Computabilità, Complessità e Logica

Lezione 15

Complementare i problemi di decisione

- Dato un linguaggio L possiamo studiare il linguaggio $\bar{L} = \Sigma^\star - L$, ovvero il complementare di L
- Abbiamo che $w \in L \iff w \notin \bar{L}$
- Dato una classe di linguaggi $\mathscr{C}$ ci chiediamo se essa si chiusa rispetto alla complementazione, ovvero se $L \in \mathscr{C} \iff \bar{L} \in \mathscr{C}$
- Sia $\mathrm{co}\mathscr{P}$ la classe $\mathrm{co}\mathscr{C} = \{\bar{L} \mid L \in \mathscr{C}\}$, se $\mathscr{C}$ è chiusa rispetto alla complementazione, allora $\mathrm{co}\mathscr{C} = \mathscr{C}$

Complementazione per MdT deterministiche

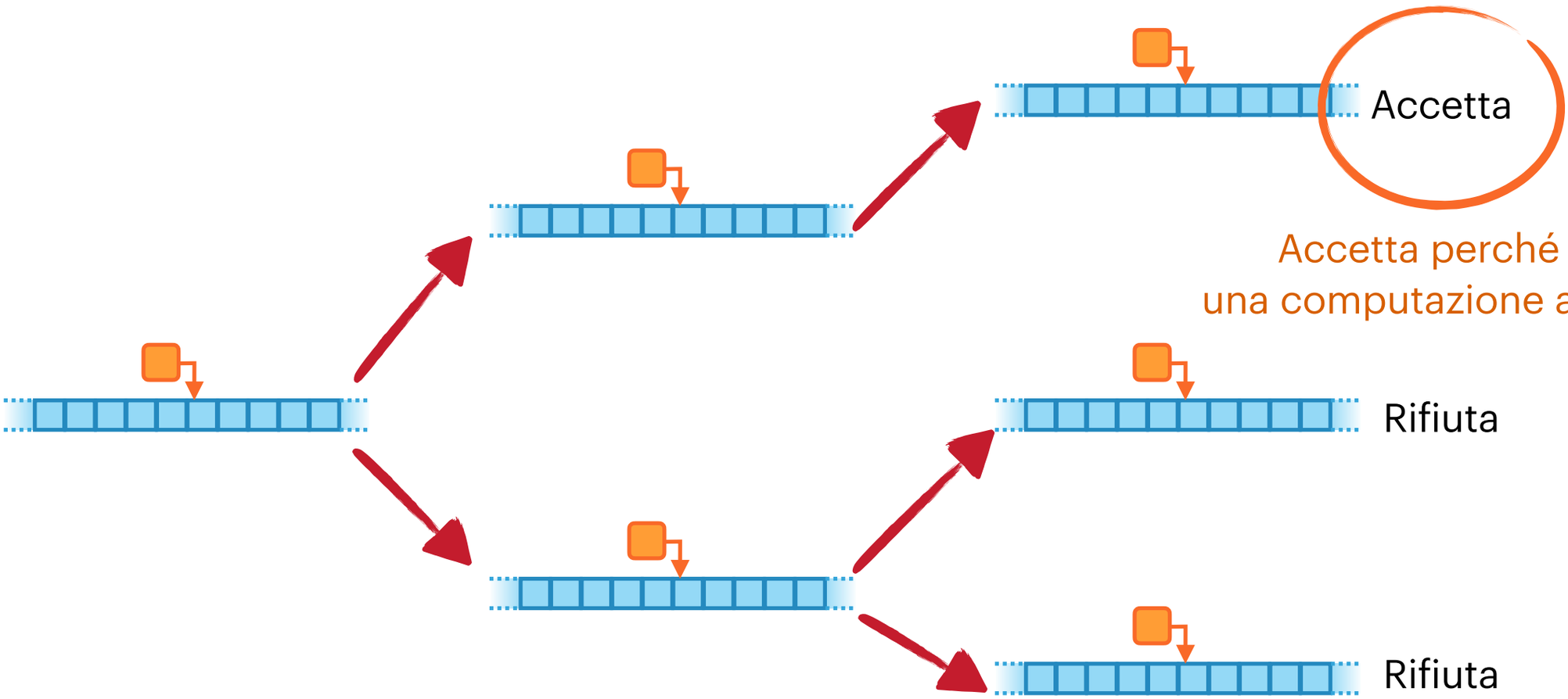
- Consideriamo un problema $L \in \mathrm{P}$
- Se L è in P allora esiste una MdT deterministica $M_L = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ che in tempo polinomiale accetta le parole in L e rifiuta le parole che non appartengono a L
- Se invertiamo gli stati accettante e rifiutante di M_L allora accettiamo le parole non in L e rifiutiamo quelle in L: abbiamo ottenuto una macchina che decide $\bar{L}$ in tempo polinomiale
- Quindi abbiamo $\mathrm{P} = \mathrm{coP}$

Complementazione per MdT non-deterministiche

- Consideriamo ora $L \in \mathrm{NP}$
- Possiamo fare la stessa costruzione usata per le macchine deterministiche?
- Se invertiamo il ruolo di stato accettante e rifiutante...
- ...otteniamo che passiamo da “esiste uno stato accettante” a “esiste uno stato rifiutante”
- Ma questo non è abbastanza per dire che rifiutiamo invece di accettare!

Complementazione per MdT non-deterministiche

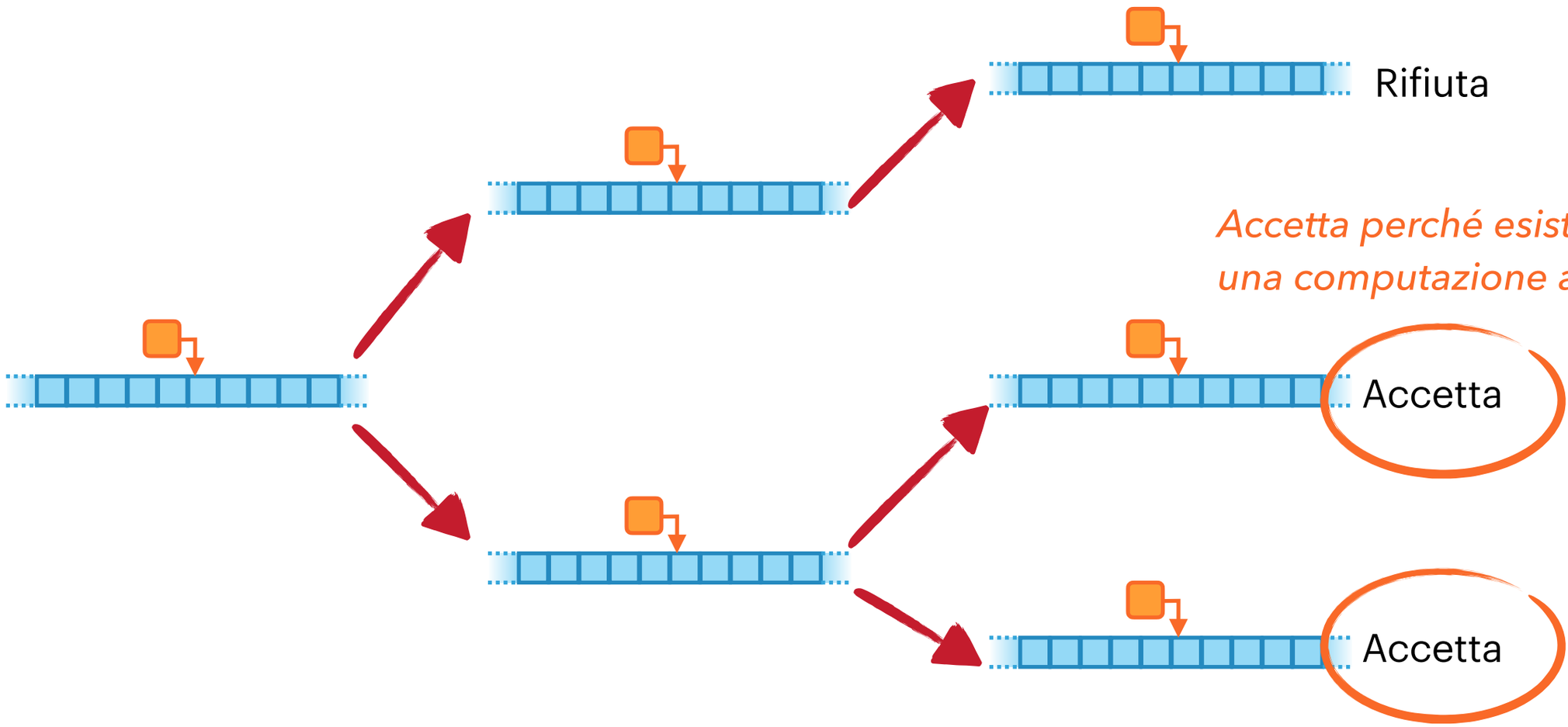
Albero di computazioni di una macchina non deterministica



Proviamo ad invertire il ruolo di stati accettanti e rifiutanti

Complementazione per MdT non-deterministiche

Albero di computazioni di una macchina non deterministica



Per rifiutare ci serve che **tutte** le computazioni siano rifiutanti!

Complementazione per MdT non-deterministiche

- Per ottenere il comportamento corretto
  (i.e., $w \in L \iff w \notin \overline{L}$) serve una condizione di accettazione diversa

- Serve che **tutte** le computazioni di una macchina non-deterministica accettino

- Macchina che decide L:
  $w \in L$ se su input w **esiste** una computazione accettante

- Macchina che decide $\overline{L}$:
  $w \in \overline{L}$ se su input w **ogni** computazione è accettante

coNP

- Non è immediato che il complementare di un problema in NP sia anch'esso in NP
- La classe dei problemi complementari a quelli in NP è chiamata coNP
- Si ha $\mathrm{P} \subseteq \mathrm{coNP}$, dato che quando complimentiamo i problemi in $\mathrm{P} \subseteq \mathrm{NP}$ riotteniamo la stessa classe di problemi
- Non è noto se $\mathrm{coNP} = \mathrm{NP}$

coNP

- Se $\mathrm{P} = \mathrm{NP}$ avremmo immediatamente $\mathrm{coNP} = \mathrm{NP} = \mathrm{P}$ dato che P è chiuso rispetto alla complementazione
- Se $\mathrm{coNP} \neq \mathrm{NP}$ avremmo immediatamente $\mathrm{P} \neq \mathrm{NP}$ perché non rispetteremmo la chiusura rispetto alla complementazione di P
- Ma potremmo avere $\mathrm{NP} = \mathrm{coNP}$ anche se $\mathrm{P} \neq \mathrm{NP}$

coNP

- $\mathrm{P} \subseteq \mathrm{NP} \cap \mathrm{coNP}$, ma ci sono anche problemi che sappiamo essere in $\mathrm{NP} \cap \mathrm{coNP}$ e che non sappiamo essere in P:
  - **Fattorizzazione**: dati $x, a, b \in \mathbb{N}$ decidere se esiste un primo $p \in [a, b]$ che divide x
- Come NP, anche coNP ammette problemi completi
- Un esempio di problema coNP-completo:
  - **Tautology/validity**: Stabilire se una formula è una tautologia, ovvero tutti gli assegnamenti la soddisfano

Problemi NP-completi

- Clique
- Vertex Cover
- Travelling Salesman Problem
  (problema del commesso viaggiatore)
- Subset Sum
- Problema dello zaino