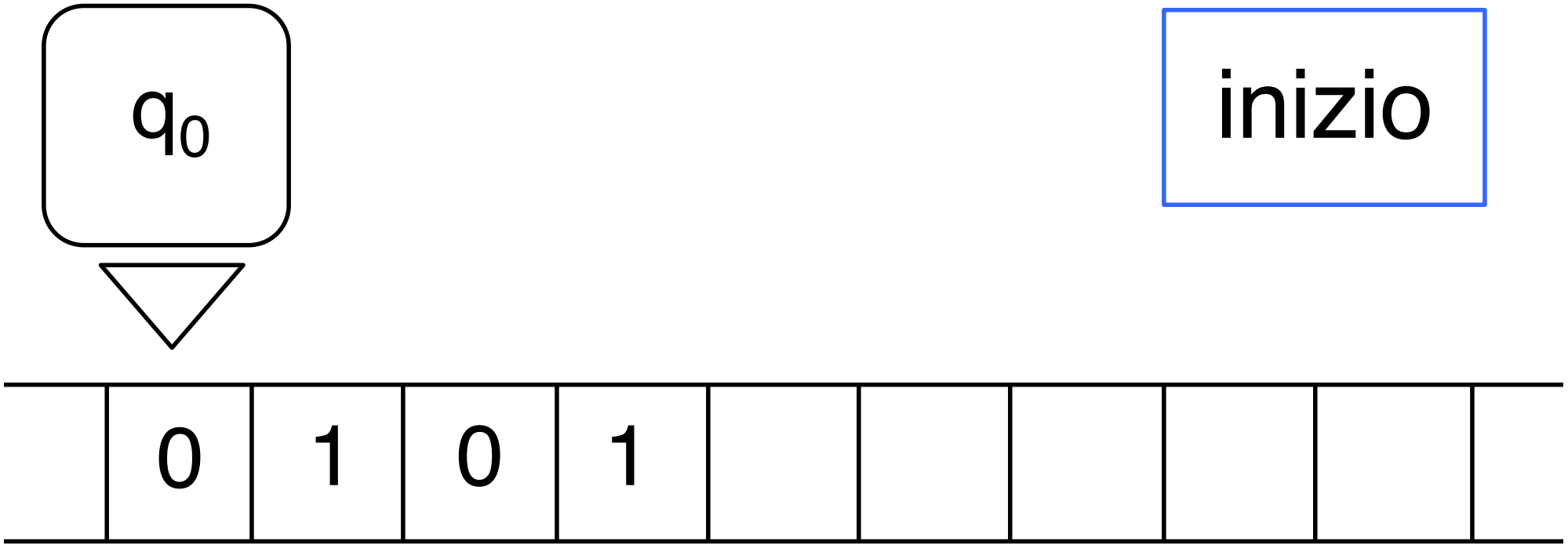
Mario Verdicchio
Università degli Studi di Bergamo
Anno Accademico 2022-2023

Esercizio

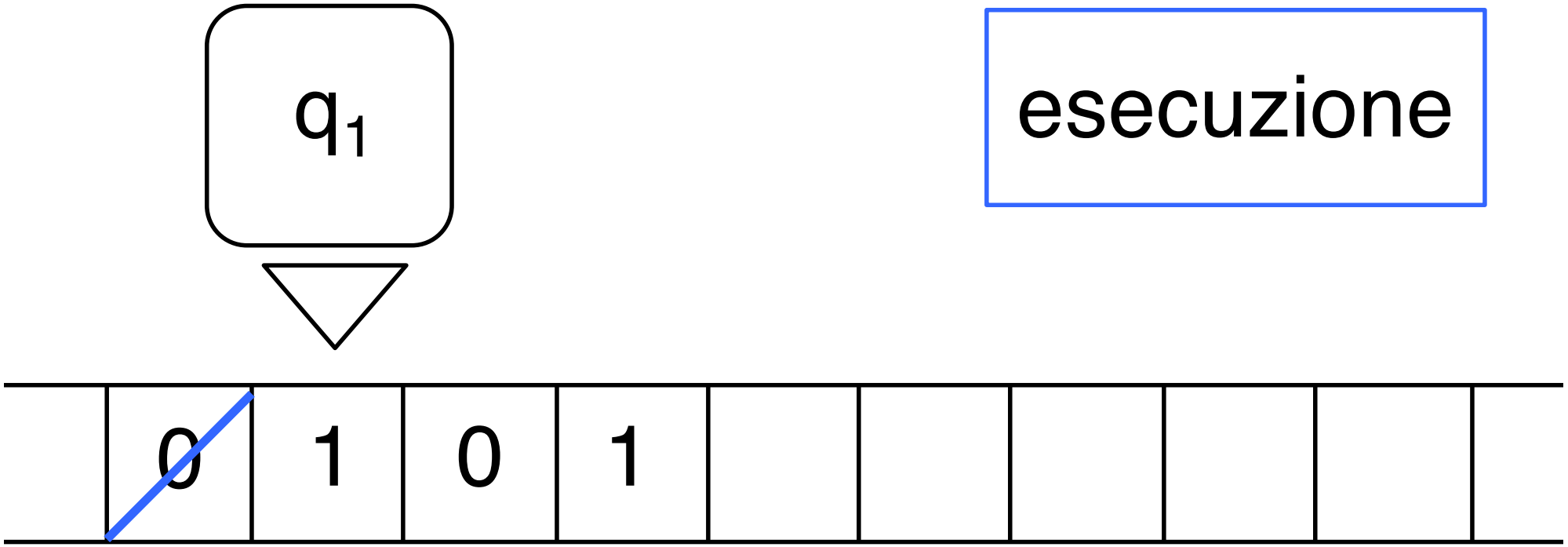
- Creare una macchina di Turing (MT) che accetti il seguente linguaggio:
  - $(01)^i$ (con $i > 0$)


q_0
inizio
0
1
0
1
q_f
fine
0

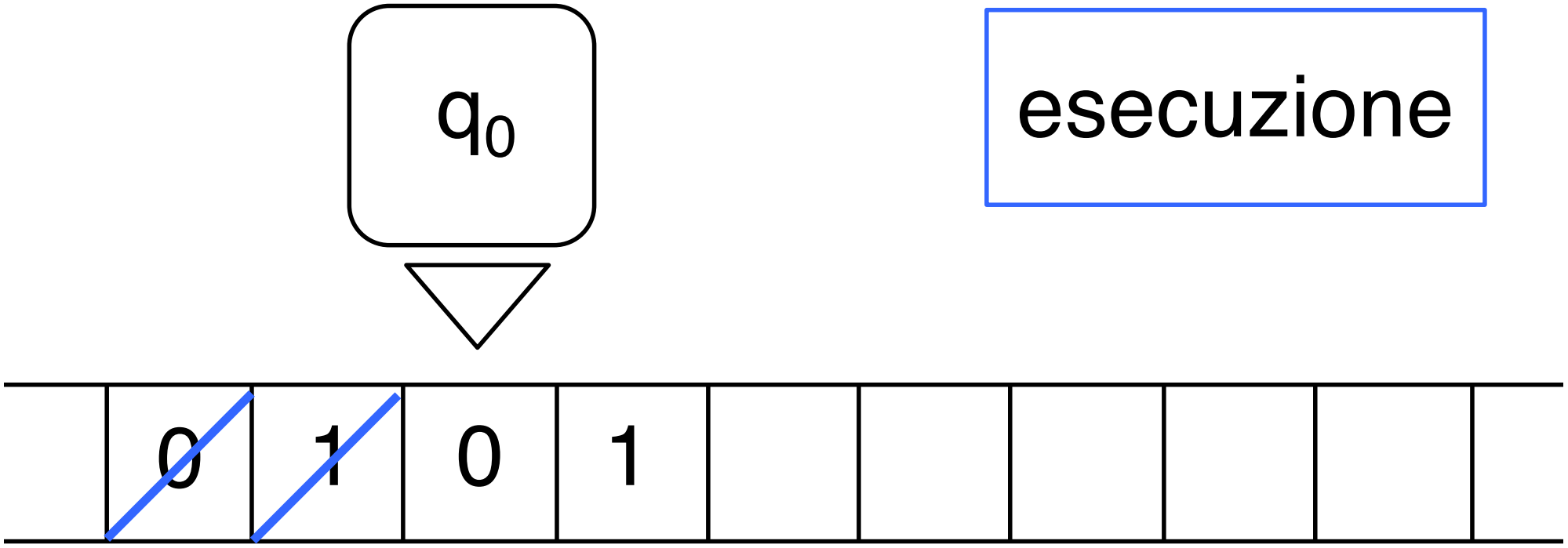

q_0
inizio
0
1
1
1
q_f
fine
1



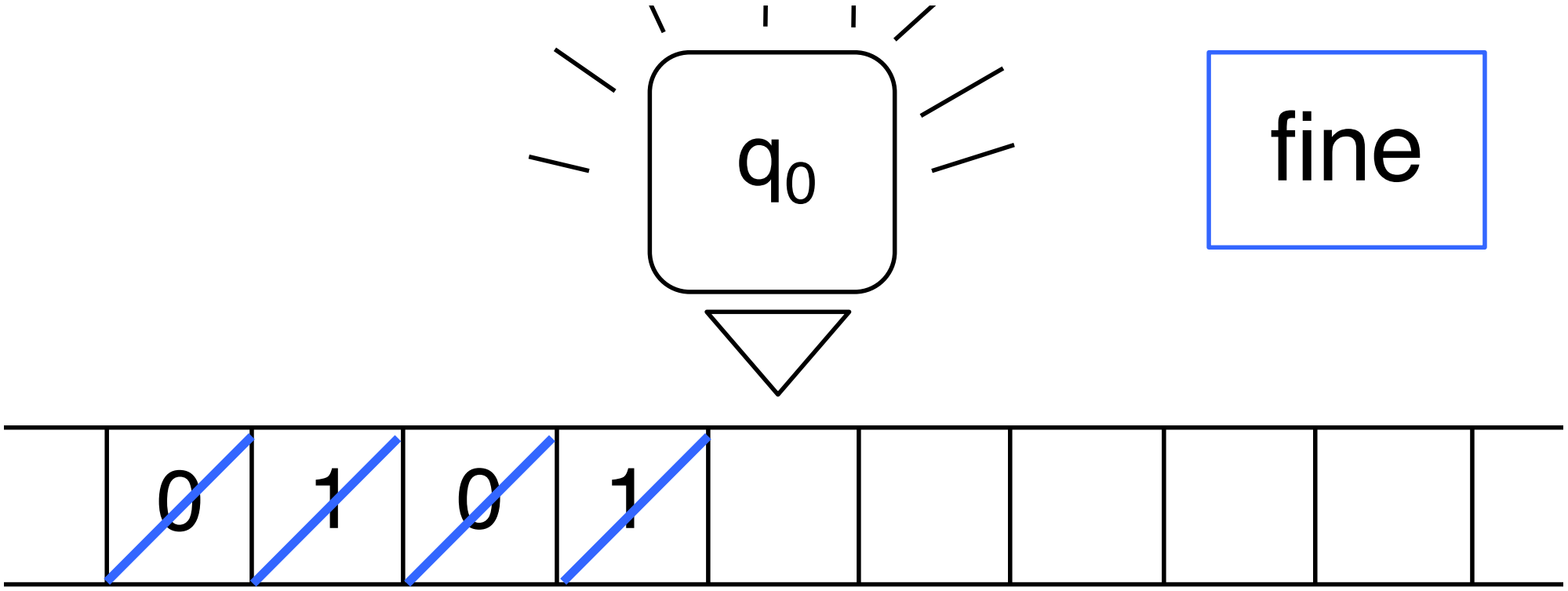
- la MT accetta la stringa presente sul nastro se è una sequenza di 0 e 1 alternati, che inizia con uno 0
- visto che l'input deve essere cancellato, ogni volta che la MT legge un simbolo, lo cancella e va a destra



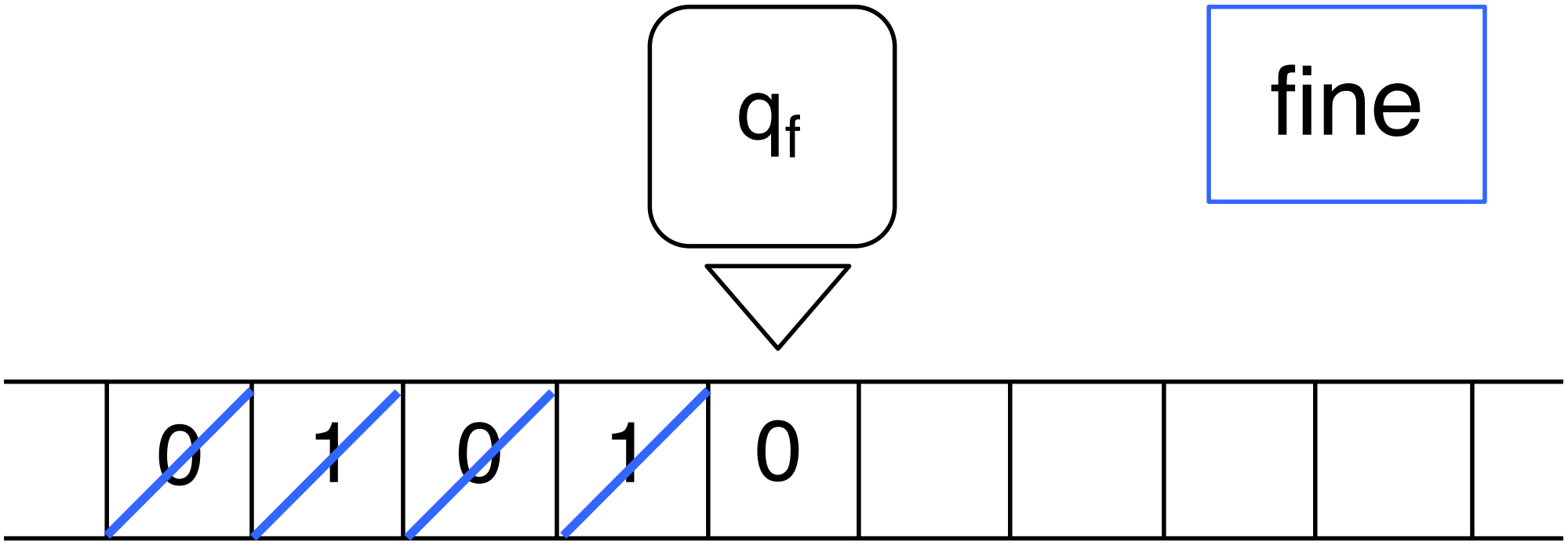
- cancellato il primo 0, la MT si aspetta, secondo la grammatica del linguaggio che accetta, di trovare un 1
- entra quindi nello stato q_1



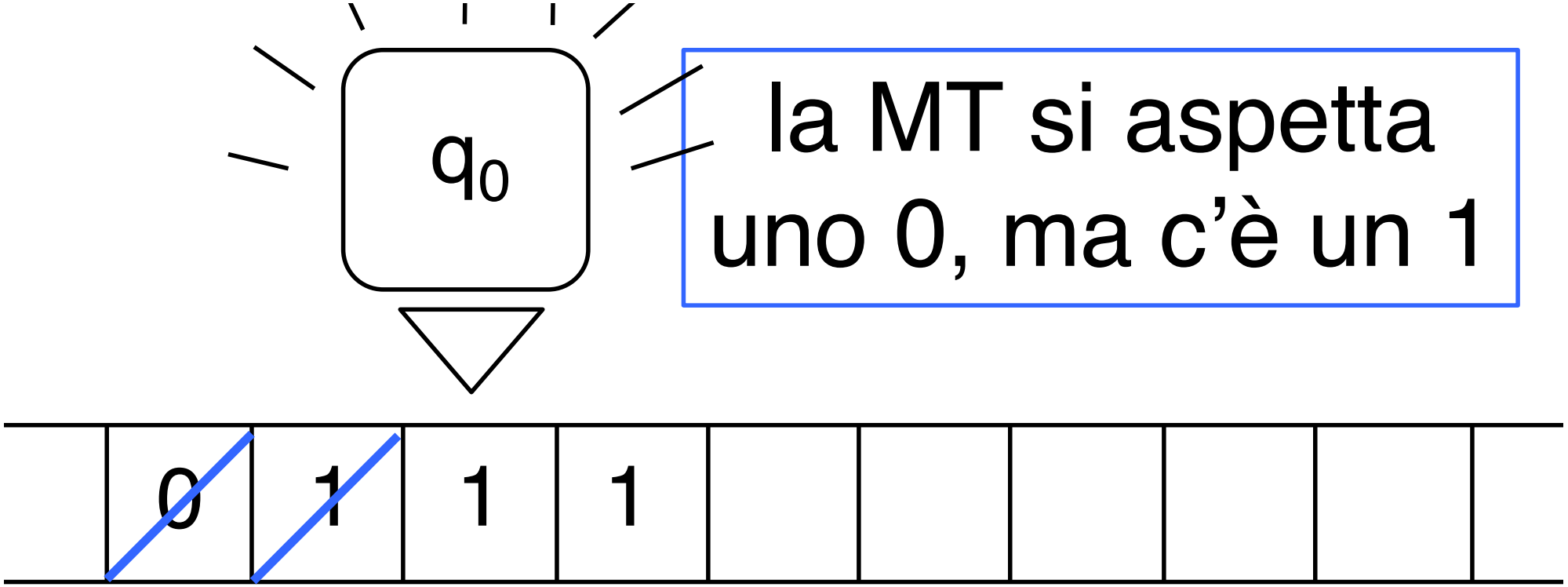
- dopo aver cancellato un 1 (che, con lo 0 precedente cancellato forma la coppia 01), la MT torna ad aspettarsi uno 0
- quindi possiamo farla tornare nello stato iniziale q_0, e farle ripetere la procedura per elaborare un'eventuale coppia 01 successiva



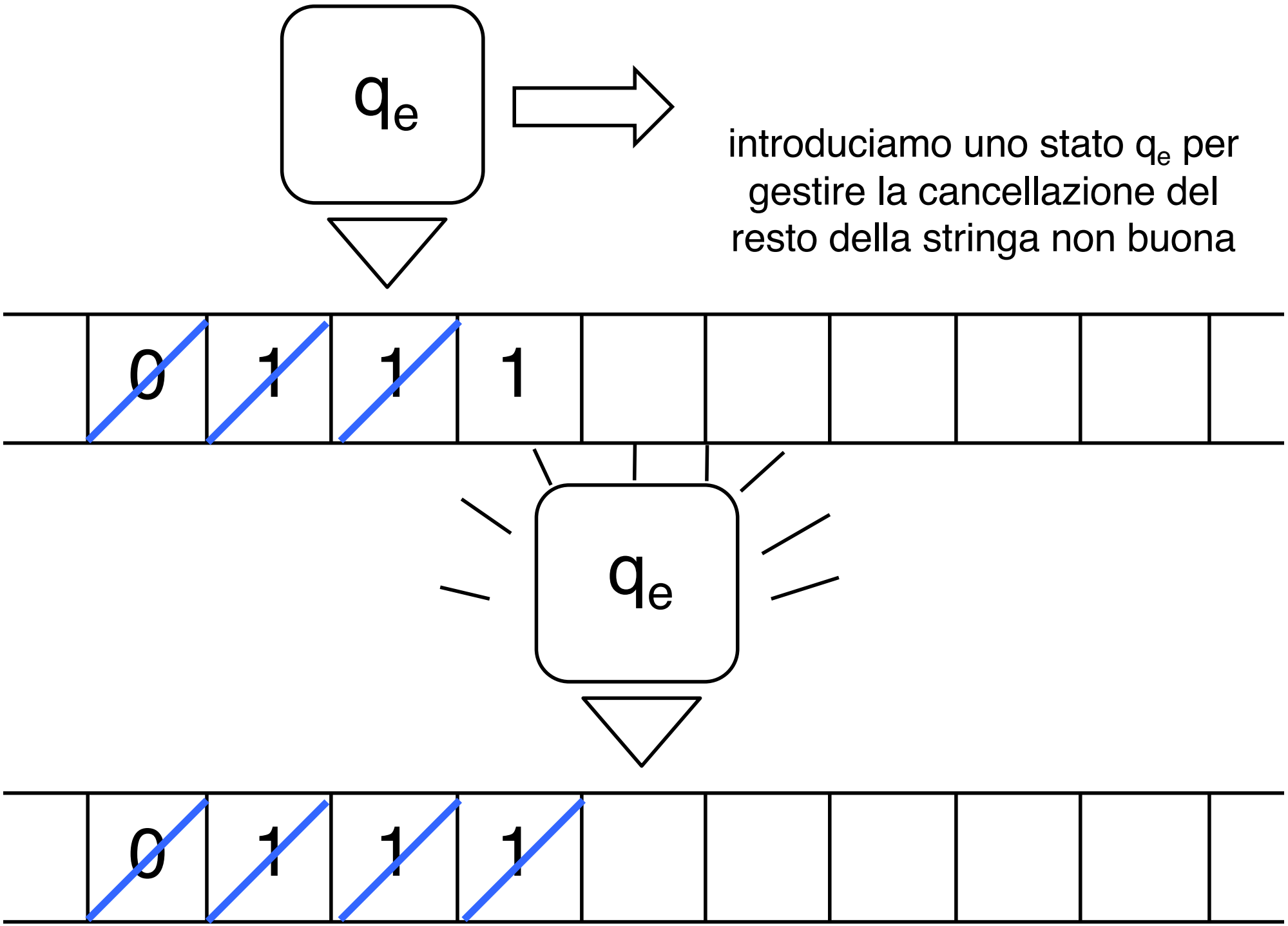
- quando la MT, nello stato q_0, anziché un nuovo 0 si ritrova una cella vuota sotto la testina, vuol dire che ha cancellato una sequenza di coppie 01
- a questo punto la MT ha cancellato una stringa buona: scrive 0 per accettarla e termina

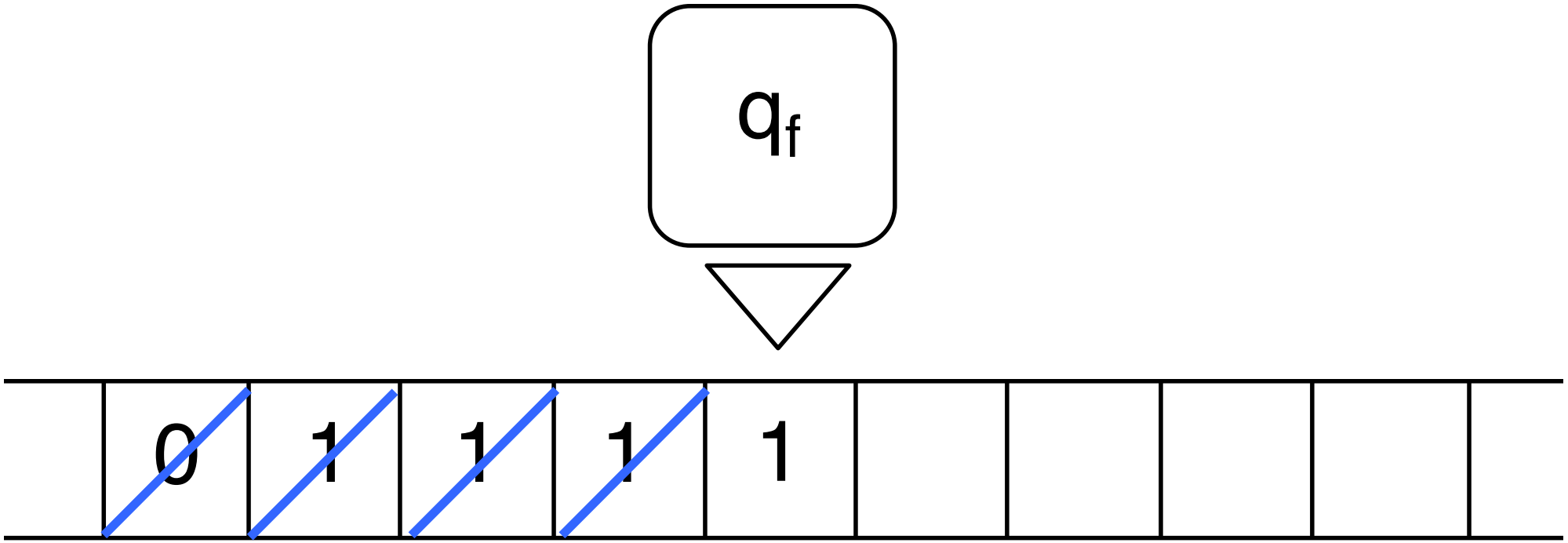


- questa “run” riguarda un caso buono, in cui sul nastro è presente una stringa da accettare
- naturalmente, dobbiamo occuparci anche dei casi in cui la stringa è da rifiutare

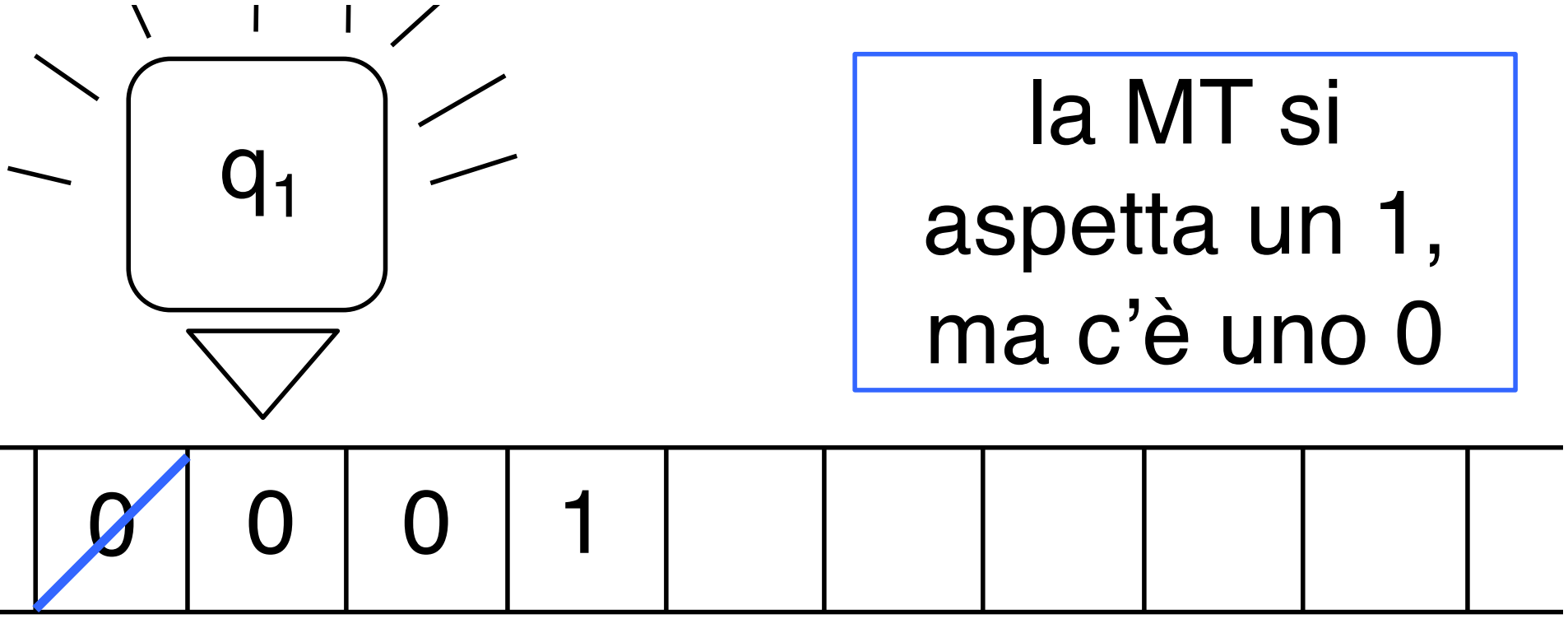


- nello stato q_0 la MT si aspetta uno 0 sul nastro, altrimenti la stringa va rifiutata
- la MT dovrà cancellare l'1 inaspettato, e tutto quello che segue, fino alla prima cella vuota


q_e
introduciamo uno stato q_e per gestire la cancellazione del resto della stringa non buona
0
1
1
1
q_e
0
1
1
1

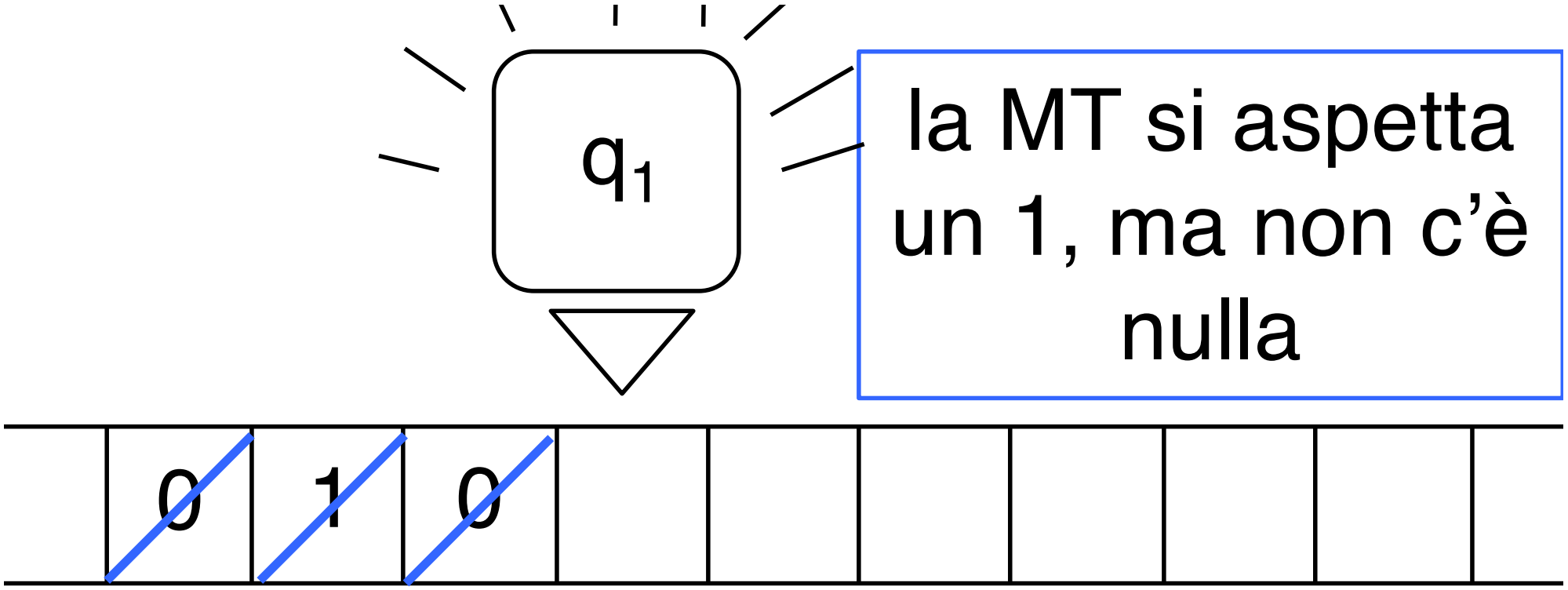


- arrivata alla prima cella vuota, la MT entrerà nello stato finale e scriverà un 1 in output per rifiutare la stringa

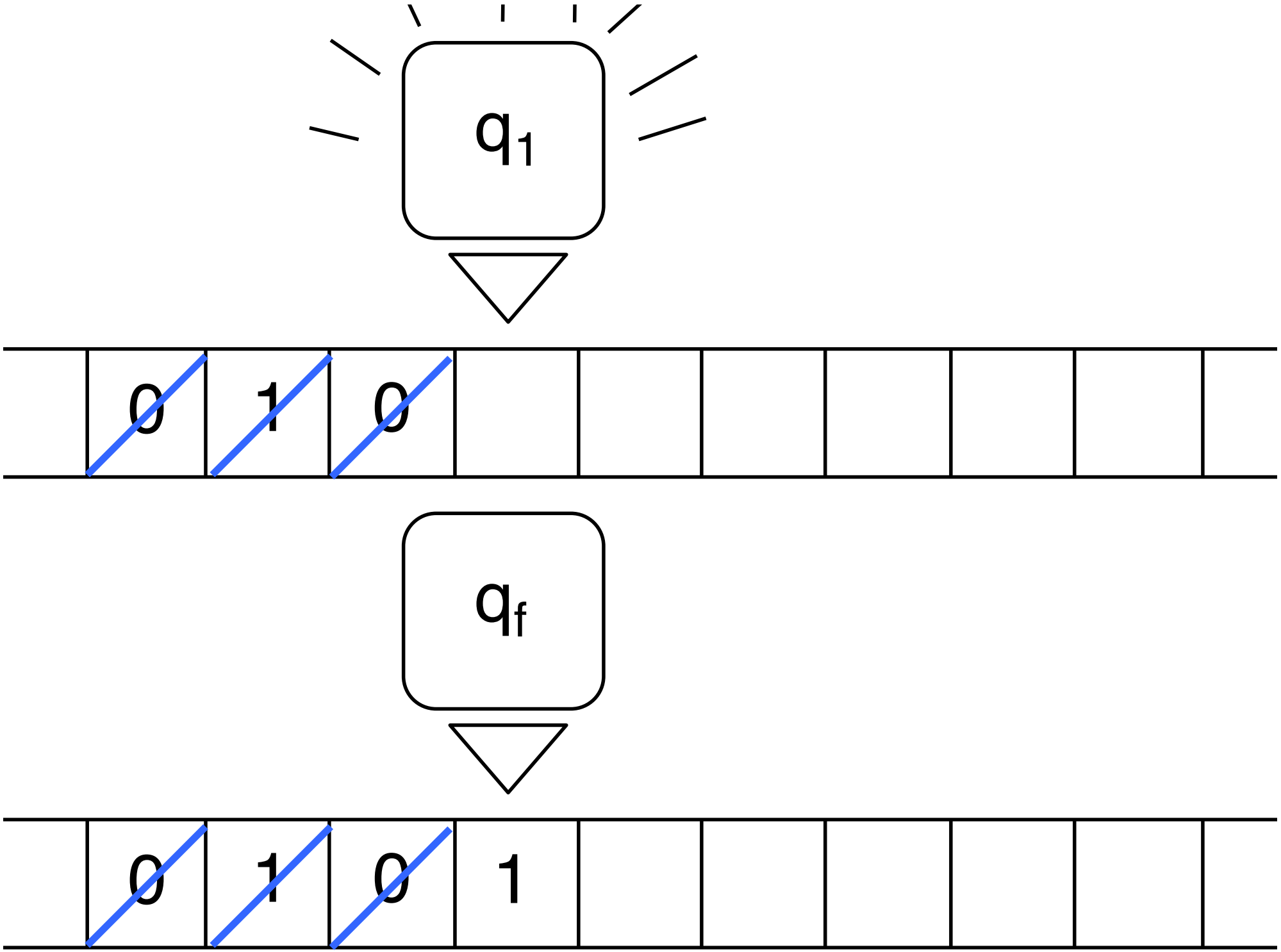


la MT si aspetta un 1, ma c'è uno 0

- analogamente, se la MT nello stato q_1, ossia che si aspetta un 1 sotto la testina, trova uno 0, si procederà allo stesso modo di prima: cancellare tutto scorrendo a destra e poi terminare con la scrittura di un 1



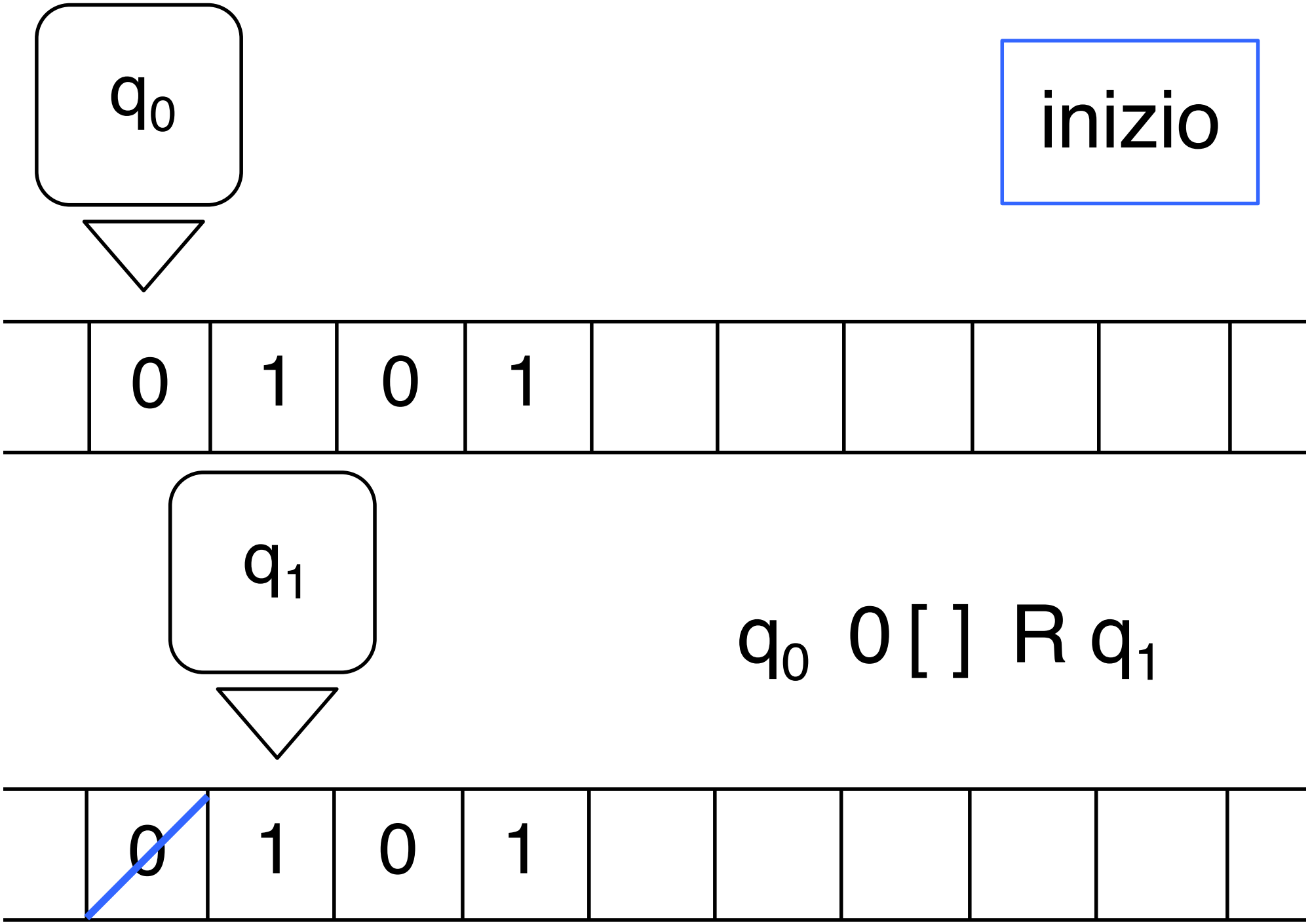
- questo è un altro possibile caso negativo: la stringa finisce con uno 0 e non c'è l'1 che lo accompagna
- in questo caso non serve cancellare ulteriormente, basta scrivere 1 e terminare

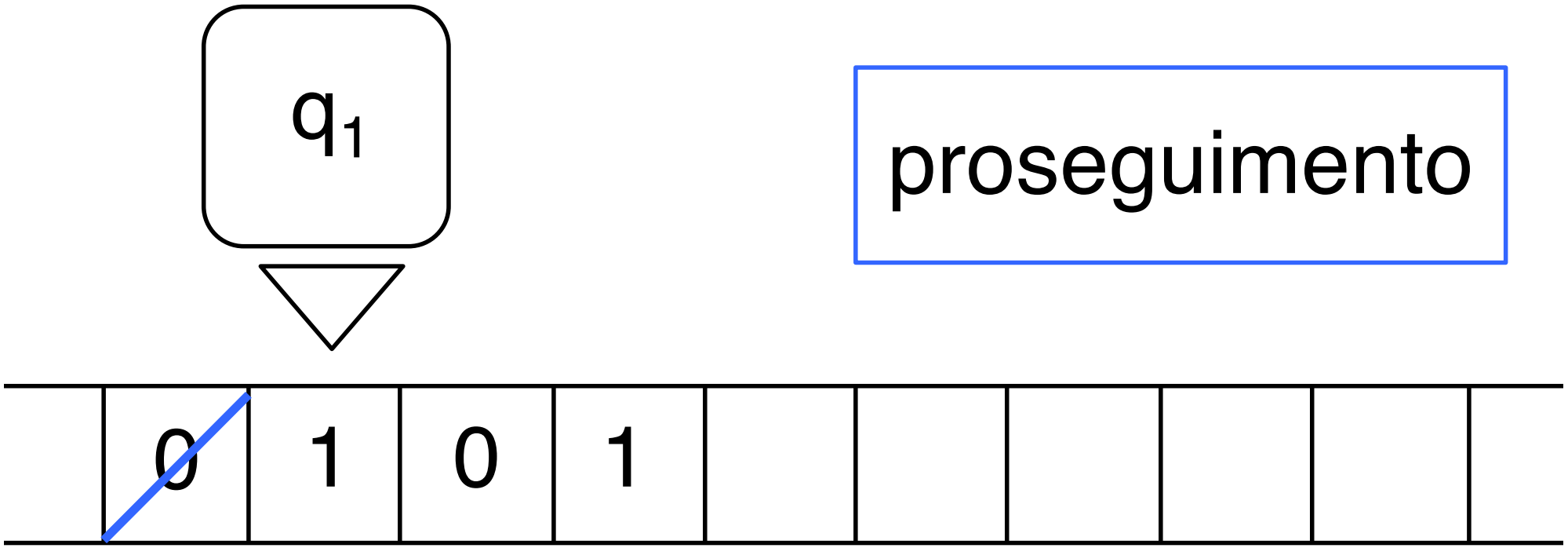

q_1
0
1
0
q_f
0
1
0
1

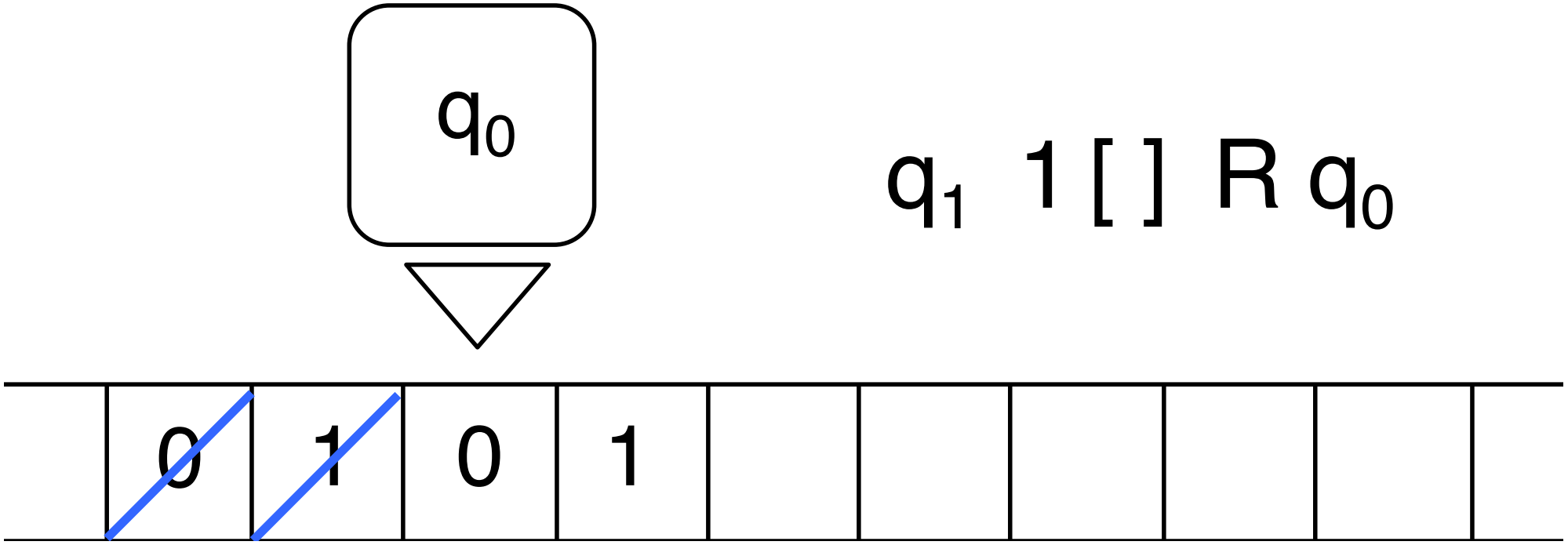
Scrittura del programma

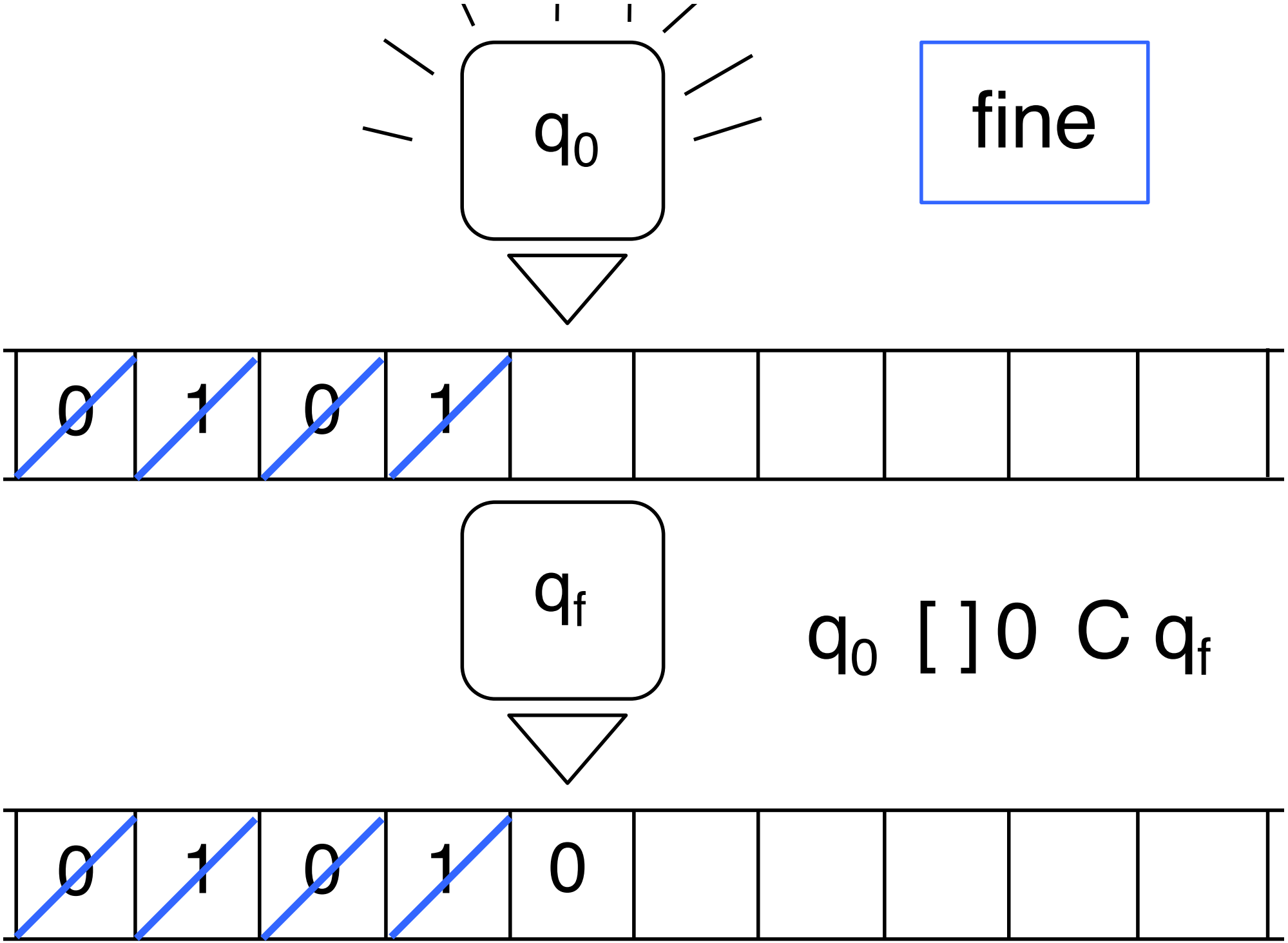
- Ora esprimiamo queste diverse configurazioni e azioni della MT in termini di istruzioni del suo programma
- Ricordiamo che le istruzioni hanno la seguente forma:

$$q_{attuale} \; s_{letto} \; s_{scritto} \; M_{ovimento} \; q_{nuovo}$$


q_0
inizio
0 1 0 1
q_1
q_0 0 [ ] R q_1
0 1 0 1


q_1
proseguimento
0
1
0
1


q_0
q_1 1 [ ] R q_0
0
1
0
1

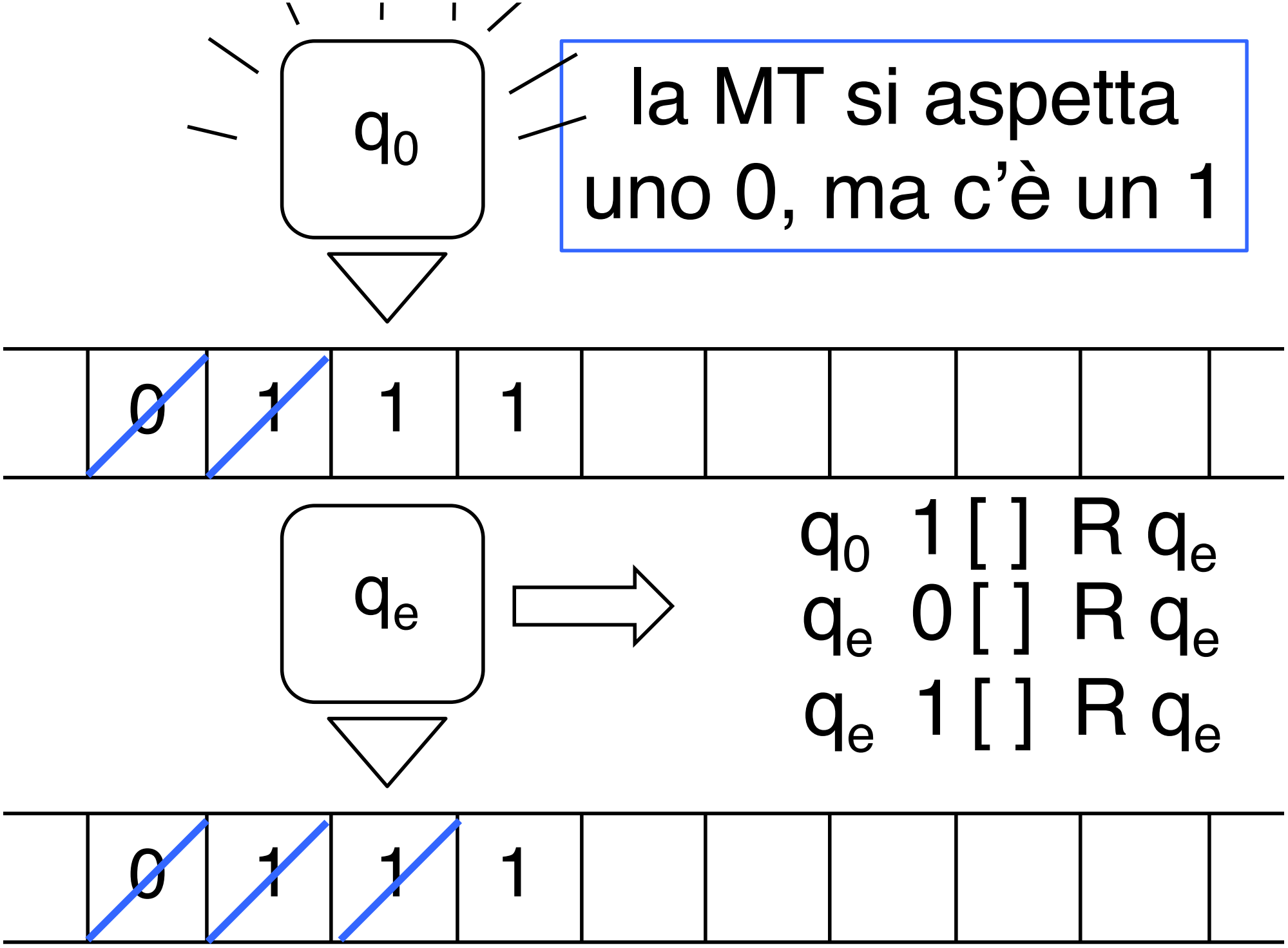

q_0
fine
0
1
0
1
q_f
q_0 [ ] 0 C q_f
0
1
0
1
0

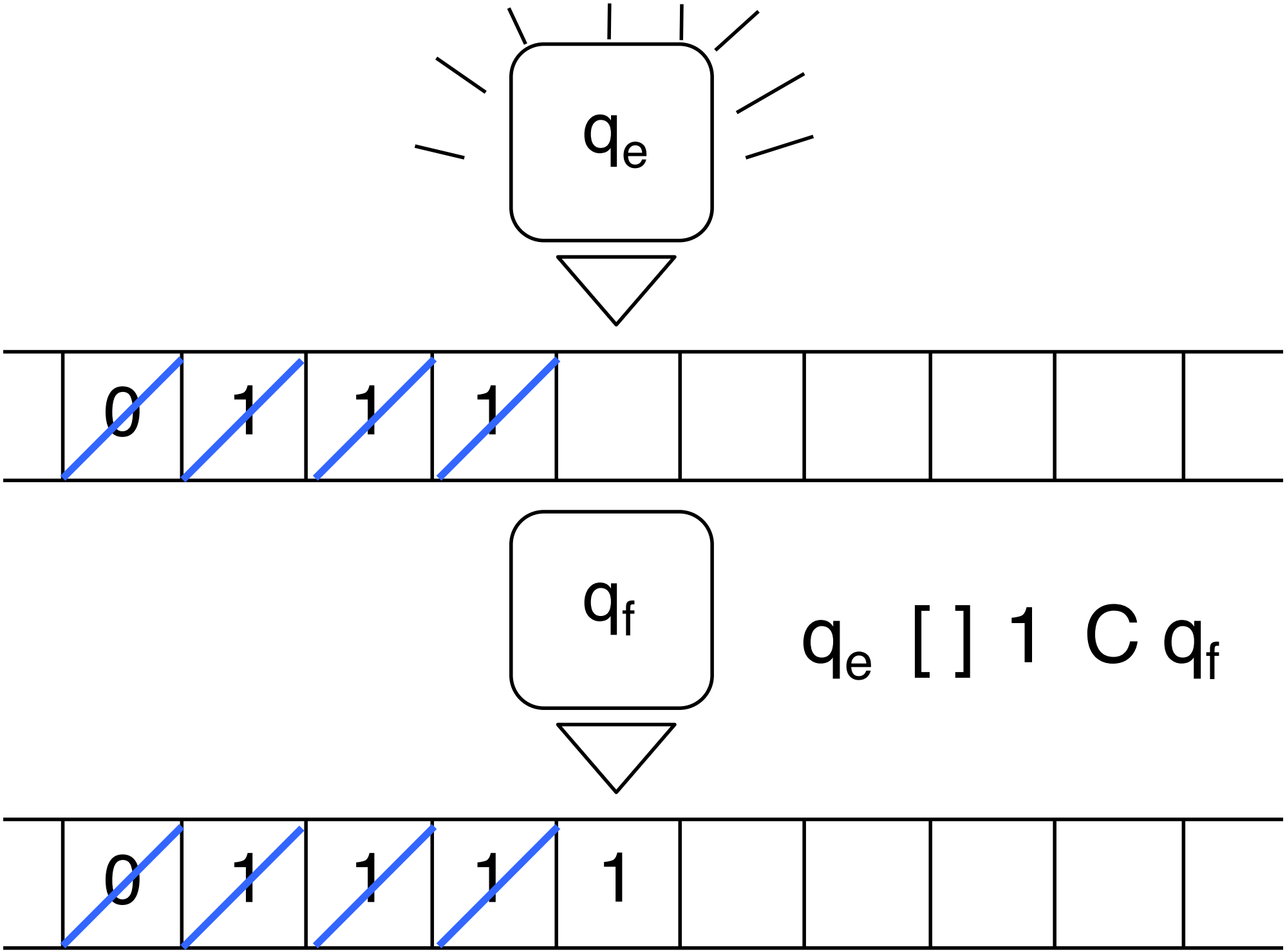
Istruzioni per l'accettazione di stringhe del linguaggio

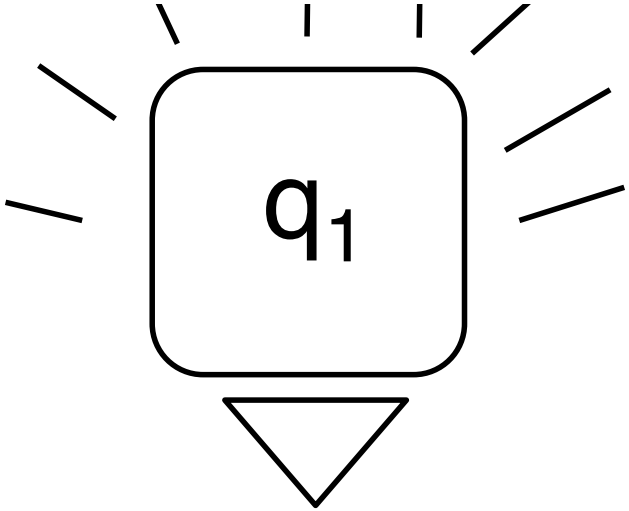
q_0 0 [ ] R q_1

q_1 1 [ ] R q_0

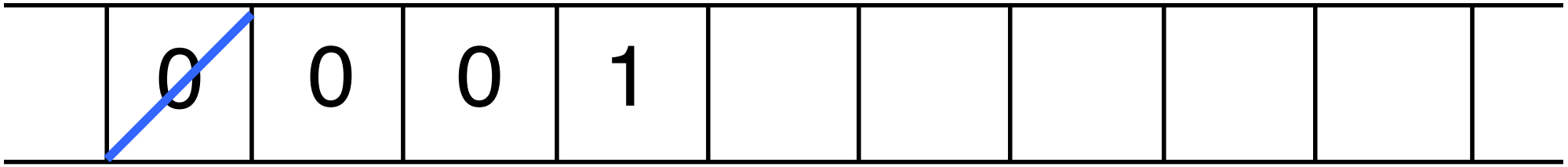
q_0 [ ] 0 C q_f


q_0
la MT si aspetta uno 0, ma c'è un 1
0 1 1 1
q_e
q_0 1 [ ] R q_e
q_e 0 [ ] R q_e
q_e 1 [ ] R q_e
0 1 1 1


q_e
0
1
1
1
q_f
q_e [ ] 1 C q_f
0
1
1
1
1

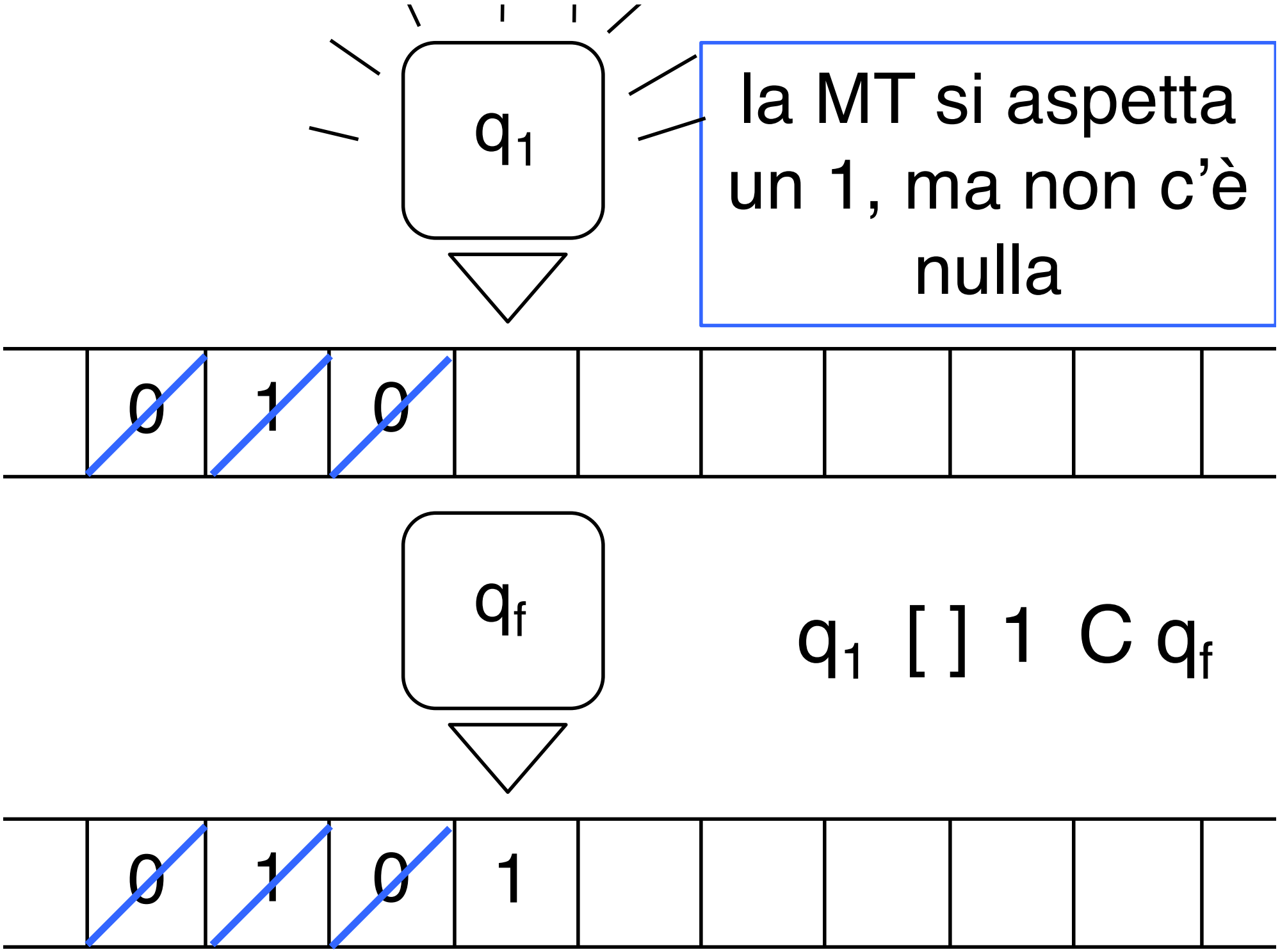


la MT si aspetta un 1, ma c'è uno 0



$q_1\ 0\,[\,]\ R\ q_e$

- non serve scrivere altre istruzioni perché sono le stesse del caso negativo precedente
- questo è solo un altro modo perché la MT finisca nello stato q_e


q_1
la MT si aspetta un 1, ma non c'è nulla
0
1
0
q_f
q_1 [ ] 1 C q_f
0
1
0
1

Mettiamo assieme le istruzioni

q_0 0 [ ] R q_1
q_1 1 [ ] R q_0

Parsing regolare di coppie 01

q_0 [ ] 0 C q_f

Accettazione della stringa

q_0 1 [ ] R q_e

Scoperta di un 1 irregolare

q_e 0 [ ] R q_e
q_e 1 [ ] R q_e
q_e [ ] 1 C q_f

Cancellazione del resto della stringa irregolare e rifiuto della stringa

q_1 0 [ ] R q_e

Scoperta di uno 0 irregolare

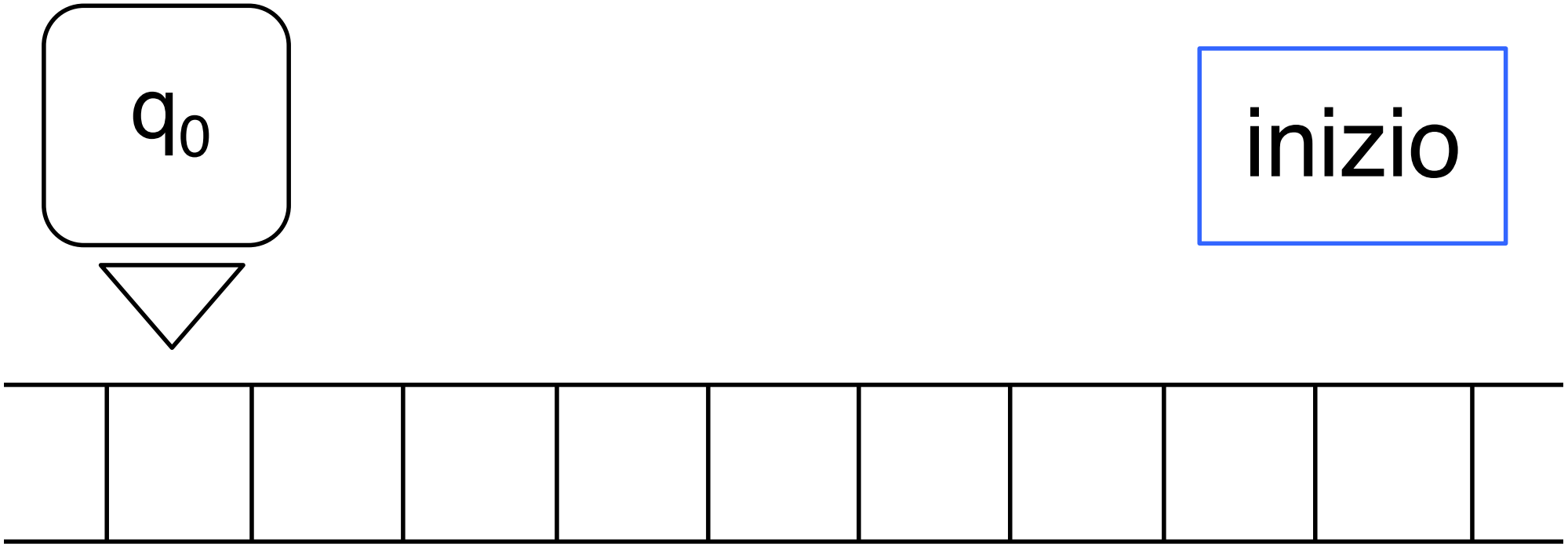
q_1 [ ] 1 C q_f

Stringa terminata irregolarmente

Condizioni al contorno

- Interroghiamoci sulle condizioni al contorno, ovvero i casi estremi
- Ecco due domande importanti, simili ma distinte:
  - Come si deve comportare la MT se fin da subito non trova niente sul nastro?
  - Come si comporta la MT se fin da subito non trova niente sul nastro?

Come si comporta la MT se fin da subito non trova niente sul nastro?



- Apparentemente non ci siamo occupati di questo caso, perché lo prendiamo in considerazione per la prima volta ora
- In realtà, c'è già un'istruzione che viene attivata da questa configurazione

q_0 0 [ ] R q_1

q_1 1 [ ] R q_0

q_0 [ ] 0 C q_f

q_0 1 [ ] R q_e

q_e 0 [ ] R q_e

q_e 1 [ ] R q_e

q_e [ ] 1 C q_f

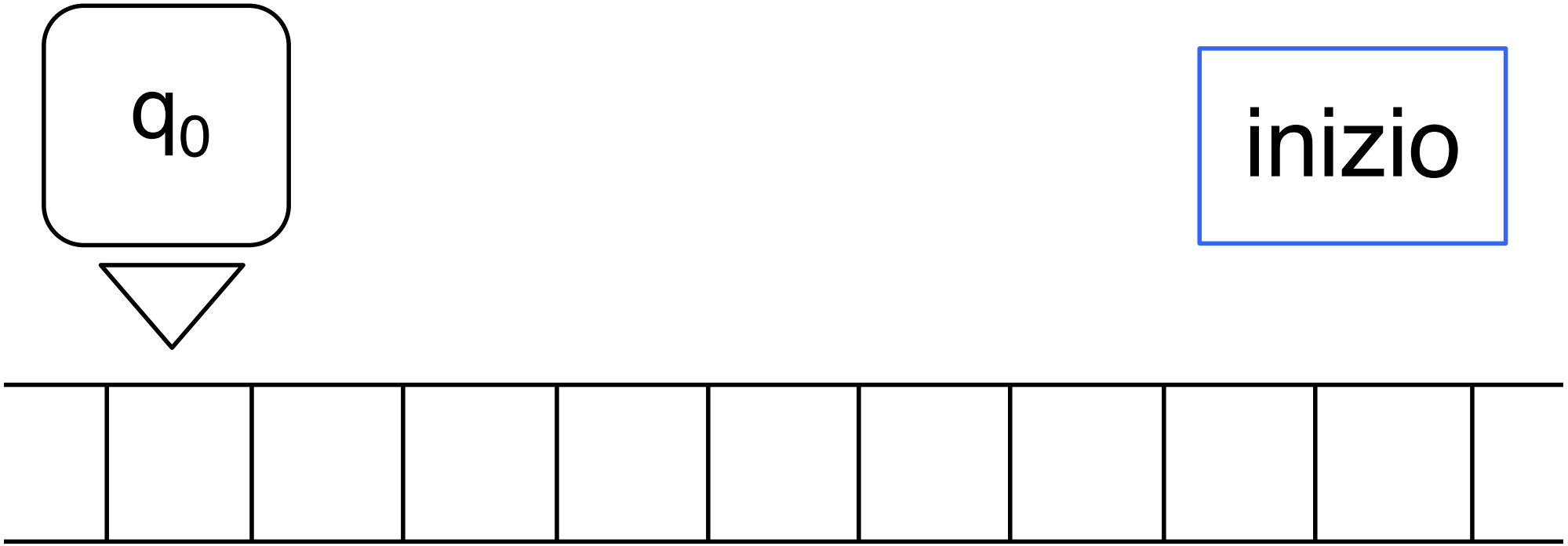
q_1 0 [ ] R q_e

q_1 [ ] 1 C q_f

Accettazione della stringa

Abbiamo scritto questa istruzione per la situazione in cui, dopo aver cancellato un certo numero di coppie 01, la MT si ritrova con una cella vuota sotto la testina, e quindi può accettare la stringa appena cancellata.

Come si comporta la MT se fin da subito non trova niente sul nastro?



- Quindi, in questa situazione, la MT, con questo programma, accetterebbe la stringa vuota, scrivendo uno 0 sul nastro e terminando

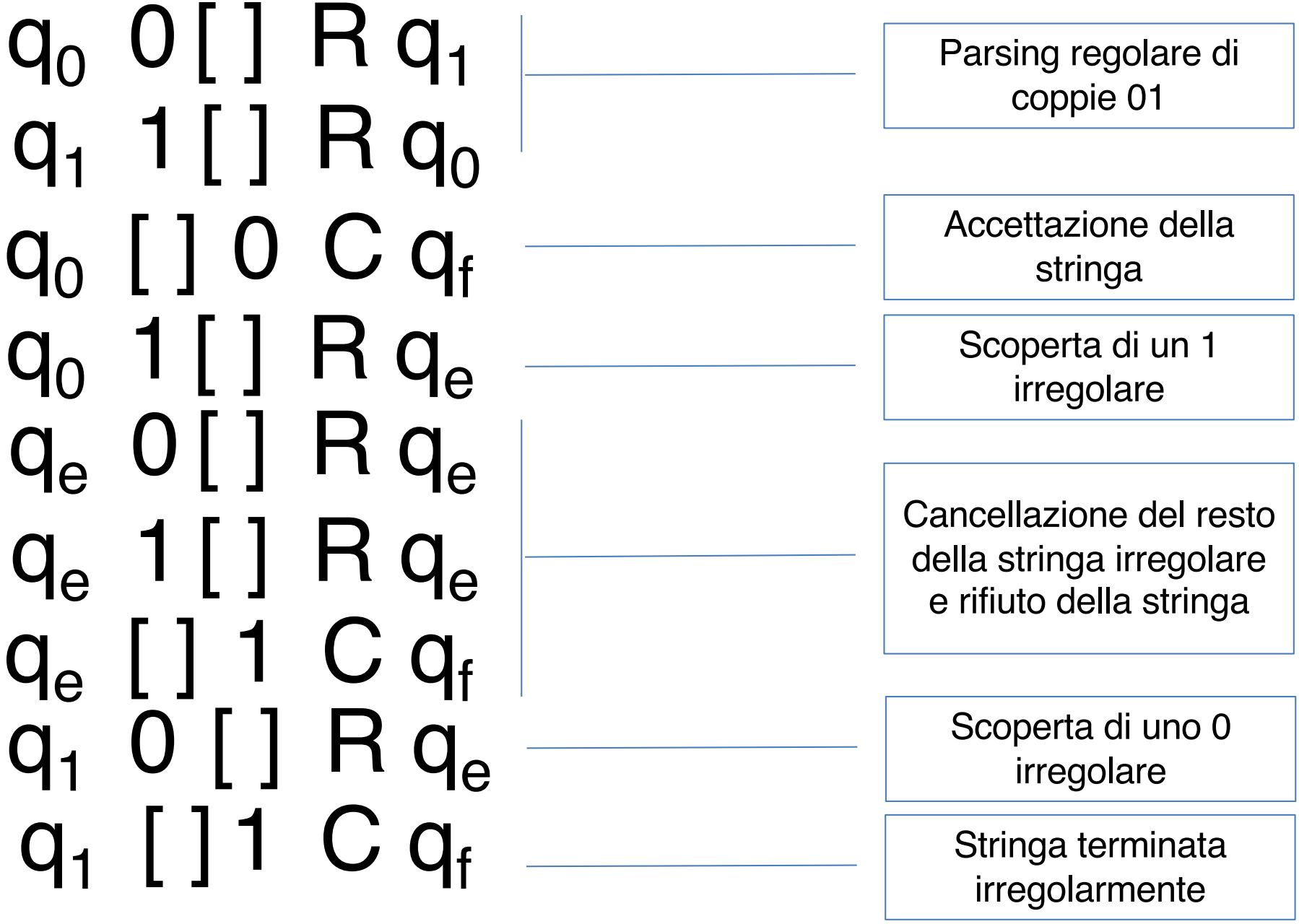
Bene così?

No.

- Creare una macchina di Turing (MT) che accetti il seguente linguaggio:
  - $(01)^i$ (con $i > 0$)

- Secondo la specifica del linguaggio, le stringhe accettate devono contenere almeno una coppia 01
- Ossia, la stringa vuota deve essere rifiutata

Come fare?


q_0 0 [ ] R q_1
q_1 1 [ ] R q_0
Parsing regolare di coppie 01
q_0 [ ] 0 C q_f
Accettazione della stringa
q_0 1 [ ] R q_e
Scoperta di un 1 irregolare
q_e 0 [ ] R q_e
q_e 1 [ ] R q_e
q_e [ ] 1 C q_f
Cancellazione del resto della stringa irregolare e rifiuto della stringa
q_1 0 [ ] R q_e
Scoperta di uno 0 irregolare
q_1 [ ] 1 C q_f
Stringa terminata irregolarmente

È come se avessimo bisogno di usare q_0 in due modi diversi

$q_0\ [\ ]\ 0\ C\ q_f$

$q_0\ [\ ]\ 1\ C\ q_f$

Accettazione della stringa DOPO AVER CANCELLATO ALMENO UNA COPPIA 01

Rifiuto della stringa PERCHÉ NON C'È NEMMENO UNA COPPIA 01

Usiamo due stati diversi

$q_{0'}$ [ ] 0 C q_f

Accettazione della stringa DOPO AVER CANCELLATO ALMENO UNA COPPIA 01

q_0 [ ] 1 C q_f

Rifiuto della stringa PERCHÉ NON C'È NEMMENO UNA COPPIA 01

q_0: stato iniziale, in cui la MT si aspetta uno 0

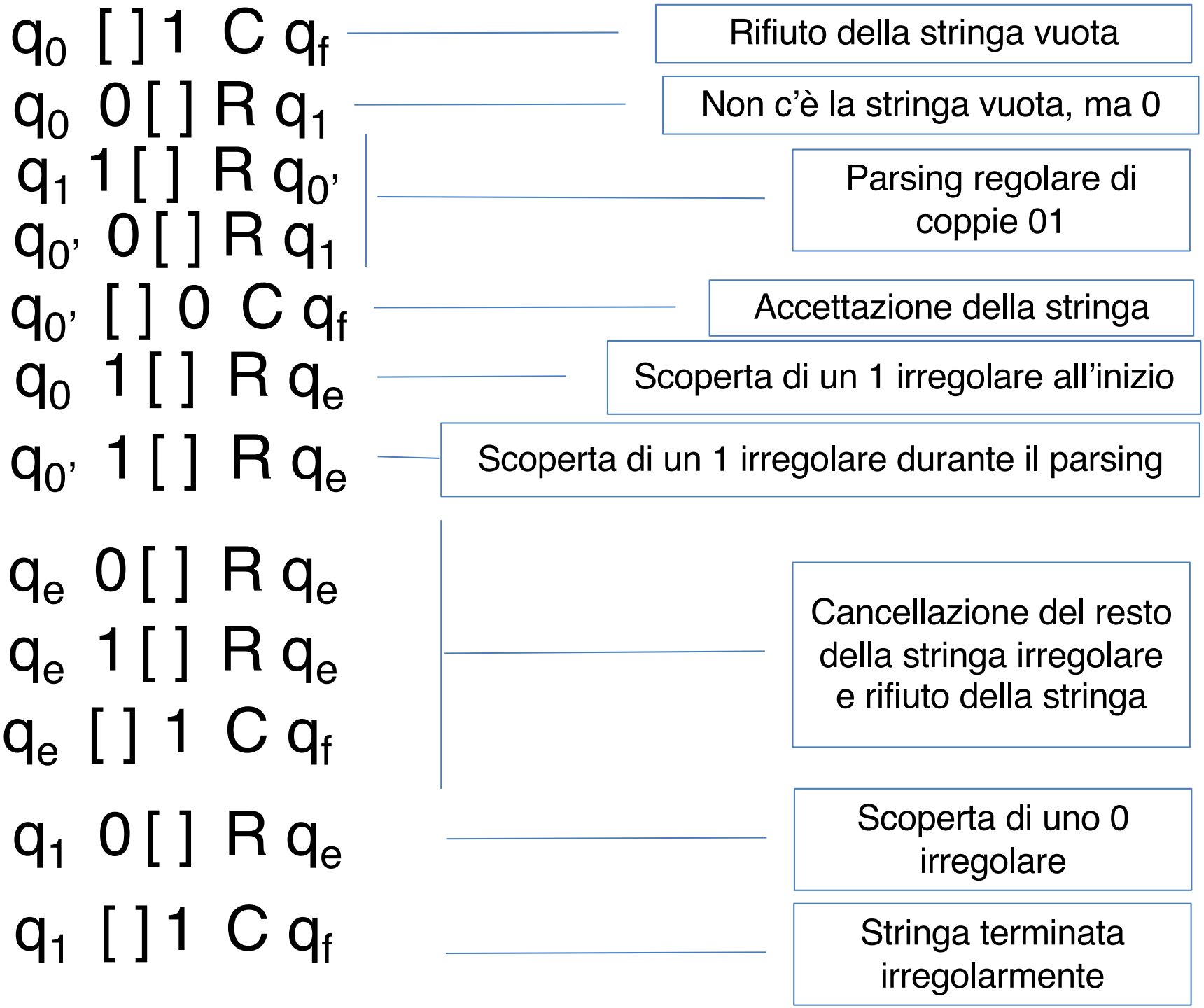
$q_{0'}$: stato intermedio, in cui la MT si aspetta uno 0

Come usare q_0 e $q_{0'}$

- q_0: stato iniziale
- $q_{0'}$: stato intermedio

} la MT si aspetta uno 0

- La MT inizia nello stato q_0 in cui può rifiutare la stringa vuota, oppure leggere e cancellare uno 0
- Se la MT legge e cancella uno 0, in futuro non tornerà più nello stato iniziale q_0 bensì andrà nello stato $q_{0'}$ quando si aspetta di leggere uno 0 sul nastro

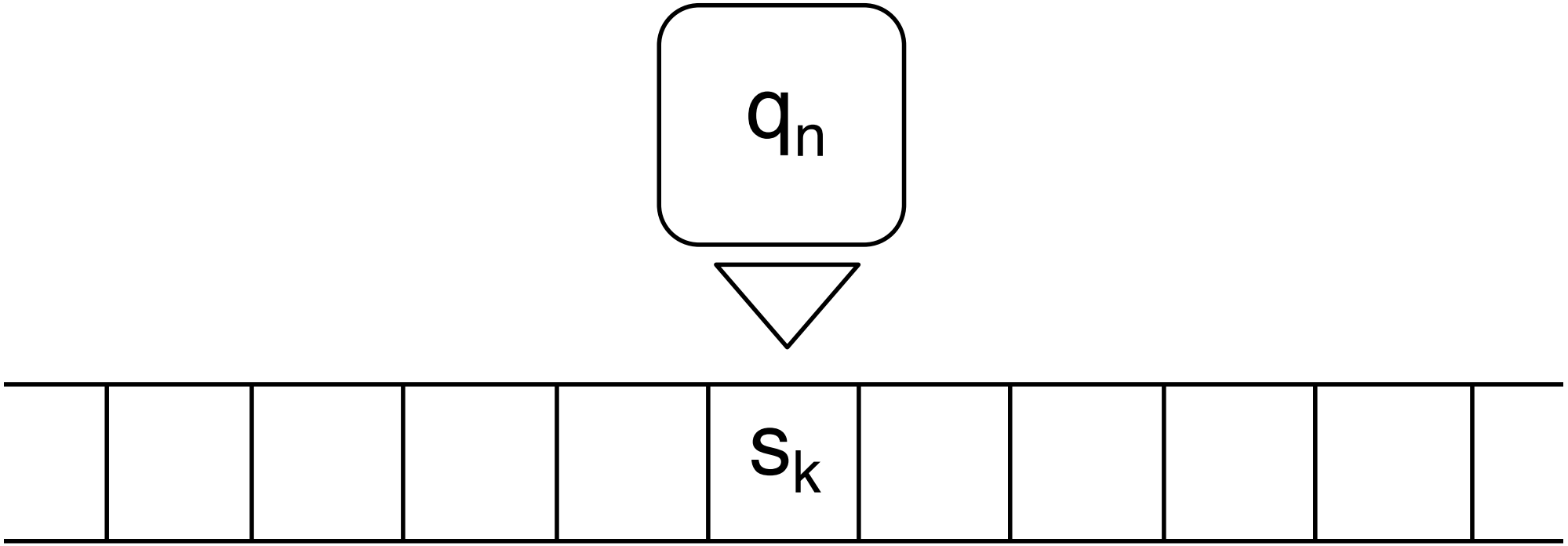

q_0 [ ] 1 C q_f
Rifiuto della stringa vuota
q_0 0 [ ] R q_1
Non c'è la stringa vuota, ma 0
q_1 1 [ ] R $q_{0'}$
$q_{0'}$ 0 [ ] R q_1
Parsing regolare di coppie 01
$q_{0'}$ [ ] 0 C q_f
Accettazione della stringa
q_0 1 [ ] R q_e
Scoperta di un 1 irregolare all'inizio
$q_{0'}$ 1 [ ] R q_e
Scoperta di un 1 irregolare durante il parsing
q_e 0 [ ] R q_e
q_e 1 [ ] R q_e
q_e [ ] 1 C q_f
Cancellazione del resto della stringa irregolare e rifiuto della stringa
q_1 0 [ ] R q_e
Scoperta di uno 0 irregolare
q_1 [ ] 1 C q_f
Stringa terminata irregolarmente

q_0 1 [ ] R q_e — Scoperta di un 1 irregolare all'inizio

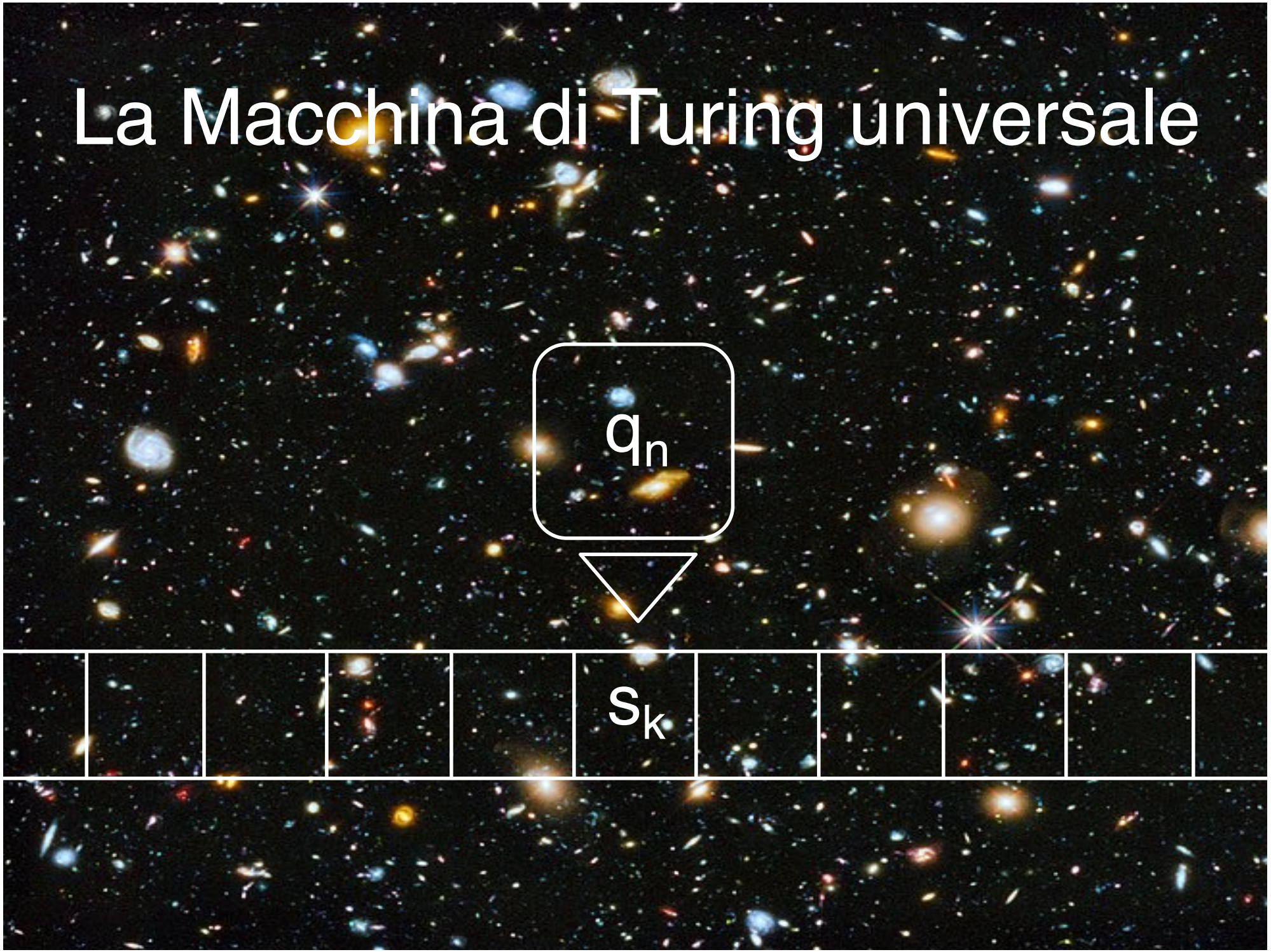
$q_{0'}$ 1 [ ] R q_e — Scoperta di un 1 irregolare durante il parsing

- Da notare questo "sdoppiamento"
- Prima c'era un solo stato in cui la MT si aspetta uno 0, ossia q_0
- Ora, invece, con la distinzione dell'inizio (q_0) dal parsing ($q_{0'}$) dobbiamo occuparci dell'1 irregolare due volte, per coprire entrambi gli stati

La Macchina di Turing



La Macchina di Turing universale



La Macchina di Turing universale

- Nota anche con l'acronimo MTU, è una Macchina di Turing in grado di simulare il comportamento di ogni altra MT
- Il concetto è facile da formulare, ma esiste davvero una MTU?
- Ancor prima di entrare nei dettagli della dimostrazione della sua esistenza, noi sappiamo già che la MTU esiste

La MTU esiste 1/n

- Il ragionamento che facciamo è controfattuale (a un certo punto facciamo un'ipotesi che va contro la realtà dei fatti e arriviamo ad un assurdo, il che conferma che tale ipotesi è errata)
- I computer che noi tutti usiamo esistono, e questi computer sono macchine calcolatrici che riescono a simulare la funzionalità di qualunque altra macchina calcolatrice

La MTU esiste 2/n

- Ad esempio, con il software giusto, il vostro computer può diventare:
  - la macchina Skype
  - la macchina iTunes
  - la macchina calcolatrice
  - la macchina Word
  - la macchina interprete Java
  - la macchina .NET
  - la macchina compilatore Python
  - la macchina Photoshop
  - la macchina Firefox
  - etc. etc.
- Di fatto, qualunque funzione computabile, se scritta sotto forma di programma, può essere eseguita dal computer

La MTU esiste 3/n

- Il vostro computer è, di fatto, una macchina computazionale universale
- La sua universalità deriva dal fatto che in memoria possiamo mettere non solo i dati da elaborare, ma anche le istruzioni con cui elaborare questi dati
- Questa è l'idea del computer di tipo "stored program" è attribuita a Von Neumann, ma…


John Von Neumann (1903 – 1957)

…la vera storia non è chiarissima.
Von Neumann, genio ungherese (a 22 anni aveva già una laurea in ingegneria chimica a Zurigo e un dottorato in matematica a Budapest) sfuggito ai nazisti e rifugiatosi negli USA, presentò l'idea nel 1945.
Molti storici, però, affermano che il lavoro avesse preso forte ispirazione da un lavoro di Turing del 1936. Inoltre, due collaboratori di Von Neumann per il progetto Manhattan (bomba atomica), Eckert e Mauchly, avevano elaborato simili idee in maniera indipendente.
Per questi motivi, la sola attribuzione dell'idea dello stored-program a Von Neumann appare inappropriata a numerosi studiosi.
La sua passione per la bomba atomica gli fu fatale: morì a 54 anni di cancro.

La MTU esiste 4/n

- Ora facciamo l'ipotesi controfattuale: forse i computer di oggi, grazie alla loro universalità, **non** sono basati sul modello concettuale della Macchina di Turing?
- D'altro canto, ogni volta che dobbiamo risolvere un problema, dobbiamo concepirne una nuova: sembra che ci siano numerose MT specializzate, ma non una MT universale

La MTU esiste 5/n

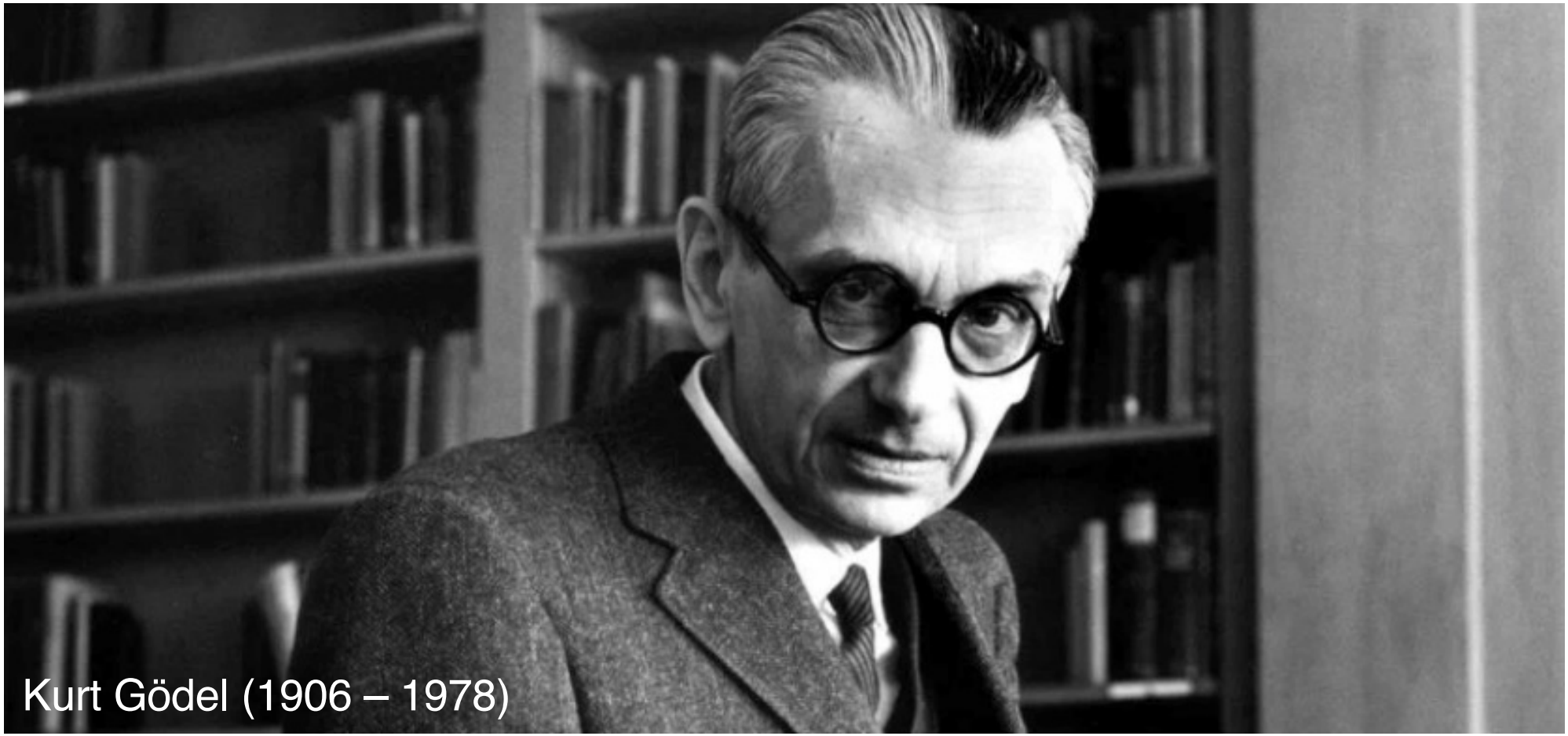
- Questo, però, va contro l'assunto che la Macchina di Turing sia un modello astratto della computazione in generale
- Molti hanno cercato di sfidare questo assunto, che non è stato dimostrato e che non può essere formalmente dimostrato perché:
  - la MT è formalmente ben definita, il concetto di "computazione" no
  - a tal punto che per definire formalmente "computazione" si fa riferimento alla MT, anche per le computazioni d'avanguardia come quelle quantistiche (con le MT non-deterministiche, su sui torneremo poi)

La MTU esiste 6/n

- A questo punto abbiamo due scelte:
  - convincerci che i computer di oggi fanno più di quello che una MT può fare
  - cercare una MTU che funga da modello teorico dei computer di oggi
- La prima scelta è una vera sfida teorica e anche tecnologica, finora mai vinta
- La seconda è sostenuta dalla seguente dimostrazione

Gödelizzazione

- Prima di iniziare la dimostrazione dobbiamo introdurre il concetto di gödelizzazione
- Il suo nome deriva dal cognome del logico matematico Kurt Gödel
- Tralasciando definizioni formali complesse e non particolarmente utili in questo contesto, possiamo intendere la gödelizzazione come un processo di codifica degli elementi di un linguaggio
- Per codifica intendiamo il suo senso più stretto: una corrispondenza biunivoca con un sottoinsieme dei numeri naturali



L’austriaco Kurt Godel è conosciuto soprattutto per la sua dimostrazione dell'incompletezza dei teoremi. Nel 1931 pubblica risultati fondamentali dimostrando che in ogni sistema matematico basato su assiomi e regole di inferenza ci sono proposizioni che non possono essere dimostrate o smentite dentro gli assiomi del sistema. In particolare, la consistenza degli assiomi non può essere dimostrata. Questo risultato segna la fine di decenni di tentativi di formalizzazione della matematica in termini di assiomi da parte di numerosi studiosi, tra cui Bertrand Russell e David Hilbert. I risultati di Godel sono un punto di riferimento per la matematica del XX secolo e implicano che un computer non potrà mai essere programmato per rispondere a tutti i quesiti matematici.
Da sempre affetto da disturbi psichici, morì di fame in ospedale, convinto che ci fossero persone intenzionate ad avvelenarlo.

Gödelizzazione delle MT

- Creiamo una funzione d() che associa a ciascun simbolo un numero dispari (es.: d(C)=5)

L	R	C	s_0	q_0	s_1	q_1	s_2	q_2	…
1	3	5	7	9	11	13	15	17	…

- All'istruzione I: $q_1\ s_0\ s_2\ R\ q_3$ associamo il numero
  $g(I) = 2^{d(q1)} * 3^{d(s0)} * 5^{d(s2)} * 7^{d(R)} * 11^{d(q3)}$
- alla macchina $MT=\{I_1;I_2;\ldots;I_n\}$ associamo il numero
  $g(MT) = 2^{g(I1)} * 3^{g(I2)} * \ldots * p_n^{g(In)}$
  dove p_n è l'n-esimo numero primo

Enumerazione delle MT

- Per l'unicità della scomposizione dei numeri in fattori primi, a ogni MT corrisponde uno e un solo gödeliano g(MT), e a un numero naturale può corrispondere al massimo una sola MT (attenzione: non tutti i numeri sono gödeliani di MT, ad esempio i numeri dispari non lo sono)
- Si induce in questo modo una enumerazione di MT, con le MT ordinate secondo l'ordine dei corrispondenti gödeliani in N:
  $MT_i < MT_j \Leftrightarrow g(MT_i) < g(MT_j)$

Considerazioni 1/3

- Occorre una precisazione sull'univocità della corrispondenza tra MT e numeri naturali
- L'ordine con cui le istruzioni di una MT sono elencate
  - non conta per la funzione svolta dalla MT:
    - $\{I_1;I_2;I_3;I_4\}$ e $\{I_3;I_1;I_4;I_2\}$ sono la stessa MT
  - conta per calcolare il suo gödeliano
    - $2^{g(I1)}*3^{g(I2)}*5^{g(I3)}*7^{g(I4)} \neq 2^{g(I3)}*3^{g(I1)}*5^{g(I4)}*7^{g(I2)}$

Considerazioni 2/3

- Abbiamo due scelte:
  - introdurre una regola nell'insieme delle MT: le loro istruzioni vanno scritte in ordine alfabetico, così non ci saranno "doppioni" di MT che hanno le stesse istruzioni ma in ordine diverso
  - considerare la tavola di una MT, così come è scritta come la definizione univoca di una MT, quindi considerare diversa la MT che ha $\{I_1;I_2;I_3;I_4\}$ come tavola dalla MT che ha $\{I_3;I_1;I_4;I_2\}$
- In questo contesto, vanno bene entrambe

Considerazioni 3/3

- Attenzione: in questo procedimento abbiamo codificato sia singole istruzioni,

$$g(I) = 2^{d(q1)} * 3^{d(s0)} * 5^{d(s2)} * 7^{d(R)} * 11^{d(q3)}$$

  sia intere MT

$$g(MT) = 2^{g(I1)} * 3^{g(I2)} * \ldots * p_n^{g(In)}$$

- C'è il rischio che i numeri che corrispondono a istruzioni e numeri che corrispondono a macchine si confondano tra loro?

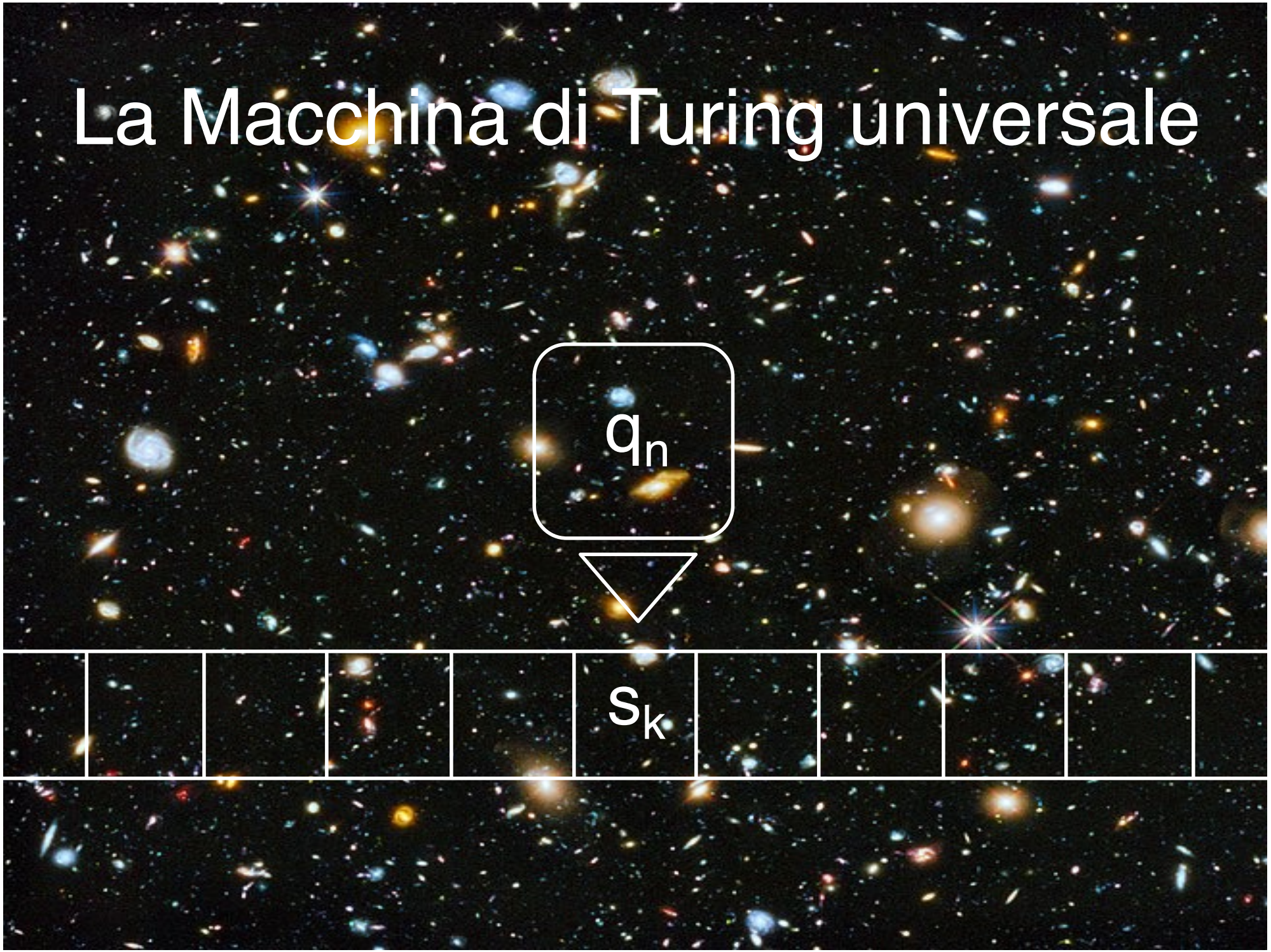
Gödelizzazione e MTU

- La gödelizzazione è il metodo per assimilare programmi e dati: in fondo, sono tutti espressi da numeri
- A questo punto, l'idea alla base dell'implementazione della MTU è chiara:
  - sul suo nastro metteremo in input il codice della MT che deve simulare e i dati da elaborare
  - in maniera analoga, nella memoria del mio computer ci sono sia il programma Powerpoint, sia il file di queste slide

Funzionamento della MTU

- In input sul nastro:
  - codifica della tavola della MT da simulare
  - input su cui eseguire la tavola
- La MTU decodifica (secondo le regole precedentemente descritte) la tavola e scrive le istruzioni della MT sul nastro
- La parte di nastro con il programma diventa la tavola da eseguire
- Una volta raggiunto lo stato finale della MT, la MTU cancella tutto sul nastro tranne il risultato in output

La Macchina di Turing universale



Funzione parziale

- Una funzione f: D $\rightarrow$ C si dice parziale quando per alcuni elemeni x in D non è definita, ossia non esiste alcun elemento y in C tale che y = f(x)
- In tal caso si scrive f(x) = $\perp$
- Il sottoinsieme di D di tutti e soli gli elementi x per cui esiste una y = f(x) si chiama campo di esistenza della funzione

Esistenza delle funzioni parziali (1/3)

- ALG, l'insieme degli algoritmi, è numerabile
- Visto che gli algoritmi sono stringhe di lunghezza finita di un linguaggio, possono essere messi in ordine (componendo l'ordinamento in base alla lunghezza delle stringhe con il criterio di ordinamento lessicografico), e quindi possiamo considerare ALG enumerabile, e quindi esiste una funzione ϕ: $N \rightarrow$ ALG totale e computabile che genera tutti gli algoritmi in ordine

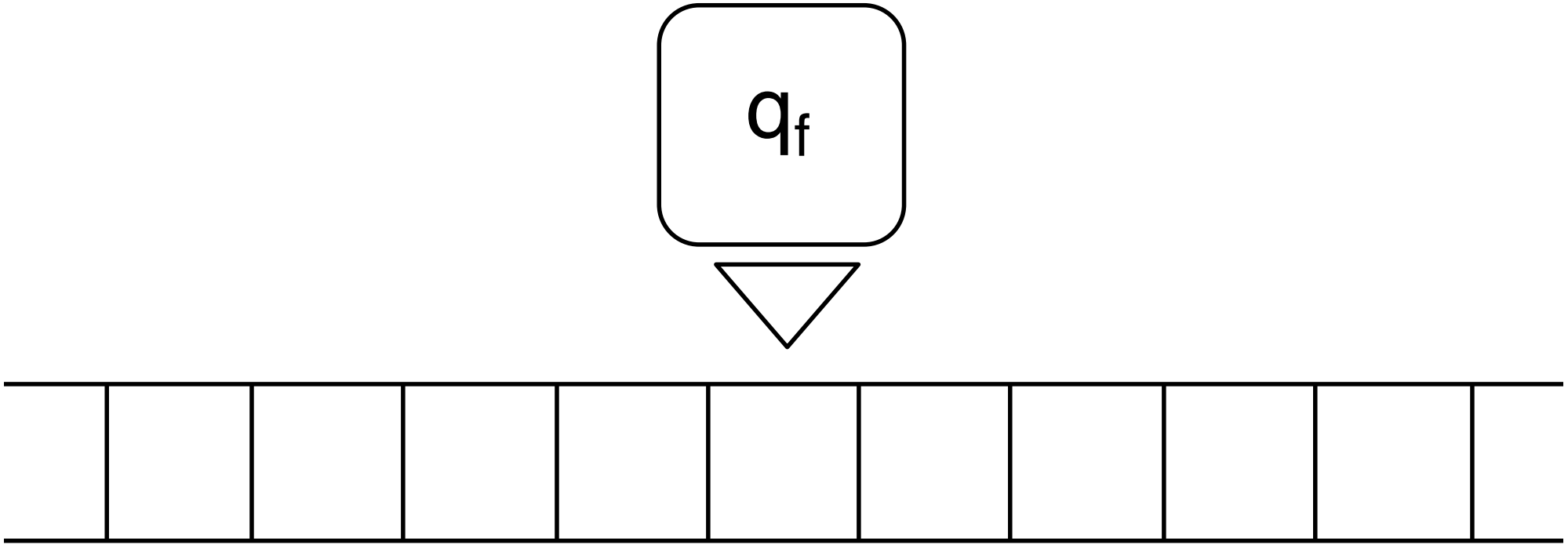
Esistenza delle funzioni parziali (2/3)

- Supponiamo (per assurdo) che gli algoritmi in ALG computino tutti funzioni totali: ossia dando in input i numeri naturali, la funzione ϕ genera una sequenza di algoritmi A_i, ciascuno dei quali computa una funzione ϑ_i totale
- Definiamo la funzione $f(x)= \vartheta_x(x) + 1$
- Essendo ϕ computabile, la ricerca della x-esima funzione ϑ_x è un procedimento algoritmico, inoltre ϑ_x è totale, quindi f è computabile e totale
- Essendo computabile e totale, f è nella lista delle ϑ_i

Esistenza delle funzioni parziali (3/3)

- Essendo computabile e totale, f è nella lista delle ϑ_i
- Chiamiamo k la posizione della funzione f nella lista
- Calcoliamo f(k):
  - per la definizione di f, $f(k) = \vartheta_k(k) + 1$
  - poiché f è la k-esima funzione nella lista, $f(k) = \vartheta_k(k)$
  - si ottiene $\vartheta_k(k) + 1 = \vartheta_k(k)$, ossia $1 = 0$, che è assurdo
- L'unica via d'uscita dall'assurdo è ipotizzare che esistano funzioni parziali nella lista, e che quindi ϑ_k non sia definita in k

Macchina di Turing che computa una funzione parziale? (A)



Macchina di Turing che computa una funzione parziale? (B)