Informatica Teorica

Anno Accademico 2022/2023

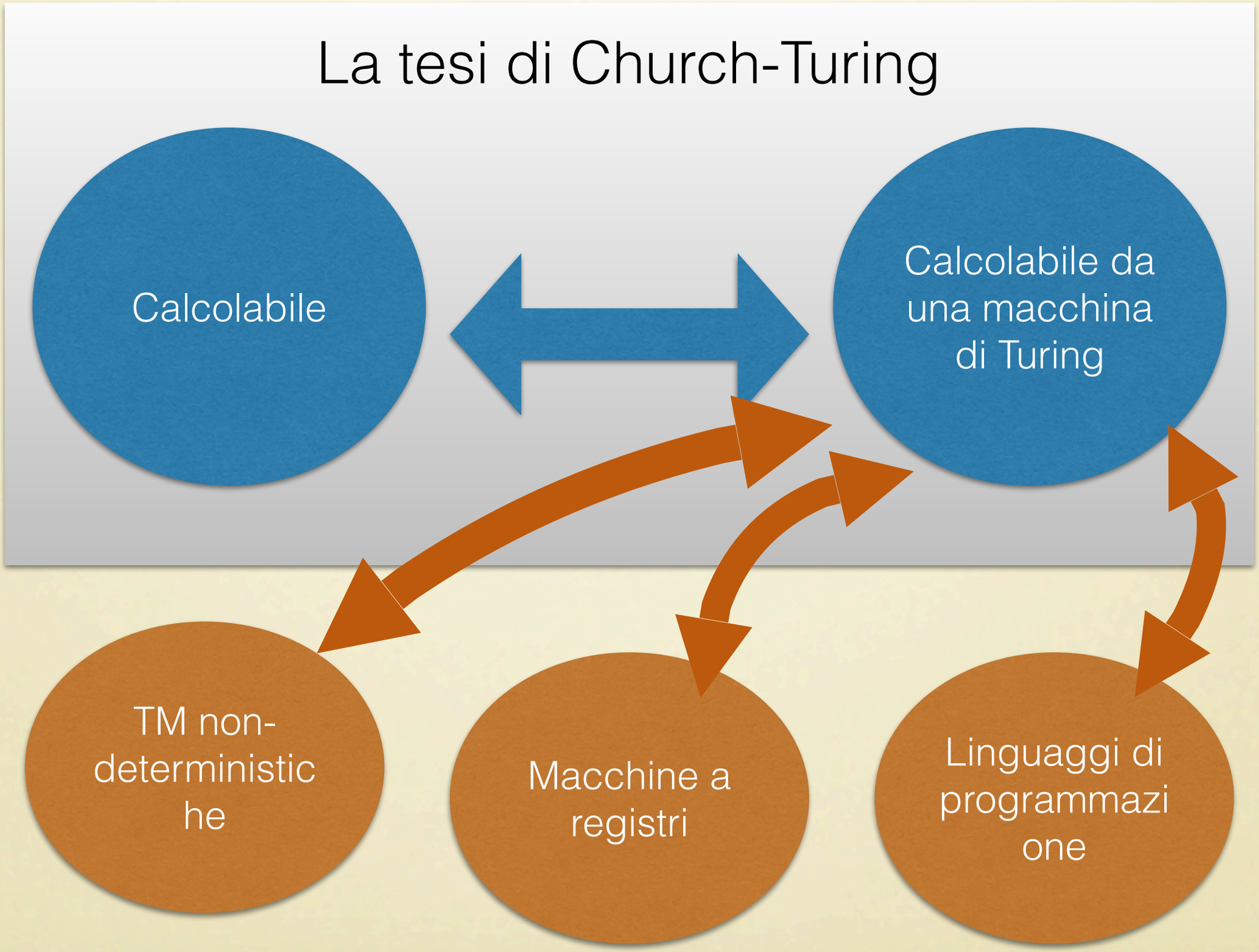
Fabio Zanasi
https://www.unibo.it/sitoweb/fabio.zanasi

Terza lezione

Nelle puntate precedenti…

- La macchina di Turing
- Problemi di decisione codificati come linguaggi formali
- Problemi decidibili e problemi riconoscibili

Questa lezione



La tesi di Church-Turing

La tesi di Church-Turing

Se la soluzione di un dato problema può essere calcolata attraverso una procedura algoritmica, allora può essere calcolata da una macchina di Turing.

Riflessioni sulla tesi di Church-Turing

É una congettura: non abbiamo una dimostrazione.

Tuttavia, abbiamo moltissime ragioni per ritenere sia vera.

Le macchine di Turing calcolano la stessa classe di funzioni di:

- Macchine di Turing 'migliorate' (non-deterministiche, probabilistiche, più nastri…)
- Macchine a registri
- Linguaggi di programmazione di alto livello come Python, Java, C, …
- (Codice macchina di) computer classici
- Computer quantistici

Riflessioni sulla tesi di Church-Turing

La tesi di Church-Turing non afferma nulla riguardo l'**efficienza** o la **semplicità** della computazione.

In effetti, le macchine di Turing sono severamente limitate in certi aspetti, che le rendono inutilizzabili per fini 'pratici'.

- Le macchine di Turing sono intrinsecamente più lente di altri modelli di calcolo perché l'accesso ai dati é **sequenziale**.
- Sono estremamente **difficili da progettare**.
  - Provate a scrivere la funzione di transizione di una TM che ordina una lista di interi a 32-bit…

Riflessioni sulla tesi di Church-Turing

…tuttavia le TM mantengono una grande importanza concettuale:

1. perché forniscono una fondazione matematica chiara per **definire in modo rigoroso** cos'è un algoritmo (incluso cos'é un algoritmo non-deterministico/probabilistico/quantistico/…)

2. perché (se la tesi di C-T é vera) sono capaci di simulare qualsiasi altra macchina di calcolo. Perciò possiamo utilizzarle per **dimostrare enunciati matematici sulle possibilità e i limiti di ciò che é calcolabile**.

Nella prossima parte

Daremo prove della tesi di Church-Turing in due modi differenti:

- mostrando che la definizione di macchina di Turing é robusta.

- mostrando che altri modelli di computazione (anche simili a quelli che utilizziamo nella nostra pratica quotidiana) sono in realtà equivalenti alle macchine di Turing.

Variazioni della macchina di Turing

Robustezza della definizione

Gli scienziati misurano la robustezza di un concetto matematico (tipicamente, una definizione) come la sua capacità di rimanere invariato rispetto ai cambiamenti.

La robustezza é sintomo della bontà di una certa definizione.

La definizione di macchina di Turing é robusta?

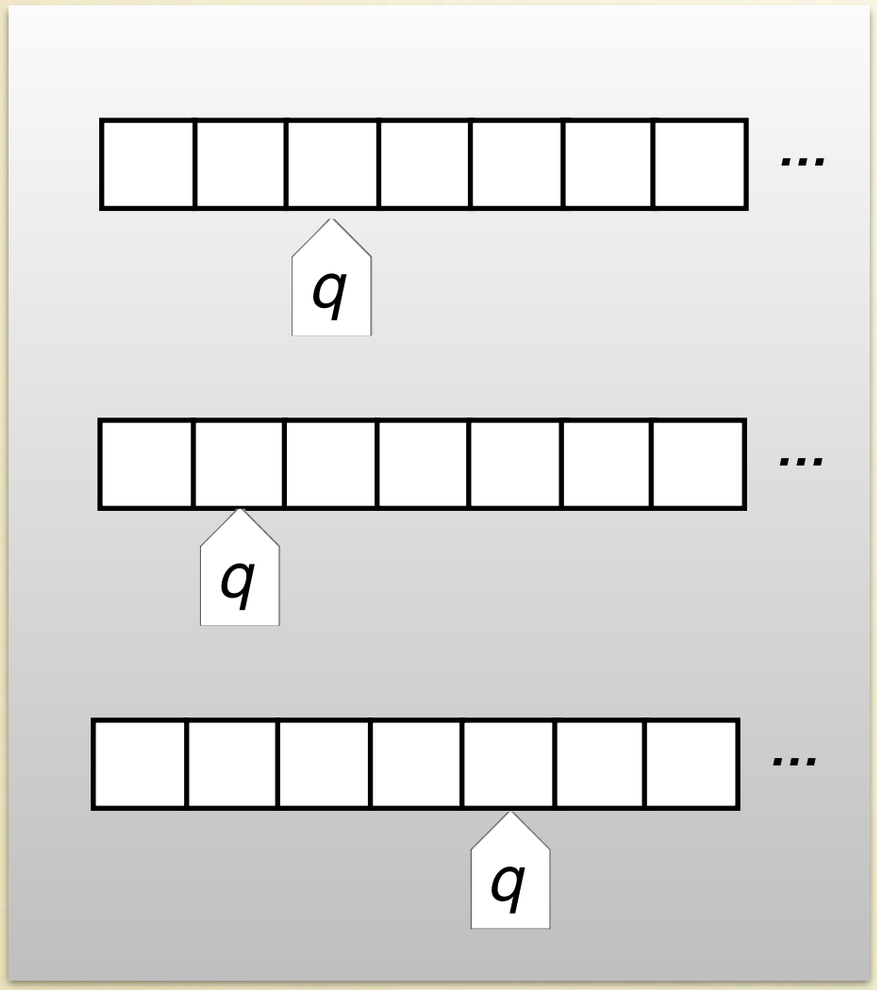
Variazioni della macchina di Turing

Sono state proposte molte varianti del concetto di macchina di Turing, che all'apparenza la rendono più espressiva.

- Nastri addizionali
- Testine addizionali
- Nastri infiniti su entrambi i lati
- Non-determinismo
- Scelta probabilistica
- Scelta quantistica
- ….

Queste variazioni sono tutte dimostrabilmente equivalenti alle macchine di Turing classiche.

Macchine di Turing con nastri addizionali

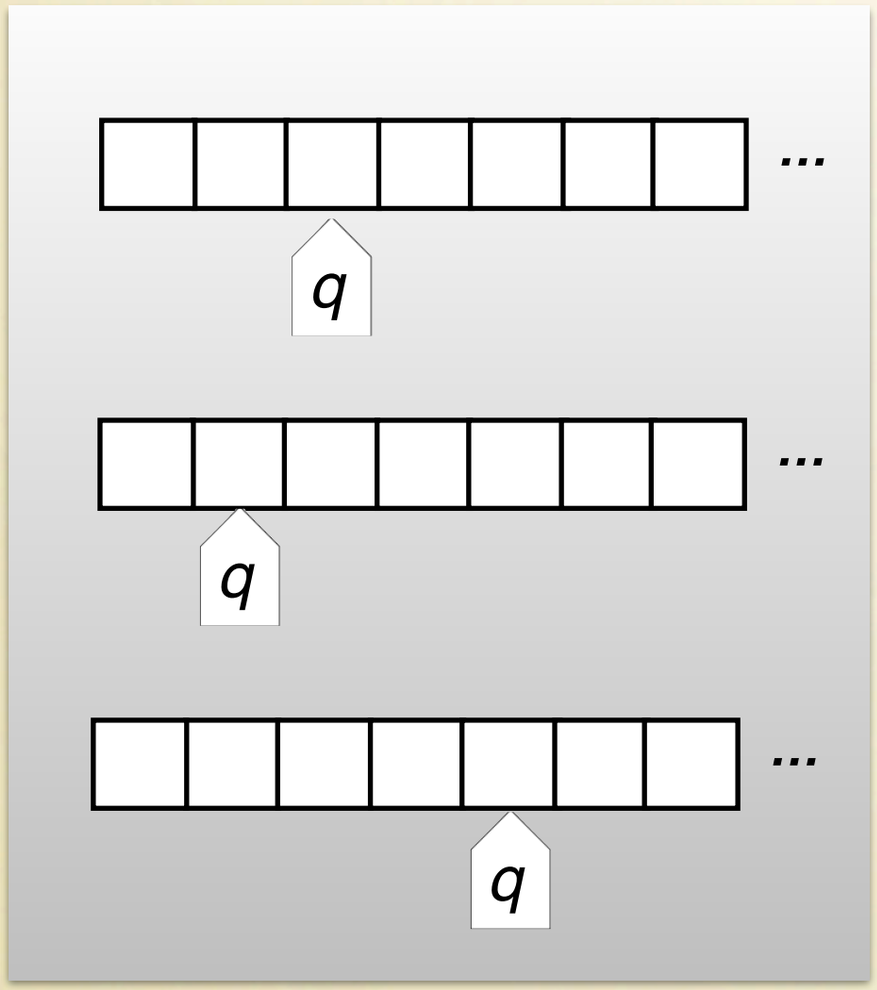


La computazione comincia con l'input sul primo nastro, e tutti gli altri nastri vuoti.

In ciascun passo di computazione, ogni testina é nello stesso stato, ma può essere in una posizione diversa, leggere un simbolo differente, e compiere un'azione diversa.

Se si raggiunge uno stato finale, l'output é letto dal primo nastro.

Macchine di Turing con nastri addizionali



Formalmente, questo modello é definito come una tupla $\langle \Sigma, Q, q_0, H, \delta \rangle$, proprio come le TM classiche.

L'unica differenza é il tipo di δ.

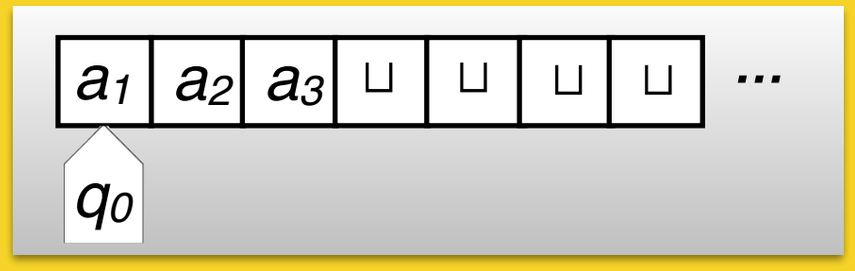
$$\delta : \quad (Q \setminus H) \times \Sigma^k \quad \rightarrow \quad Q \times (\Sigma \times \{\rightarrow, \leftarrow\})^k$$

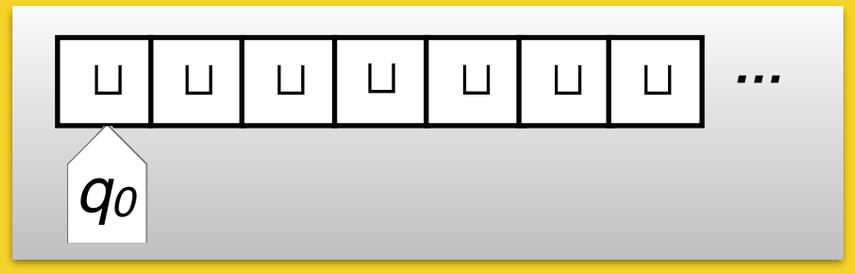
k é il numero di nastri

Macchine di Turing con nastri addizionali

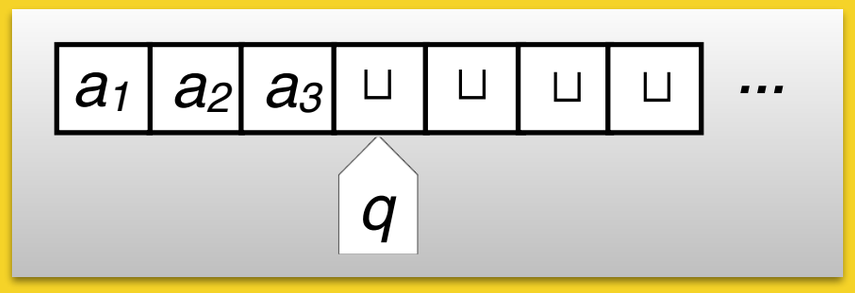
Esempio Possiamo facilmente costruire una TM con due nastri che verifica se una string di input $x = a_1\ a_2 \ldots a_m$ é palindroma.

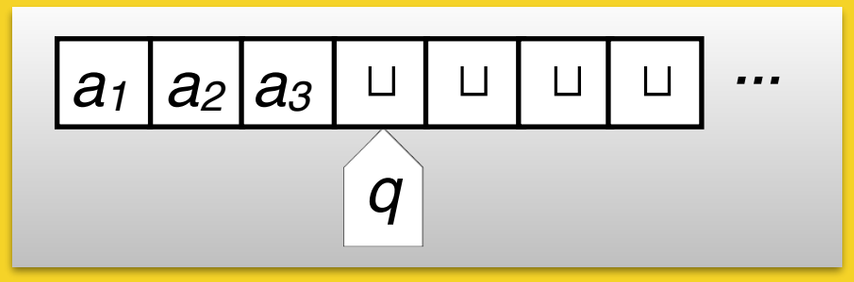
1.Configurazione iniziale.





2.Scriviamo l'input $a_1\ a_2 \ldots a_m$ sul secondo nastro.





3.Torniamo alla prima cella sul primo nastro, poi leggiamo simultaneamente da sinistra a destra sul primo nastro e da destra a sinistra sul secondo nastro. Il confronto dei simboli in ciascuna cella determina se accettare o meno.

Macchine di Turing con nastri addizionali

Teorema Macchine di Turing e macchine di Turing con nastri addizionali sono equivalenti.

Idea della dimostrazione

Una direzione dell'equivalenza é ovvia (quale?).

Per la direzione opposta: sia $\mathcal{M}$ una TM con nastri addizionali. Costruiamo una TM $\mathcal{M}'$ con un solo nastro che sia equivalente: cioè, per ogni input x,

$\mathcal{M}$ non termina su x $\Leftrightarrow$ $\mathcal{M}'$ non termina su x

$\mathcal{M}$ termina su x con output ***y*** $\Leftrightarrow$ $\mathcal{M}'$ termina su x con output

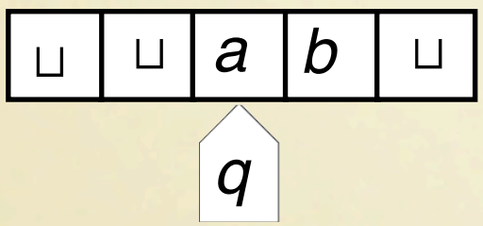
Macchine di Turing con nastri addizionali

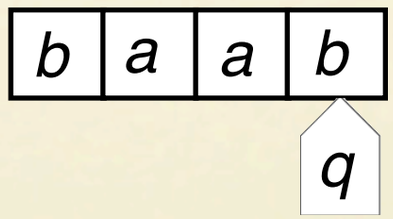
Idea della dimostrazione

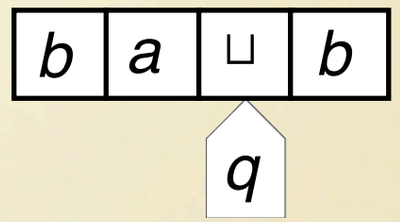
Se $\mathcal{M}$ é basata su alfabeto Σ, $\mathcal{M}'$ sarà basata su alfabeto

$$\Sigma \uplus \{\bar{a} \mid a \in \Sigma\} \uplus \{\#\}.$$

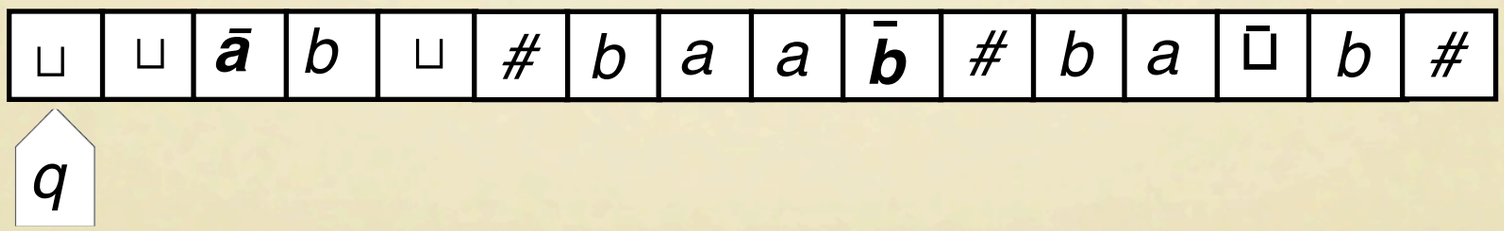
Idea: una configurazione di $\mathcal{M}$







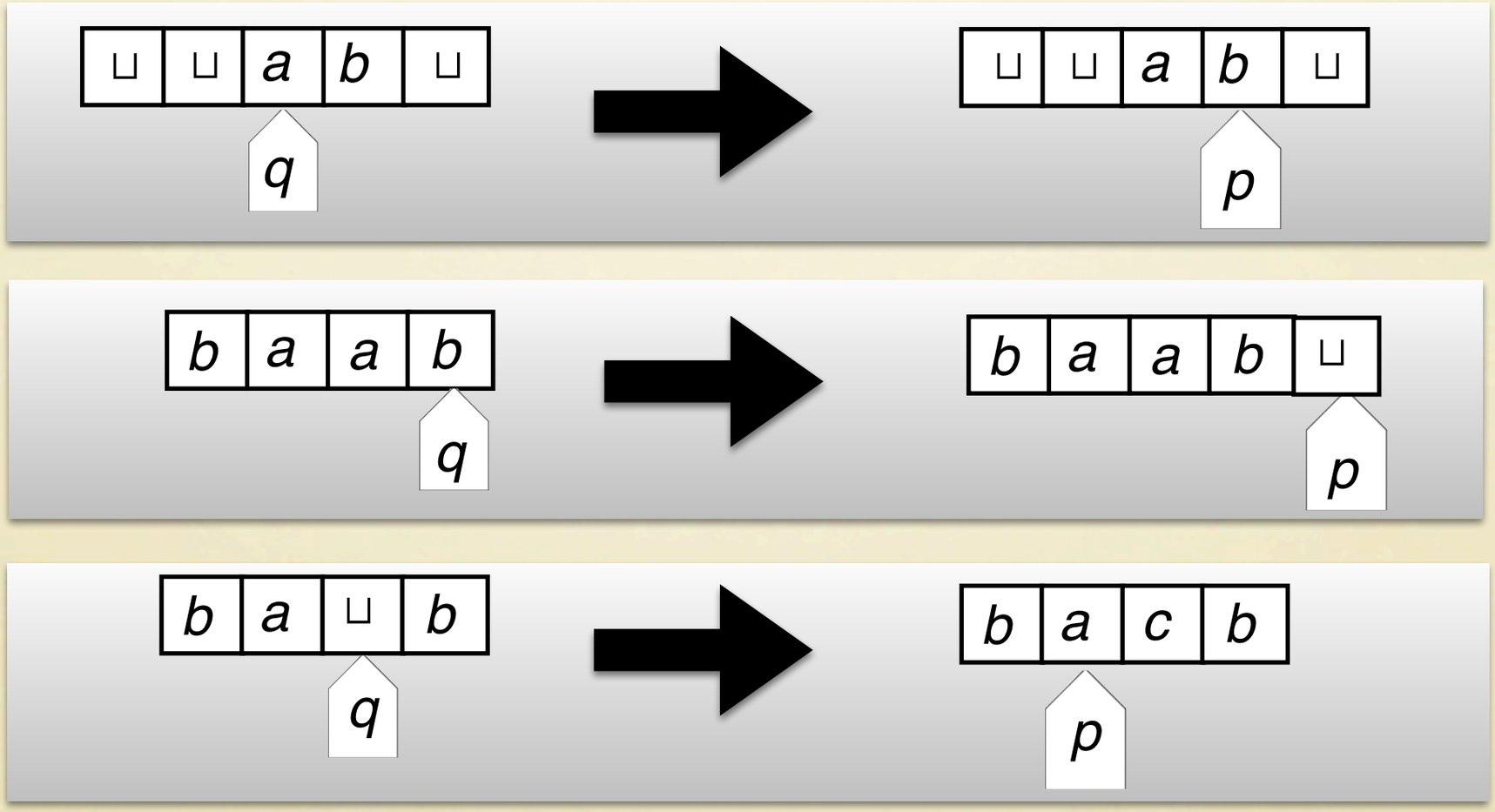
é rappresentata sul singolo nastro di $\mathcal{M}'$ come



Macchine di Turing con nastri addizionali

Idea della dimostrazione

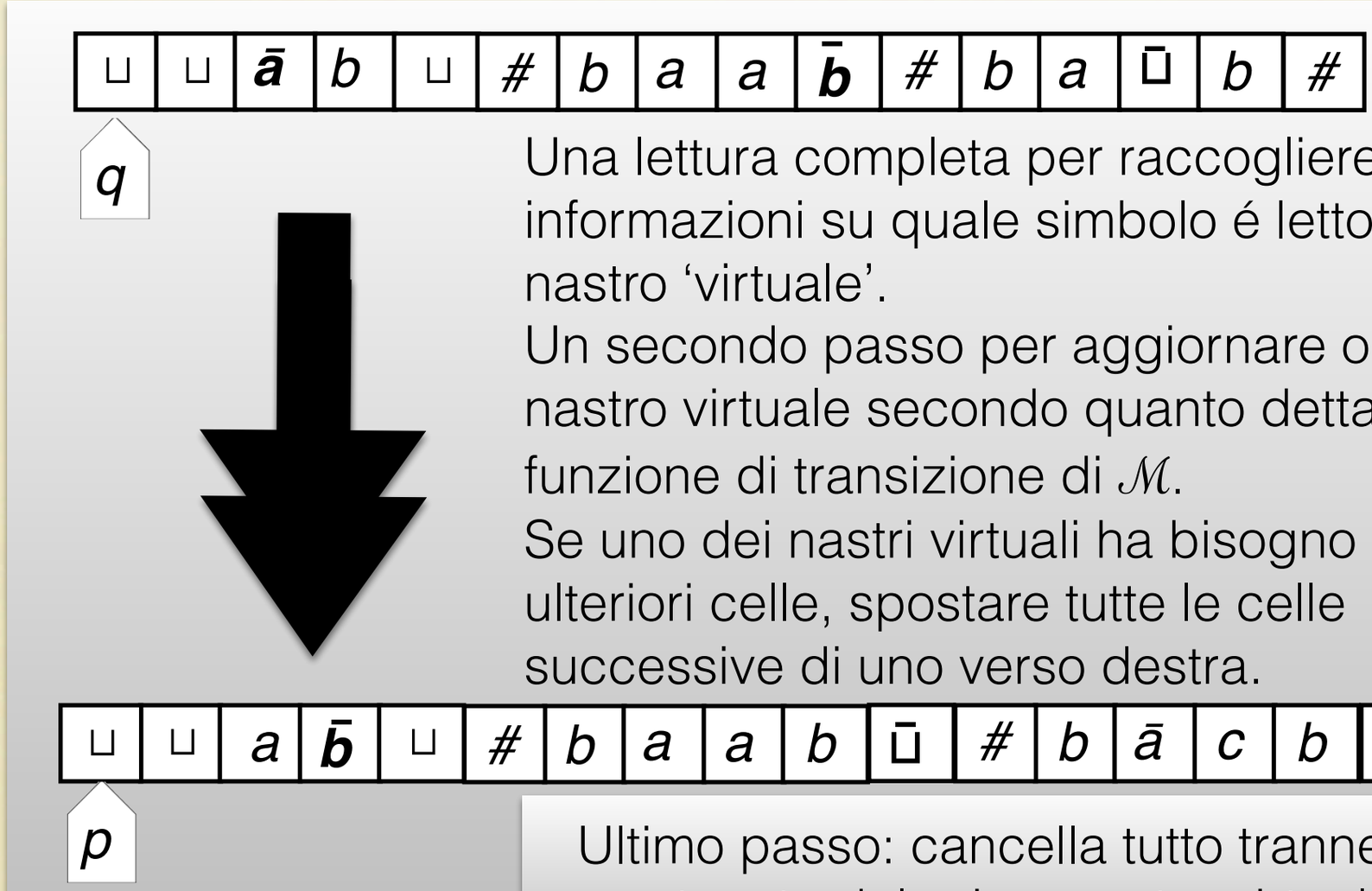
In un singolo passo di computazione, $\mathcal{M}$ raggiunge una nuova configurazione su ciascun nastro.



Macchine di Turing con nastri addizionali

Idea della dimostrazione

Questo passo é simulato da **molteplici** passi di computazione sul nastro di $\mathcal{M}'$.



Un appunto su questa (e prossime) dimostrazioni

Questa dimostrazione non é particolarmente dettagliata. Per essere completamente rigorosi, dovremmo dare la definizione completa di $\mathcal{M}'$, e dimostrare la sua equivalenza con $\mathcal{M}$.

Dati i vincoli a cui siamo soggetti, ci ‘accontentiamo' di uno sketch che sia sufficientemente convincente da credere che, se avessimo ulteriore tempo e spazio a disposizione, potremmo dare la dimostrazione completa.

Questa situazione é tipica con le macchine di Turing, che sono formalismi di ‘basso' livello. Per la tesi di Chruch-Turing, se possiamo descriverlo con un algoritmo, allora esiste una TM che lo calcola (ma negli esercizi dovete essere abbastranza convincenti!)

Macchine di Turing non-deterministiche

Stessa definizione di una TM classica, ma le transizioni sono descritte da una **relazione** anziché una funzione.

$\delta : (Q \backslash H) \times \Sigma \not\rightarrow Q \times \Sigma \times \{\rightarrow, \leftarrow\}$ **DET**

$\delta : (Q \backslash H) \times \Sigma \times Q \times \Sigma \times \{\rightarrow, \leftarrow\}$ **NON-DET**

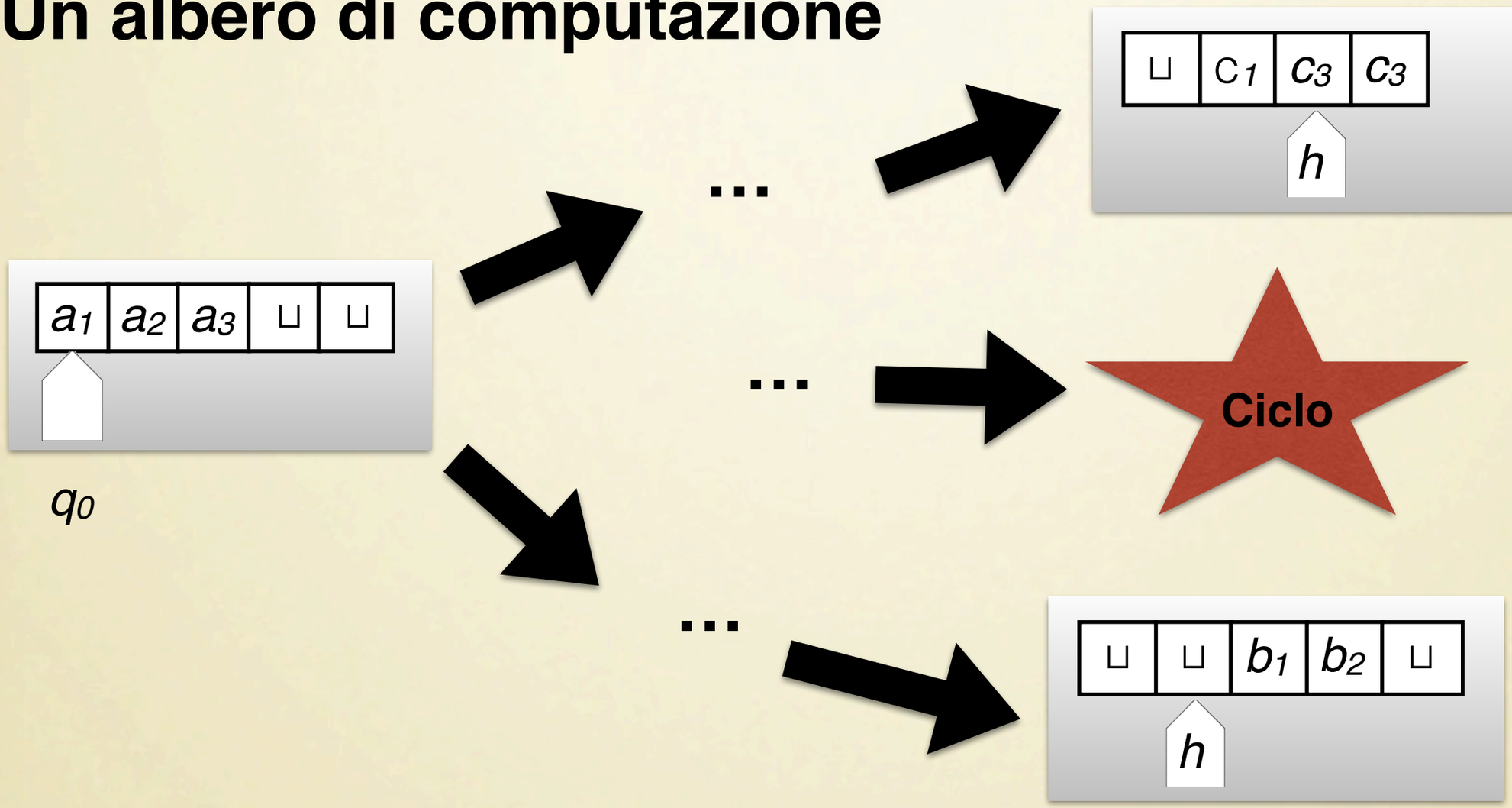
Perciò le configurazioni attraversate nel corso di una computazione non formano una **sequenza**, bensì un **albero**.

Ramificazioni = scelta non-deterministica

Un input é accettato se esiste un ramo accettante nell'albero.

Macchine di Turing non-deterministiche

Un albero di computazione



La TM accetta se almeno un ramo arriva ad uno stato finale

Macchine di Turing non-deterministiche: esempio

Diamo uno sketch della TM $\mathcal{M}$ non deterministica che decide il linguaggio dei numeri non-primi. Lavora su due nastri.

Il nastro 1 é di sola lettura e contiene il numero n di input.

in maniera non-deterministica, scegli m tale che 1 < m < n e scrivilo sul nastro 2.

Se n mod m = 0

Ferma e accetta

altrimenti

Ferma e rigetta

Perciò n é accettato se e solo se esiste un numero m tra 2 e n-1 che divide n.

Macchine di Turing non-deterministiche: equivalenza

Teorema Macchine di Turing e macchine di Turing non-deterministiche sono equivalenti.

Idea della dimostrazione

Data una TM $\mathcal{M}$ non-deterministica, costruiamo una TM $\mathcal{M}'$ (deterministica) equivalente.

L'idea é che $\mathcal{M}'$ su input ***x*** esplora tutto l'albero di computazione di $\mathcal{M}$ sullo stesso input ***x*** e accetta se almeno un ramo di computazione arriva ad uno stato finale.

Esercizio: depth-first o breadth-first? Perché?

Macchine di Turing non-deterministiche: equivalenza

Idea della dimostrazione

Per fare ciò, $\mathcal{M}'$ usa tre nastri.

a_1	a_2	a_3	⊔	⊔

Nastro per l'input
(sola lettura)

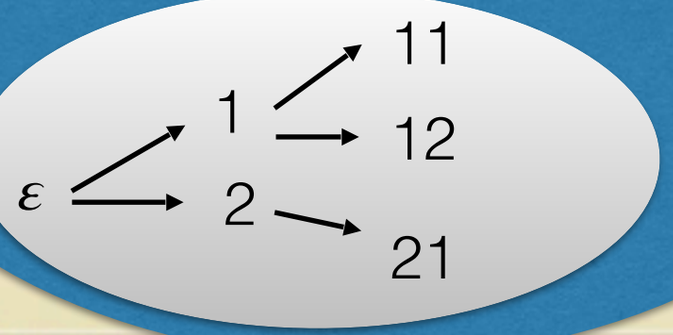
⊔	⊔	b_1	b_2	⊔

Nastro di simulazione
(esplora un ramo)

1	3	2	1

Nastro di indice
(tiene conto di quale ramo stiamo esaminando)

Ordine lessicografico sui nodi dell'albero di computazione di $\mathcal{M}$.



Ricapitolando

La definizione di macchina di Turing é ben lontana dall’essere arbitraria.

Anche se proviamo a ‘migliorarla' in vari modi (più nastri, non-determinismo, …), il potere espressivo del modello di calcolo rimane lo stesso.

Nella pratica, possiamo trarre vantaggio da queste variazioni per costruire più agilmente macchine di Turing che svolgano un determinato compito.

Case study:

proprietà di chiusura dei linguaggi

Proprietà di chiusura

Ora che abbiamo più libertà nel design delle nostre macchine di Turing, possiamo facilmente dimostrare alcune **proprietà di chiusura** della classi di linguaggi decidibili/riconoscibili da una macchina di Turing.

Proprietà di chiusura

Ricorda: i linguaggi sono insiemi.

Perciò, possiamo ragionare su operazioni come l'unione, l'intersezione, e il complemento di linguaggi (sullo stesso alfabeto Σ).

Il **complemento** L^- di un linguaggio L su Σ *é* $\Sigma^* \setminus L$.

Dal momento che i linguaggi sono insiemi di **stringhe**, abbiamo anche un'operazione di **concatenazione** tra linguaggi.

$$L_1L_2 = \{yz \mid y \in L_1 \text{ and } z \in L_2\}$$

Proprietà di chiusura

Prendiamo un insieme X di linguaggi.

Dimostrare che X é **chiuso rispetto all'unione** significa dimostrare che:

$$L_1 \in X \text{ e } L_2 \in X \quad \text{implica} \quad L_1 \cup L_2 \in X$$

Dimostrare che X é **chiuso rispetto al complemento** significa dimostrare che:

$$L \in X \quad \text{implica} \quad L^{-} \in X$$

E così via.

Ripasso: linguaggi decidibili

$\mathcal{M}$ **decide** un linguaggio L se:

- Quando $x \in L$, allora $\mathcal{M}$ accetta x (= ferma nello stato Y).
- Quando $x \notin L$, allora $\mathcal{M}$ rigetta x (= ferma nello stato N).

Proprietà di chiusura dei linguaggi decidibili

Teorema I linguaggi decidibili sono chiusi rispetto a:

complemento 

Idea della dimostrazione: scambia Y e N.

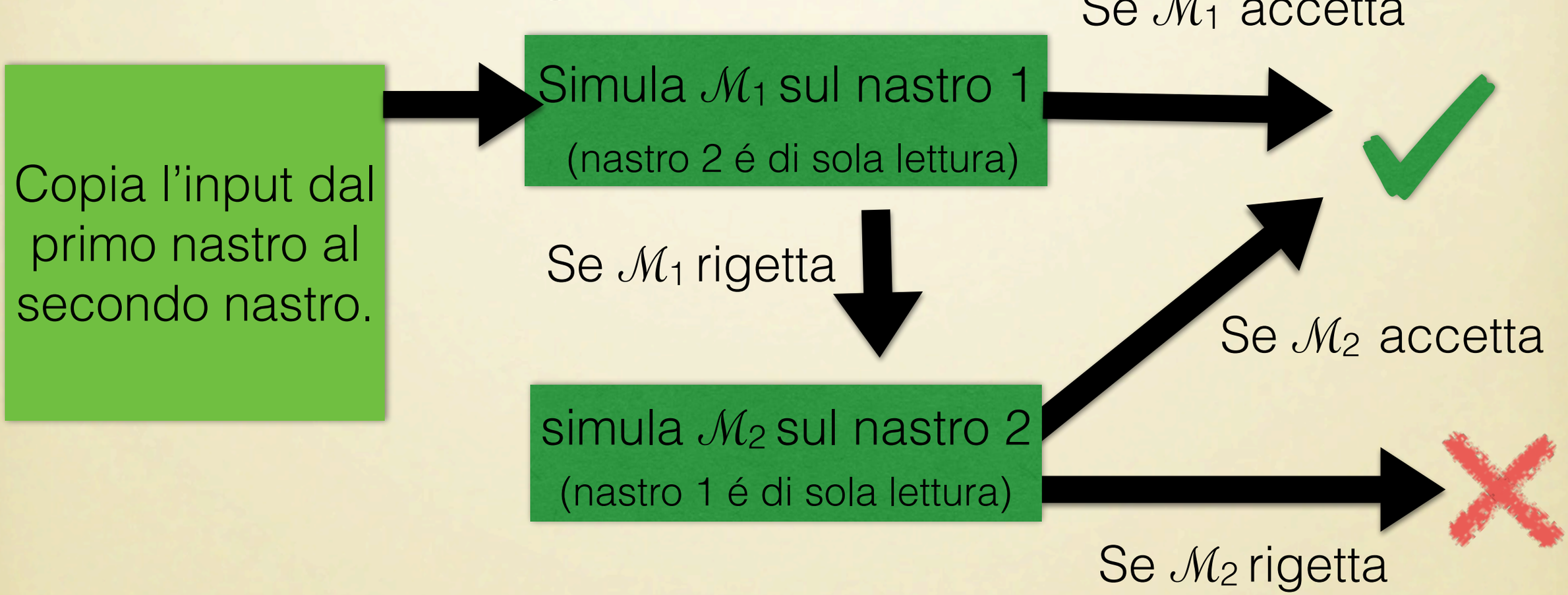
unione

intersezione

concatenazione

Chiusura rispetto a unione

Sia $\mathcal{M}_1$ una TM che decide L_1 e $\mathcal{M}_2$ una TM che decide L_2. La TM che decide $L_1 \cup L_2$ ha due nastri, ed é definita come segue.



(Sapreste dettagliare questa definizione?)

Proprietà di chiusura dei linguaggi decidibili

Teorema I linguaggi decidibili sono chiusi rispetto a:

complemento ✓

unione ✓

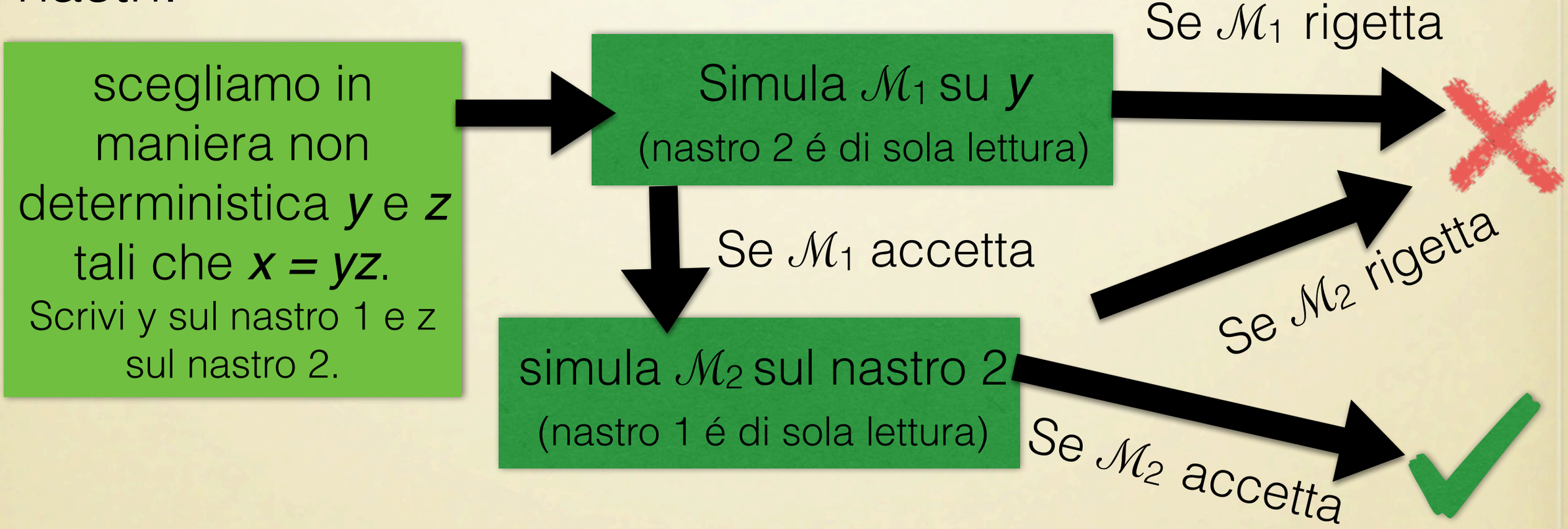
intersezione ✓ Esercizio

concatenazione

Chiusura rispetto a concatenazione

Promemoria: $L_1L_2 = \{yz \mid y \in L_1 \text{ and } z \in L_2\}$

Sia $\mathcal{M}_1$ una TM che decide L_1 e $\mathcal{M}_2$ una TM che decide L_2. La TM che decide L_1L_2 é non deterministica su due nastri.



Il non determinismo significa: se c'é almeno una decomposizione $x = yz$ tale che $y \in L_1$ e $z \in L_2$, allora x é accettato.

Proprietà di chiusura dei linguaggi decidibili

Teorema I linguaggi decidibili sono chiusi rispetto a:

complemento 

unione 

intersezione 

concatenazione 

Esercizio

Ripasso: linguaggi riconoscibili

$\mathcal{M}$ **riconosce** un linguaggio L se:

- Quando $x \in L$, allora $\mathcal{M}$ termina.
- Quando $x \notin L$, allora $\mathcal{M}$ non termina.

Proprietà di chiusura dei linguaggi riconoscibili

Teorema I linguaggi riconoscibili sono chiusi rispetto a:

unione

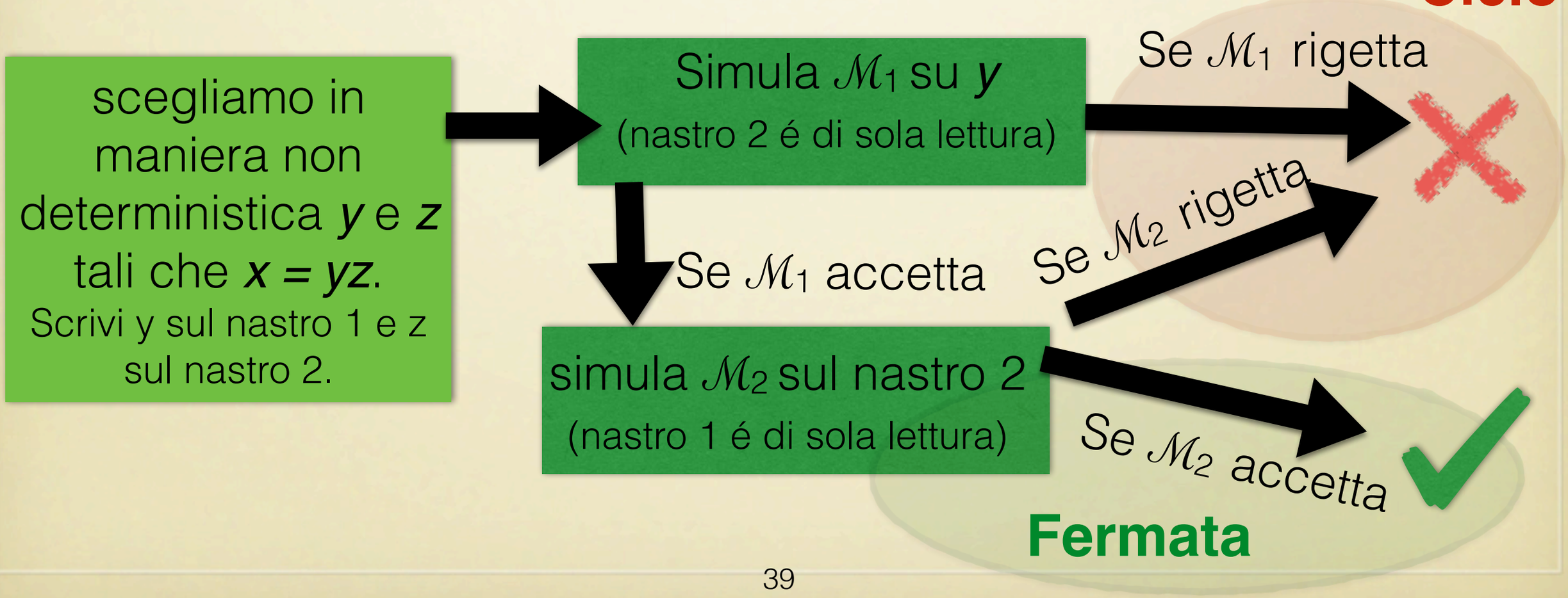
intersezione

concatenazione

Chiusura rispetto a concatenazione

Esercizio: possiamo riutilizzare la dimostrazione data per i linguaggi decidibili?

Sia $\mathcal{M}_1$ una TM che riconosce L_1 e $\mathcal{M}_2$ una TM che riconosce L_2. Costruiamo la TM che riconosce L_1L_2 :



Proprietà di chiusura dei linguaggi riconoscibili

Teorema I linguaggi riconoscibili sono chiusi rispetto a:

unione ✓

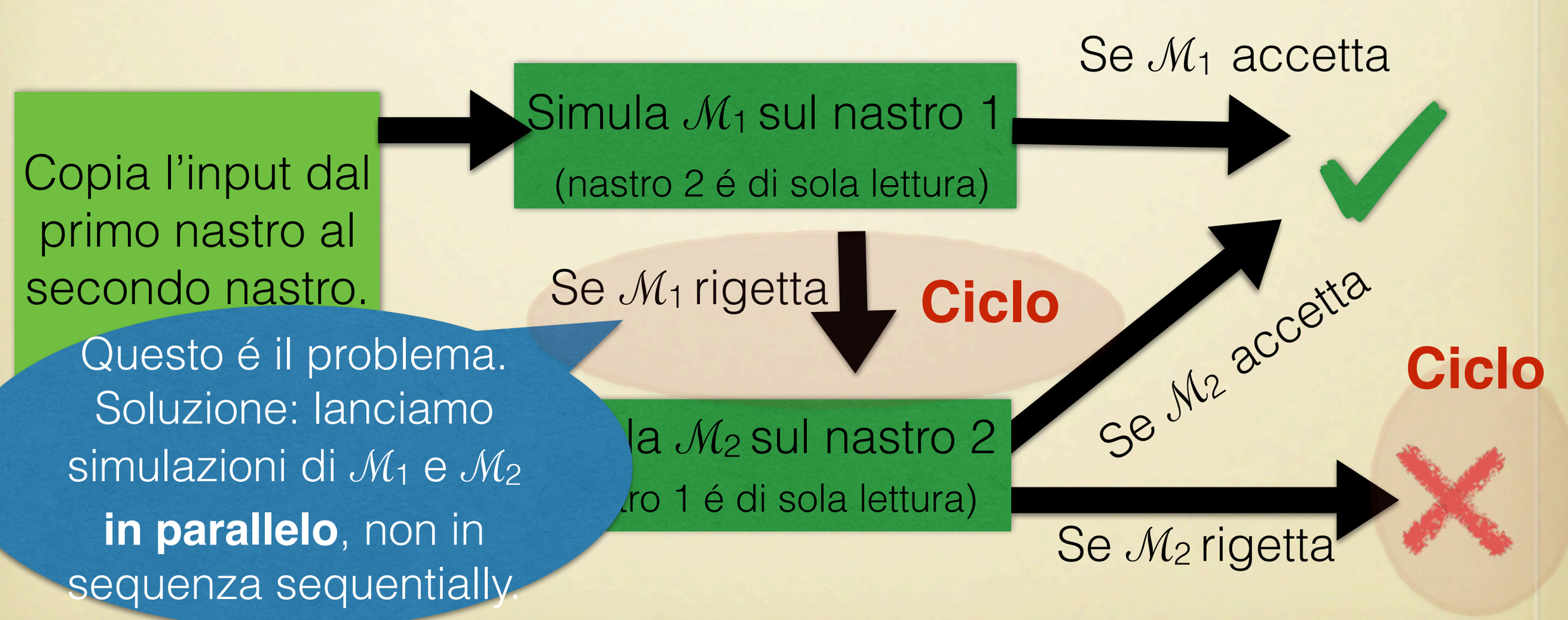
intersezione ✓ Esercizio

concatenazione

Chiusura rispetto a unione

Esercizio: possiamo riutilizzare la dimostrazione data per i linguaggi decidibili?

Sia $\mathcal{M}_1$ una TM che riconosce L_1 e $\mathcal{M}_2$ una TM che riconosce L_2. Costruiamo la TM che riconosce $L_1 \cup L_2$



Proprietà di chiusura dei linguaggi riconoscibili

Teorema I linguaggi riconoscibili sono chiusi rispetto a:

unione ✓

intersezione ✓

concatenazione ✓

Esercizio

...e il complemento?

Proprietà di chiusura dei linguaggi riconoscibili

...e il complemento?

Al contrario dei linguaggi decidibili, i linguaggi riconoscibili **non** sono chiusi rispetto al complemento. Vedremo la dimostrazione più avanti, dal momento che segue dall'esistenza di un linguaggio riconoscibile che non é decidibile.

Una tecnica utile

Possiamo fare leva sulle proprietà di chiusura per semplificare le dimostrazioni che un certo linguaggio é decidibile (o riconoscibile).

Per esempio, al fine di dimostrare che il linguaggio

$$L = \{x \in \{0,1\}^* \mid x \text{ ha lunghezza dispari e più 1 che 0}\}$$

é decidibile, basta dimostrare che

$$L_1 = \{x \in \{0,1\}^* \mid x \text{ ha lunghezza dispari}\}$$

$$L_2 = \{x \in \{0,1\}^* \mid \text{in } x \text{ ci sono più 1 che 0}\}$$

sono decidibili, perché $L = L_1 \cap L_2$.