Logica e Modelli Computazionali

Complessità Computazionale

Marco Console

Ingegneria Informatica e Automatica (Sapienza, Univerità di Roma)

Limiti Pratici

- Abbiamo discusso la **teoria della computabilità** e costruito funzioni non Computabili
  - Non esiste cioè una macchina di Turing in grado di calcolare tale funzioni
  - Nessun modello computazionale noto fa meglio della MT da questo punto di vista

- La **Turing Computabilità** non assicura però una soluzione pratica ad un problema
  - Il miglior algoritmo possibile potrebbe impiegare moltissimo tempo per risolverlo
  - Il miglior algoritmo possibile potrebbe richiedere moltissima memoria per risolverlo

- Per studiare quest'ulteriore aspetto della natura dei problemi computazionali ci rivolgiamo alla **teoria della complessità computazionale**

- Tenteremo (non sempre riuscendoci) di rispondere alla seguente domanda

Quante risorse sono necessarie per risolvere un problema computazionale?

Risorse – Tempo e Spazio

- Dobbiamo per prima cosa definire quali sono le risorse a cui siamo interessati
  - Considereremo le risorse **Tempo** e **Spazio** come segue

- **Tempo:** Quanto tempo impiega una macchina per risolvere il problema ?
  - Anche se un problema è Turing Computabile, se la macchina che lo risolve ha bisogno di mesi, anni o secoli per terminare, la computabilità non ci aiuta affatto

- **Spazio:** Quanta memoria utilizza la macchina per risolvere il problema?
  - Anche se un problema è Turing Computabile, se la macchina che lo risolve ha bisogno di milioni o miliardi di Terabyte di memoria, non riusciremo a lanciare la nostra soluzione su nessun calcolatore reale

- Come per la teoria della computabilità noi considereremo Macchine di Turing
  - Per una certa funzione f, quanto tempo impiega una MT che calcola f a terminare?
  - Per una certa funzione f, quanto nastro impiega una MT che calcola f a terminare?

Complessità del Caso Peggiore (1/3)

- Ovviamente diversi input potrebbero dar vita a diversi comportamenti di un MT
  - Con un input (**difficile da valutare**) una macchina potrebbe dover consumare tantissime risorse
  - Con un altro (**facile da valutare**) input la stessa macchina potrebbe usare pochissime risorse

- Per capire quanto un input è "difficile" **ci baseremo sulla sua dimensione**
  - Più celle di nastro occupa più lo riteniamo complesso
  - Anche se potrebbero esserci altri parametri, questa scelta è ragionevole perché, per problemi reali, una MT deve almeno controllare tutto o quasi l'input prima di decidere se accettare

- Ovviamente, la quantità di risorse richieste da un macchina varia per input della stessa dimensione
  - Un input σ di dimensione n potrebbe essere **difficile da valutare**
  - Un altro input τ di dimensione n potrebbe essere **facile da valutare**

- Noi considereremo le risorse nel **caso peggiore**, cioè per ogni possibile dimensione $n \in \mathbb{N}$ dell'input, considereremo l'input di dimensione n che da vita al consumo di risorse maggiore

Complessità del Caso Peggiore (2/3)

- La teoria della complessità computazionale parte dalle premesse che abbiamo identificato

- Per la **risorsa Tempo** impiegata da una Macchia di Turing M
  - Assumiamo che ogni **cambio di configurazione di M** costi 1 unità (nessuna in particolare)
  - Definiamo una funzione matematica $f_M: \mathbb{N} \to \mathbb{N}$ che, per ogni $n \in \mathbb{N}$ **restituisce il numero di configurazioni che la macchina deve attraversare** prima di terminare per l'input di dimensione n che richiede **il numero maggiore di configurazioni**
  - Studiamo l'andamento asintotico di tale funzione f_M

- Per la **risorsa Spazio** impiegata da una Macchia di Turing M
  - Assumiamo che ogni **cella del nastro di M** costi 1 unità (nessuna in particolare)
  - Definiamo una funzione matematica $g_M: \mathbb{N} \to \mathbb{N}$ che, per ogni $n \in \mathbb{N}$ **restituisce il massimo numero di celle scritte dalla macchina** durante una esecuzione per un input di dimensione n
  - Studiamo l'andamento asintotico di tale funzione g_M

Complessità del Caso Peggiore (3/3)

- Ovviamente queste sono analisi "poco raffinate"
  - L'andamento asintotico di una funzione perde molta informazione

- Ci danno però una indicazione di quanto tempo e spazio impieghi la macchina.
  - In particolar modo se ci accorgiamo che l'andamento asintotico di f_M o g_M è quello di una funzione la cui forma chiusa è nota

- Supponiamo, ad esempio, che un passo di una MT possa essere eseguito in $1\mu s$

$f_M(n)$	$n = 10$	$n = 20$	$n = 50$	$n = 100$	$n = 1000$
n	$10\mu s$	$20\mu s$	$50\mu s$	$100\mu s$	$1ms$
$n * \log_2 n$	$33.2\mu s$	$86.4\mu s$	$0.28ms$	$0.6ms$	$9.9ms$
n^2	$0.1ms$	$0.4ms$	$2.5ms$	$10ms$	$1s$
n^3	$1ms$	$8ms$	$125ms$	$1s$	$16.6min$
2^n	$1ms$	$1s$	$35.7a$	$\approx 10^{14}\ a$	??

Notazione O-Grande

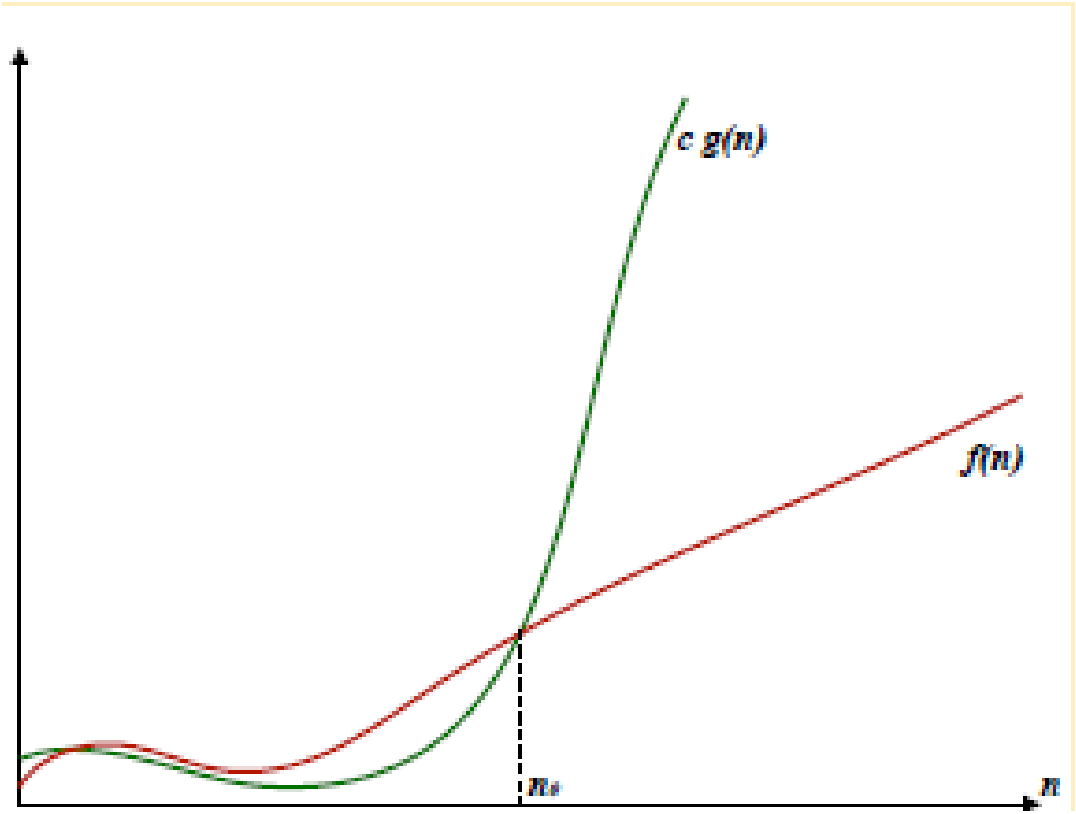
Notazione O-Grande

- Per descrivere il comportamento asintotico di una funzione utilizziamo la notazione O-grande
- **Definizione**. Siano $f: \mathbb{N} \to \mathbb{N}$ e $g: \mathbb{N} \to \mathbb{N}$ due funzioni.
- **Definizione**. $f(n) = \mathcal{O}(g(n))$ ($f(n)$ è un **O-grande** di $g(n)$) se esistono due costanti $c > 0$ e $n_0 \geq 0$ tali che

$$\boldsymbol{f(n) \leq c \cdot g(n)}\textbf{, per ogni } \boldsymbol{n \geq n_0}$$

- **Intuitivamente**, $f(n) = \mathcal{O}(g(n))$ se esiste un certo valore di n (n_0) dopo il quale il valore $f(n)$ è più piccolo di $c \cdot g(n)$.
- **In altre parole**, $f(n)$ cresce al più quanto $g(n)$, a partire da un certo valore di n

Notazione O Grande – Esempio Grafico



Alcune Regole Utili

- Assumiamo tre funzioni $f(n)$, $g(n)$ e $h(n)$

1. $f(n) + g(n) = \mathcal{O}(\max\{f(n), g(n)\})$
   - L'andamento asintotico di una somma dipende dal termine più veloce (**dominante**)

2. $f(n) \cdot g(n) = \mathcal{O}(f(n) \cdot g(n))$
   - L'andamento asintotico di un prodotto dipende da entrambi gli operandi

3. Se $f(n) = \mathcal{O}(g(n))$ e $g(n) = \mathcal{O}(h(n))$ allora $f(n) = \mathcal{O}(h(n))$
   - L'andamento asintotico è transitivo

Alcuni Andamenti Asintotici Utili

- **Costanti**. Per ogni costante c, $\boldsymbol{c = \mathcal{O}(g(n))}$**, per ogni** $\boldsymbol{g(n)}$
  - Funzioni costanti non crescono al variare di n
- **Polinomi**. Sia $p(n) = c_0 + c_1\, n^1 + \cdots + c_m\, n^m$ un polinomio di grado m. $\boldsymbol{p(n) = \mathcal{O}(n^m)}$
  - Un polinomio cresce asintoticamente come il suo coefficiente di grado maggiore
- **Logaritmi**. Sia $p(n)$ un polinomio. $\boldsymbol{\log(n) = \mathcal{O}(p(n))}$
  - Il logaritmo cresce asintoticamente meno di un polinomio
- **Esponenziali**. Sia $p(n)$ un polinomio. $\boldsymbol{p(n) = \mathcal{O}(t^n)}$**, per ogni** $\boldsymbol{t > 1}$
  - Un polinomio cresce asintoticamente meno di un esponenziale

Esempi di Crescita

	constant	*logarithmic*	*linear*	*N-log-N*	*quadratic*	*cubic*	*exponential*
n	$O(1)$	$O(\log n)$	$O(n)$	$O(n \log n)$	$O(n^2)$	$O(n^3)$	$O(2^n)$
1	1	1	1	1	1	1	2
2	1	1	2	2	4	8	4
4	1	2	4	8	16	64	16
8	1	3	8	24	64	512	256
16	1	4	16	64	256	4,096	65536
32	1	5	32	160	1,024	32,768	4,294,967,296
64	1	6	64	384	4,069	262,144	1.84×10^{19}

Esempi di Crescita – In Prospettiva

- $n = 300$
  - Iscritti alla triennale di ingegneria informatica ogni anno (circa)
- $n^2 = 90000$
  - Studenti iscritti a Sapienza 115000 (la più grande in Europa per numero di studenti)
- $n^3 = 27000000$ (**27 milioni**)
  - Cittadini Italiani circa 60000000
- $n^4 = 8100000000$ (**8 miliardi 100 milioni**)
  - Popolazione mondiale circa 8000000000 (8 miliardi)
- $2^n > 2 \cdot 10^{90}$ (**un bel po' ☺**)
  - Atomi nell'universo $\leq 10^{82}$

Complessità Temporale

Complessità Temporale

- Sia $M = \langle \Sigma, \Gamma, Q, \delta, q_0, q_{yes}, q_{no} \rangle$ una Macchina di Turing terminante con k nastri e x un input per M

- **Definizione**. **L'esecuzione di M** con input x ($E_M(x)$) è la sequenza di configurazioni di M $C_1, C_2, \dots, C_n$ tale che
  - C_1 è la configurazione iniziale di M con input x
  - C_n è una configurazione **accettante\rifiutante**
  - $C_i \Rightarrow_M C_{i+1}$ per ogni $i = 1, \dots, n-1$

- **Definizione**. La **complessità temporale** $f_M: \mathbb{N} \to \mathbb{N}$ di M è la funzione $f_M(n) = \max_{\{x \in \Sigma^* \mid |x| = n\}} |E_M(x)|$
  - $f_M(n)$ è la lunghezza della più lunga esecuzione di M con input σ tale che $|\sigma| = n$

- **Definizione**. Sia $f: \mathbb{N} \to \mathbb{N}$ una funzione. $TIME(g)$ è la famiglia dei linguaggi che possono essere decisi da una Macchina di Turing M la cui **complessità temporale** f_M è un O-grande di g
  - Problemi che possono essere risolti con una macchina il cui tempo di esecuzione cresce al più quanto g

Complessità Temporale – Esempio

- Consideriamo una macchina M che riconosce il linguaggio $\mathrm{PAL}_{\{a,b\}} = \{w \in \{a,b\}^* \mid w = \overline{w}\}$

(q,x)	$\delta(q,x)$
(q_0,a)	$(q_a,\#,\rightarrow)$
(q_0,b)	$(q_b,\#,\rightarrow)$
$(q_0,\#)$	$(q_0,\#,\rightarrow)$
$(q_0,\sqcup)$	$(q_{yes},\sqcup,\rightarrow)$
$(q_a,\sqcup)$	$(q'_a,\sqcup,\leftarrow)$
$(q_a,x) \mid x \neq \sqcup$	$(q_a,x,\rightarrow)$
$(q_b,\sqcup)$	$(q'_b,\sqcup,\leftarrow)$
$(q_b,x) \mid x \neq \sqcup$	$(q_b,x,\rightarrow)$
$(q'_a,x) \mid x \in \{a,\#\}$	$(q_{ret},\sqcup,\leftarrow)$
$(q'_a,x) \mid x \notin \{a,\#\}$	$(q_{no},x,-)$
$(q'_b,x) \mid x \in \{b,\#\}$	$(q_{ret},\sqcup,\leftarrow)$
$(q'_b,x) \mid x \notin \{b,\#\}$	$(q_{no},x,-)$
$(q_{ret},\#)$	$(q_0,\#,\rightarrow)$
$(q_{ret},x) \mid x \neq \#$	$(q_{ret},x,\leftarrow)$

Complessità Temporale – Esempio

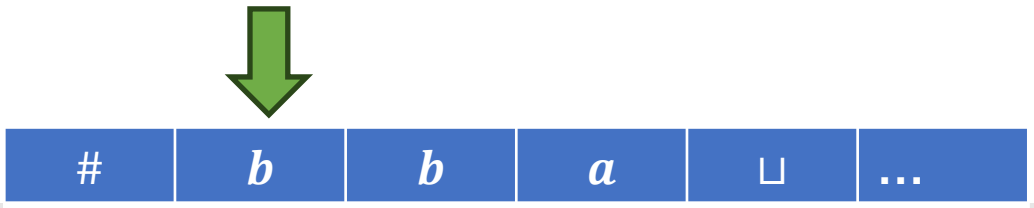
(q, x)	$\delta(q, x)$
(q_0, a)	$(q_a, \#, \rightarrow)$
(q_0, b)	$(q_b, \#, \rightarrow)$
$(q_0, \#)$	$(q_0, \#, \rightarrow)$
$(q_0, \sqcup)$	$(q_{yes}, \sqcup, \rightarrow)$
$(q_a, \sqcup)$	$(q'_a, \sqcup, \leftarrow)$
$(q_a, x) \mid x \neq \sqcup$	$(q_a, x, \rightarrow)$
$(q_b, \sqcup)$	$(q'_b, \sqcup, \leftarrow)$
$(q_b, x) \mid x \neq \sqcup$	$(q_b, x, \rightarrow)$
$(q'_a, x) \mid x \in \{a, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_a, x) \mid x \notin \{a, \#\}$	$(q_{no}, x, -)$
$(q'_b, x) \mid x \in \{b, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_b, x) \mid x \notin \{b, \#\}$	$(q_{no}, x, -)$
$(q_{ret}, \#)$	$(q_0, \#, \rightarrow)$
$(q_{ret}, x) \mid x \neq \#$	$(q_{ret}, x, \leftarrow)$

Stato
q_0

Complessità Temporale – Esempio

(q, x)	$\delta(q, x)$
(q_0, a)	$(q_a, \#, \rightarrow)$
(q_0, b)	$(q_b, \#, \rightarrow)$
$(q_0, \#)$	$(q_0, \#, \rightarrow)$
$(q_0, \sqcup)$	$(q_{yes}, \sqcup, \rightarrow)$
$(q_a, \sqcup)$	$(q'_a, \sqcup, \leftarrow)$
$(q_a, x) \mid x \neq \sqcup$	$(q_a, x, \rightarrow)$
$(q_b, \sqcup)$	$(q'_b, \sqcup, \leftarrow)$
$(q_b, x) \mid x \neq \sqcup$	$(q_b, x, \rightarrow)$
$(q'_a, x) \mid x \in \{a, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_a, x) \mid x \notin \{a, \#\}$	$(q_{no}, x, -)$
$(q'_b, x) \mid x \in \{b, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_b, x) \mid x \notin \{b, \#\}$	$(q_{no}, x, -)$
$(q_{ret}, \#)$	$(q_0, \#, \rightarrow)$
$(q_{ret}, x) \mid x \neq \#$	$(q_{ret}, x, \leftarrow)$

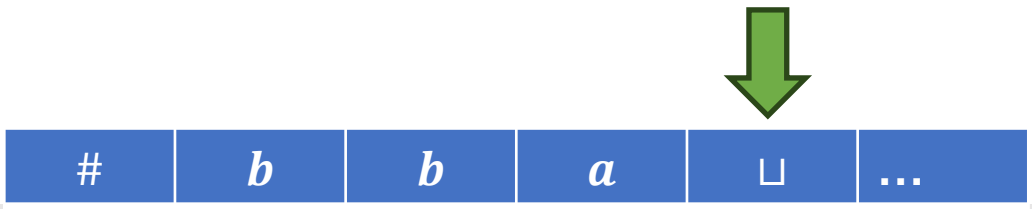
Stato
q_a



Complessità Temporale – Esempio

(q, x)	$\delta(q, x)$
(q_0, a)	$(q_a, \#, \rightarrow)$
(q_0, b)	$(q_b, \#, \rightarrow)$
$(q_0, \#)$	$(q_0, \#, \rightarrow)$
$(q_0, \sqcup)$	$(q_{yes}, \sqcup, \rightarrow)$
$(q_a, \sqcup)$	$(q'_a, \sqcup, \leftarrow)$
$(q_a, x) \mid x \neq \sqcup$	$(q_a, x, \rightarrow)$
$(q_b, \sqcup)$	$(q'_b, \sqcup, \leftarrow)$
$(q_b, x) \mid x \neq \sqcup$	$(q_b, x, \rightarrow)$
$(q'_a, x) \mid x \in \{a, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_a, x) \mid x \notin \{a, \#\}$	$(q_{no}, x, -)$
$(q'_b, x) \mid x \in \{b, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_b, x) \mid x \notin \{b, \#\}$	$(q_{no}, x, -)$
$(q_{ret}, \#)$	$(q_0, \#, \rightarrow)$
$(q_{ret}, x) \mid x \neq \#$	$(q_{ret}, x, \leftarrow)$

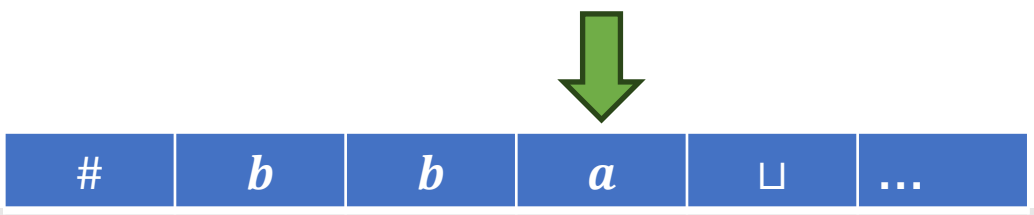
Stato
q_a



Complessità Temporale – Esempio

(q, x)	$\delta(q, x)$
(q_0, a)	$(q_a, \#, \rightarrow)$
(q_0, b)	$(q_b, \#, \rightarrow)$
$(q_0, \#)$	$(q_0, \#, \rightarrow)$
$(q_0, \sqcup)$	$(q_{yes}, \sqcup, \rightarrow)$
$(q_a, \sqcup)$	$(q'_a, \sqcup, \leftarrow)$
$(q_a, x) \mid x \neq \sqcup$	$(q_a, x, \rightarrow)$
$(q_b, \sqcup)$	$(q'_b, \sqcup, \leftarrow)$
$(q_b, x) \mid x \neq \sqcup$	$(q_b, x, \rightarrow)$
$(q'_a, x) \mid x \in \{a, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_a, x) \mid x \notin \{a, \#\}$	$(q_{no}, x, -)$
$(q'_b, x) \mid x \in \{b, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_b, x) \mid x \notin \{b, \#\}$	$(q_{no}, x, -)$
$(q_{ret}, \#)$	$(q_0, \#, \rightarrow)$
$(q_{ret}, x) \mid x \neq \#$	$(q_{ret}, x, \leftarrow)$

Stato
q'_a



Complessità Temporale – Esempio

(q, x)	$\delta(q, x)$
(q_0, a)	$(q_a, \#, \rightarrow)$
(q_0, b)	$(q_b, \#, \rightarrow)$
$(q_0, \#)$	$(q_0, \#, \rightarrow)$
$(q_0, \sqcup)$	$(q_{yes}, \sqcup, \rightarrow)$
$(q_a, \sqcup)$	$(q'_a, \sqcup, \leftarrow)$
$(q_a, x) \mid x \neq \sqcup$	$(q_a, x, \rightarrow)$
$(q_b, \sqcup)$	$(q'_b, \sqcup, \leftarrow)$
$(q_b, x) \mid x \neq \sqcup$	$(q_b, x, \rightarrow)$
$(q'_a, x) \mid x \in \{a, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_a, x) \mid x \notin \{a, \#\}$	$(q_{no}, x, -)$
$(q'_b, x) \mid x \in \{b, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_b, x) \mid x \notin \{b, \#\}$	$(q_{no}, x, -)$
$(q_{ret}, \#)$	$(q_0, \#, \rightarrow)$
$(q_{ret}, x) \mid x \neq \#$	$(q_{ret}, x, \leftarrow)$

Stato
q_{ret}

#	b	b	⊔	⊔	…

Complessità Temporale – Esempio

(q, x)	$\delta(q, x)$
(q_0, a)	$(q_a, \#, \rightarrow)$
(q_0, b)	$(q_b, \#, \rightarrow)$
$(q_0, \#)$	$(q_0, \#, \rightarrow)$
$(q_0, \sqcup)$	$(q_{yes}, \sqcup, \rightarrow)$
$(q_a, \sqcup)$	$(q'_a, \sqcup, \leftarrow)$
$(q_a, x) \mid x \neq \sqcup$	$(q_a, x, \rightarrow)$
$(q_b, \sqcup)$	$(q'_b, \sqcup, \leftarrow)$
$(q_b, x) \mid x \neq \sqcup$	$(q_b, x, \rightarrow)$
$(q'_a, x) \mid x \in \{a, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_a, x) \mid x \notin \{a, \#\}$	$(q_{no}, x, -)$
$(q'_b, x) \mid x \in \{b, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_b, x) \mid x \notin \{b, \#\}$	$(q_{no}, x, -)$
$(q_{ret}, \#)$	$(q_0, \#, \rightarrow)$
$(q_{ret}, x) \mid x \neq \#$	$(q_{ret}, x, \leftarrow)$

Stato
q_{ret}

Complessità Temporale – Esempio

(q, x)	$\delta(q, x)$
(q_0, a)	$(q_a, \#, \rightarrow)$
(q_0, b)	$(q_b, \#, \rightarrow)$
$(q_0, \#)$	$(q_0, \#, \rightarrow)$
$(q_0, \sqcup)$	$(q_{yes}, \sqcup, \rightarrow)$
$(q_a, \sqcup)$	$(q'_a, \sqcup, \leftarrow)$
$(q_a, x) \mid x \neq \sqcup$	$(q_a, x, \rightarrow)$
$(q_b, \sqcup)$	$(q'_b, \sqcup, \leftarrow)$
$(q_b, x) \mid x \neq \sqcup$	$(q_b, x, \rightarrow)$
$(q'_a, x) \mid x \in \{a, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_a, x) \mid x \notin \{a, \#\}$	$(q_{no}, x, -)$
$(q'_b, x) \mid x \in \{b, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_b, x) \mid x \notin \{b, \#\}$	$(q_{no}, x, -)$
$(q_{ret}, \#)$	$(q_0, \#, \rightarrow)$
$(q_{ret}, x) \mid x \neq \#$	$(q_{ret}, x, \leftarrow)$

Stato
q_0

#	*b*	*b*	⊔	⊔	…

Complessità Temporale – Esempio

(q, x)	$\delta(q, x)$
(q_0, a)	$(q_a, \#, \rightarrow)$
(q_0, b)	$(q_b, \#, \rightarrow)$
$(q_0, \#)$	$(q_0, \#, \rightarrow)$
$(q_0, \sqcup)$	$(q_{yes}, \sqcup, \rightarrow)$
$(q_a, \sqcup)$	$(q'_a, \sqcup, \leftarrow)$
$(q_a, x) \mid x \neq \sqcup$	$(q_a, x, \rightarrow)$
$(q_b, \sqcup)$	$(q'_b, \sqcup, \leftarrow)$
$(q_b, x) \mid x \neq \sqcup$	$(q_b, x, \rightarrow)$
$(q'_a, x) \mid x \in \{a, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_a, x) \mid x \notin \{a, \#\}$	$(q_{no}, x, -)$
$(q'_b, x) \mid x \in \{b, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_b, x) \mid x \notin \{b, \#\}$	$(q_{no}, x, -)$
$(q_{ret}, \#)$	$(q_0, \#, \rightarrow)$
$(q_{ret}, x) \mid x \neq \#$	$(q_{ret}, x, \leftarrow)$

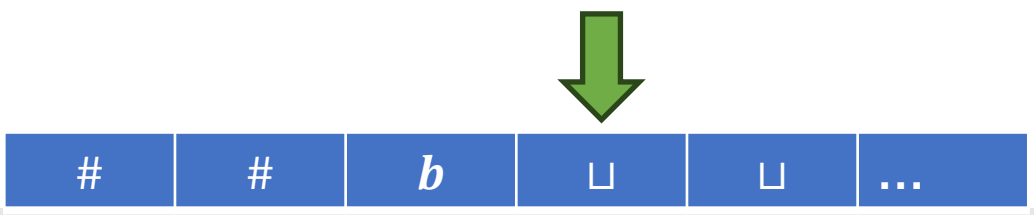
Stato
q_b

#	#	b	⊔	⊔	…

Complessità Temporale – Esempio

(q, x)	$\delta(q, x)$
(q_0, a)	$(q_a, \#, \rightarrow)$
(q_0, b)	$(q_b, \#, \rightarrow)$
$(q_0, \#)$	$(q_0, \#, \rightarrow)$
$(q_0, \sqcup)$	$(q_{yes}, \sqcup, \rightarrow)$
$(q_a, \sqcup)$	$(q'_a, \sqcup, \leftarrow)$
$(q_a, x) \mid x \neq \sqcup$	$(q_a, x, \rightarrow)$
$(q_b, \sqcup)$	$(q'_b, \sqcup, \leftarrow)$
$(q_b, x) \mid x \neq \sqcup$	$(q_b, x, \rightarrow)$
$(q'_a, x) \mid x \in \{a, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_a, x) \mid x \notin \{a, \#\}$	$(q_{no}, x, -)$
$(q'_b, x) \mid x \in \{b, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_b, x) \mid x \notin \{b, \#\}$	$(q_{no}, x, -)$
$(q_{ret}, \#)$	$(q_0, \#, \rightarrow)$
$(q_{ret}, x) \mid x \neq \#$	$(q_{ret}, x, \leftarrow)$

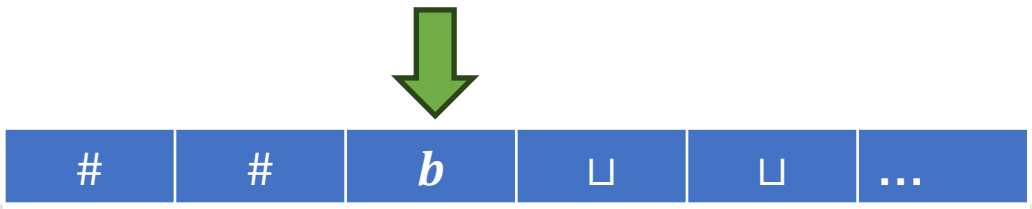
Stato
q_b



Complessità Temporale – Esempio

(q, x)	$\delta(q, x)$
(q_0, a)	$(q_a, \#, \rightarrow)$
(q_0, b)	$(q_b, \#, \rightarrow)$
$(q_0, \#)$	$(q_0, \#, \rightarrow)$
$(q_0, \sqcup)$	$(q_{yes}, \sqcup, \rightarrow)$
$(q_a, \sqcup)$	$(q'_a, \sqcup, \leftarrow)$
$(q_a, x) \mid x \neq \sqcup$	$(q_a, x, \rightarrow)$
$(q_b, \sqcup)$	$(q'_b, \sqcup, \leftarrow)$
$(q_b, x) \mid x \neq \sqcup$	$(q_b, x, \rightarrow)$
$(q'_a, x) \mid x \in \{a, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_a, x) \mid x \notin \{a, \#\}$	$(q_{no}, x, -)$
$(q'_b, x) \mid x \in \{b, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_b, x) \mid x \notin \{b, \#\}$	$(q_{no}, x, -)$
$(q_{ret}, \#)$	$(q_0, \#, \rightarrow)$
$(q_{ret}, x) \mid x \neq \#$	$(q_{ret}, x, \leftarrow)$

Stato
q'_b



Complessità Temporale – Esempio

(q, x)	$\delta(q, x)$
(q_0, a)	$(q_a, \#, \rightarrow)$
(q_0, b)	$(q_b, \#, \rightarrow)$
$(q_0, \#)$	$(q_0, \#, \rightarrow)$
$(q_0, \sqcup)$	$(q_{yes}, \sqcup, \rightarrow)$
$(q_a, \sqcup)$	$(q'_a, \sqcup, \leftarrow)$
$(q_a, x) \mid x \neq \sqcup$	$(q_a, x, \rightarrow)$
$(q_b, \sqcup)$	$(q'_b, \sqcup, \leftarrow)$
$(q_b, x) \mid x \neq \sqcup$	$(q_b, x, \rightarrow)$
$(q'_a, x) \mid x \in \{a, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_a, x) \mid x \notin \{a, \#\}$	$(q_{no}, x, -)$
$(q'_b, x) \mid x \in \{b, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_b, x) \mid x \notin \{b, \#\}$	$(q_{no}, x, -)$
$(q_{ret}, \#)$	$(q_0, \#, \rightarrow)$
$(q_{ret}, x) \mid x \neq \#$	$(q_{ret}, x, \leftarrow)$

Stato
q_{ret}

#	#	⊔	⊔	⊔	…

Complessità Temporale – Esempio

(q, x)	$\delta(q, x)$
(q_0, a)	$(q_a, \#, \rightarrow)$
(q_0, b)	$(q_b, \#, \rightarrow)$
$(q_0, \#)$	$(q_0, \#, \rightarrow)$
$(q_0, \sqcup)$	$(q_{yes}, \sqcup, \rightarrow)$
$(q_a, \sqcup)$	$(q'_a, \sqcup, \leftarrow)$
$(q_a, x) \mid x \neq \sqcup$	$(q_a, x, \rightarrow)$
$(q_b, \sqcup)$	$(q'_b, \sqcup, \leftarrow)$
$(q_b, x) \mid x \neq \sqcup$	$(q_b, x, \rightarrow)$
$(q'_a, x) \mid x \in \{a, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_a, x) \mid x \notin \{a, \#\}$	$(q_{no}, x, -)$
$(q'_b, x) \mid x \in \{b, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_b, x) \mid x \notin \{b, \#\}$	$(q_{no}, x, -)$
$(q_{ret}, \#)$	$(q_0, \#, \rightarrow)$
$(q_{ret}, x) \mid x \neq \#$	$(q_{ret}, x, \leftarrow)$

Stato
q_0

#	#	⊔	⊔	⊔	...

Complessità Temporale – Esempio

Accetta la stringa corrente.
Dimensione Input 4
Tempo = 15

Stato
q_{yes}

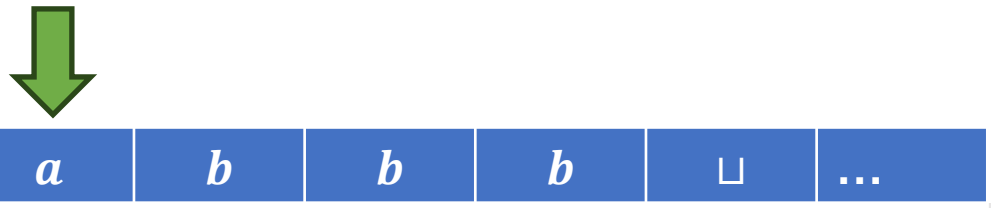
#	#	⊔	⊔	⊔	…

(q, x)	$\delta(q, x)$
(q_0, a)	$(q_a, \#, \rightarrow)$
(q_0, b)	$(q_b, \#, \rightarrow)$
$(q_0, \#)$	$(q_0, \#, \rightarrow)$
$(q_0, \sqcup)$	$(q_{yes}, \sqcup, \rightarrow)$
$(q_a, \sqcup)$	$(q'_a, \sqcup, \leftarrow)$
$(q_a, x) \mid x \neq \sqcup$	$(q_a, x, \rightarrow)$
$(q_b, \sqcup)$	$(q'_b, \sqcup, \leftarrow)$
$(q_b, x) \mid x \neq \sqcup$	$(q_b, x, \rightarrow)$
$(q'_a, x) \mid x \in \{a, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_a, x) \mid x \notin \{a, \#\}$	$(q_{no}, x, -)$
$(q'_b, x) \mid x \in \{b, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_b, x) \mid x \notin \{b, \#\}$	$(q_{no}, x, -)$
$(q_{ret}, \#)$	$(q_0, \#, \rightarrow)$
$(q_{ret}, x) \mid x \neq \#$	$(q_{ret}, x, \leftarrow)$

Complessità Temporale – Esempio

(q, x)	$\delta(q, x)$
(q_0, a)	$(q_a, \#, \rightarrow)$
(q_0, b)	$(q_b, \#, \rightarrow)$
$(q_0, \#)$	$(q_0, \#, \rightarrow)$
$(q_0, \sqcup)$	$(q_{yes}, \sqcup, \rightarrow)$
$(q_a, \sqcup)$	$(q'_a, \sqcup, \leftarrow)$
$(q_a, x) \mid x \neq \sqcup$	$(q_a, x, \rightarrow)$
$(q_b, \sqcup)$	$(q'_b, \sqcup, \leftarrow)$
$(q_b, x) \mid x \neq \sqcup$	$(q_b, x, \rightarrow)$
$(q'_a, x) \mid x \in \{a, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_a, x) \mid x \notin \{a, \#\}$	$(q_{no}, x, -)$
$(q'_b, x) \mid x \in \{b, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_b, x) \mid x \notin \{b, \#\}$	$(q_{no}, x, -)$
$(q_{ret}, \#)$	$(q_0, \#, \rightarrow)$
$(q_{ret}, x) \mid x \neq \#$	$(q_{ret}, x, \leftarrow)$

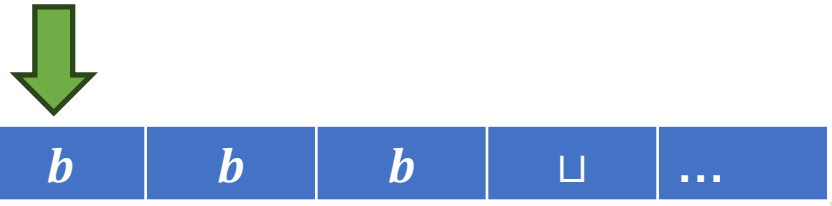
Stato
q_0



Complessità Temporale – Esempio

(q, x)	$\delta(q, x)$
(q_0, a)	$(q_a, \#, \rightarrow)$
(q_0, b)	$(q_b, \#, \rightarrow)$
$(q_0, \#)$	$(q_0, \#, \rightarrow)$
$(q_0, \sqcup)$	$(q_{yes}, \sqcup, \rightarrow)$
$(q_a, \sqcup)$	$(q'_a, \sqcup, \leftarrow)$
$(q_a, x) \mid x \neq \sqcup$	$(q_a, x, \rightarrow)$
$(q_b, \sqcup)$	$(q'_b, \sqcup, \leftarrow)$
$(q_b, x) \mid x \neq \sqcup$	$(q_b, x, \rightarrow)$
$(q'_a, x) \mid x \in \{a, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_a, x) \mid x \notin \{a, \#\}$	$(q_{no}, x, -)$
$(q'_b, x) \mid x \in \{b, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_b, x) \mid x \notin \{b, \#\}$	$(q_{no}, x, -)$
$(q_{ret}, \#)$	$(q_0, \#, \rightarrow)$
$(q_{ret}, x) \mid x \neq \#$	$(q_{ret}, x, \leftarrow)$

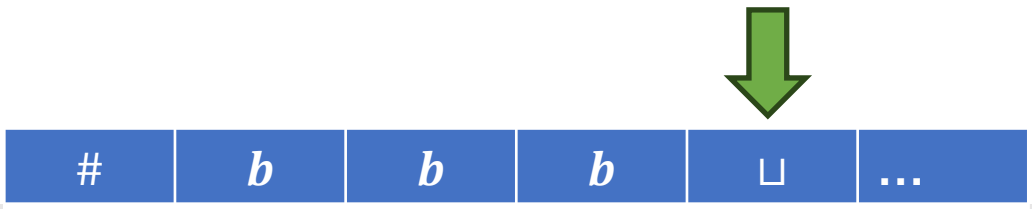
Stato
q_a



Complessità Temporale – Esempio

(q, x)	$\delta(q, x)$
(q_0, a)	$(q_a, \#, \rightarrow)$
(q_0, b)	$(q_b, \#, \rightarrow)$
$(q_0, \#)$	$(q_0, \#, \rightarrow)$
$(q_0, \sqcup)$	$(q_{yes}, \sqcup, \rightarrow)$
$(q_a, \sqcup)$	$(q'_a, \sqcup, \leftarrow)$
$(q_a, x) \mid x \neq \sqcup$	$(q_a, x, \rightarrow)$
$(q_b, \sqcup)$	$(q'_b, \sqcup, \leftarrow)$
$(q_b, x) \mid x \neq \sqcup$	$(q_b, x, \rightarrow)$
$(q'_a, x) \mid x \in \{a, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_a, x) \mid x \notin \{a, \#\}$	$(q_{no}, x, -)$
$(q'_b, x) \mid x \in \{b, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_b, x) \mid x \notin \{b, \#\}$	$(q_{no}, x, -)$
$(q_{ret}, \#)$	$(q_0, \#, \rightarrow)$
$(q_{ret}, x) \mid x \neq \#$	$(q_{ret}, x, \leftarrow)$

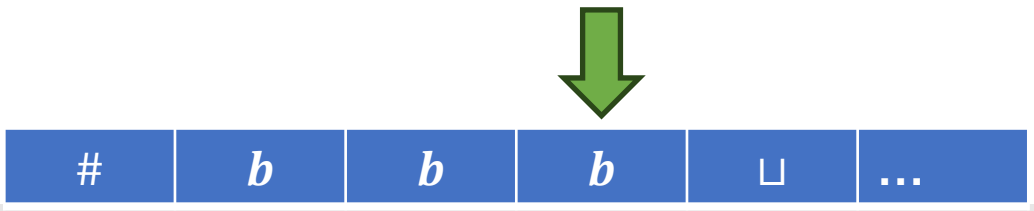
Stato
q_a



Complessità Temporale – Esempio

(q, x)	$\delta(q, x)$
(q_0, a)	$(q_a, \#, \rightarrow)$
(q_0, b)	$(q_b, \#, \rightarrow)$
$(q_0, \#)$	$(q_0, \#, \rightarrow)$
$(q_0, \sqcup)$	$(q_{yes}, \sqcup, \rightarrow)$
$(q_a, \sqcup)$	$(q'_a, \sqcup, \leftarrow)$
$(q_a, x) \mid x \neq \sqcup$	$(q_a, x, \rightarrow)$
$(q_b, \sqcup)$	$(q'_b, \sqcup, \leftarrow)$
$(q_b, x) \mid x \neq \sqcup$	$(q_b, x, \rightarrow)$
$(q'_a, x) \mid x \in \{a, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_a, x) \mid x \notin \{a, \#\}$	$(q_{no}, x, -)$
$(q'_b, x) \mid x \in \{b, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_b, x) \mid x \notin \{b, \#\}$	$(q_{no}, x, -)$
$(q_{ret}, \#)$	$(q_0, \#, \rightarrow)$
$(q_{ret}, x) \mid x \neq \#$	$(q_{ret}, x, \leftarrow)$

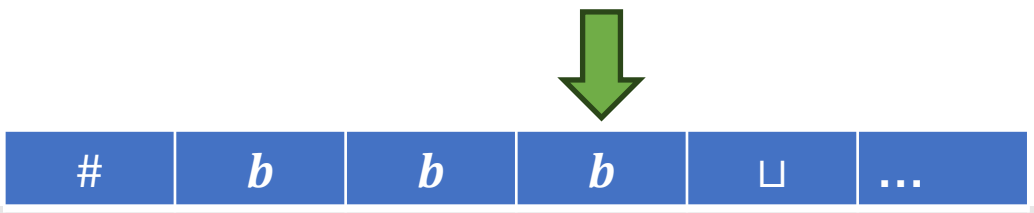
Stato
q'_a



Complessità Temporale – Esempio

Rfiuta la stringa corrente.
Dimensione Input 4
Tempo = 6

Stato
q_{no}



(q, x)	$\delta(q, x)$
(q_0, a)	$(q_a, \#, \rightarrow)$
(q_0, b)	$(q_b, \#, \rightarrow)$
$(q_0, \#)$	$(q_0, \#, \rightarrow)$
$(q_0, \sqcup)$	$(q_{yes}, \sqcup, \rightarrow)$
$(q_a, \sqcup)$	$(q'_a, \sqcup, \leftarrow)$
$(q_a, x) \mid x \neq \sqcup$	$(q_a, x, \rightarrow)$
$(q_b, \sqcup)$	$(q'_b, \sqcup, \leftarrow)$
$(q_b, x) \mid x \neq \sqcup$	$(q_b, x, \rightarrow)$
$(q'_a, x) \mid x \in \{a, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_a, x) \mid x \notin \{a, \#\}$	$(q_{no}, x, -)$
$(q'_b, x) \mid x \in \{b, \#\}$	$(q_{ret}, \sqcup, \leftarrow)$
$(q'_b, x) \mid x \notin \{b, \#\}$	$(q_{no}, x, -)$
$(q_{ret}, \#)$	$(q_0, \#, \rightarrow)$
$(q_{ret}, x) \mid x \neq \#$	$(q_{ret}, x, \leftarrow)$

Complessità Temporale – Esempio

- La macchina che abbiamo descritto esegue le seguenti operazioni per ogni x
  1. Dato un input $x \in \{a, b\}^*$ con $|x| = n$
  2. Per $i = 1, \dots, \frac{n}{2}$ confronta il simbolo i-esimo con il simbolo $n-1$-esimo di x
     1. Se sono diversi rifiuta
  3. Se tutti i controlli sono andati a buon fine accetta

- Per ogni input x con $|x| = n$
  - La macchina esegue $\frac{n}{2}$ fasi (gli $\frac{n}{2}$ controlli dei caratteri agli estremi opposti della stringa)
  - Ogni fase prevede al più $2n + 1$ passi
    - Spostare la testina dall'inizio alla fine del nastro (al più n configurazioni)
    - Eseguire il controllo (una configurazione 1)
    - Tornare indietro (al più n configurazioni)
- La macchina M ha complessità temporale $f_M = O(n^2)$

- **Proposizione**. La funzione $\mathrm{PAL}_{\{a,b\}}$ è in $TIME(n^2)$

La Classe di complessità P

- Convenzionalmente, consideriamo **trattabili nella pratica** i problemi per cui il corrispondente linguaggio si trova in $TIME(n^k)$ per un qualche $k \in N$

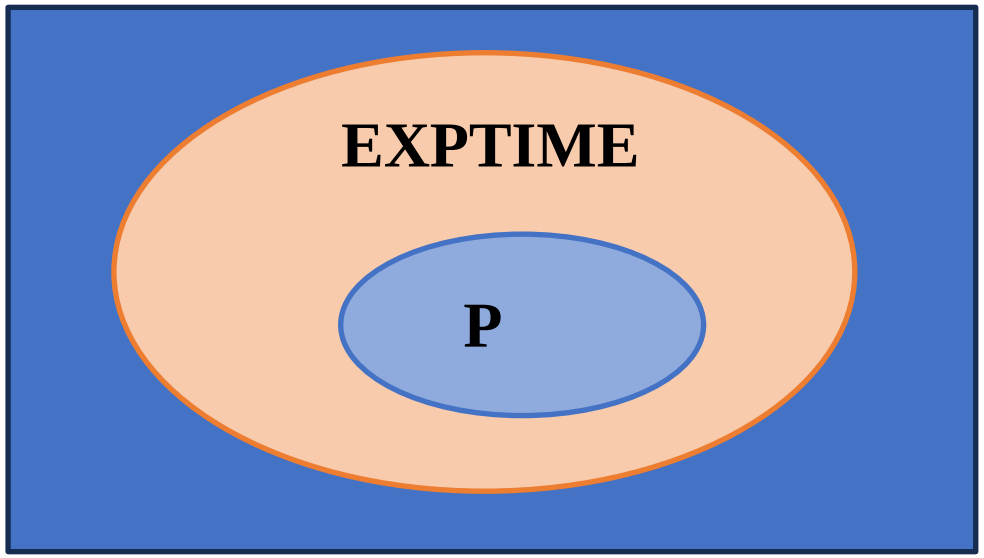
- **Definizione**. La classe $\mathbf{P}$ è definita come $\mathbf{P} = \mathrm{U}_{k \in N}\, \mathrm{TIME}(n^k)$
  - L'unione di tutti i linguaggi decidibili con una macchina di Turing il cui tempo di esecuzione cresce al più come un polinomio nella dimensione dell'input

- **Proposizione**. I linguaggi non-contestuali sono un sottoinsieme stretto di $\mathbf{P}$
- **Corollario**. I linguaggi regolari sono un sottoinsieme stretto di $\mathbf{P}$

La Classe di Complessità **EXPTIME**

- Convenzionalmente, consideriamo **non trattabili nella pratica** quelli per cui il corrispondente linguaggio si trova in $TIME(2^{c \cdot n})$ per un qualche $c \in N$ **ma non in P**

- **Definizione**. La classe **EXPTIME** è definita come $\mathbf{EXPTIME} = \bigcup_{k \in N} \mathrm{TIME}(2^{c \cdot n})$
  - L'unione di tutti i linguaggi decidibili con una macchina di Turing il cui tempo di esecuzione cresce al più come un esponenziale nella dimensione dell'input



Problemi in **EXPTIME – Esempio**

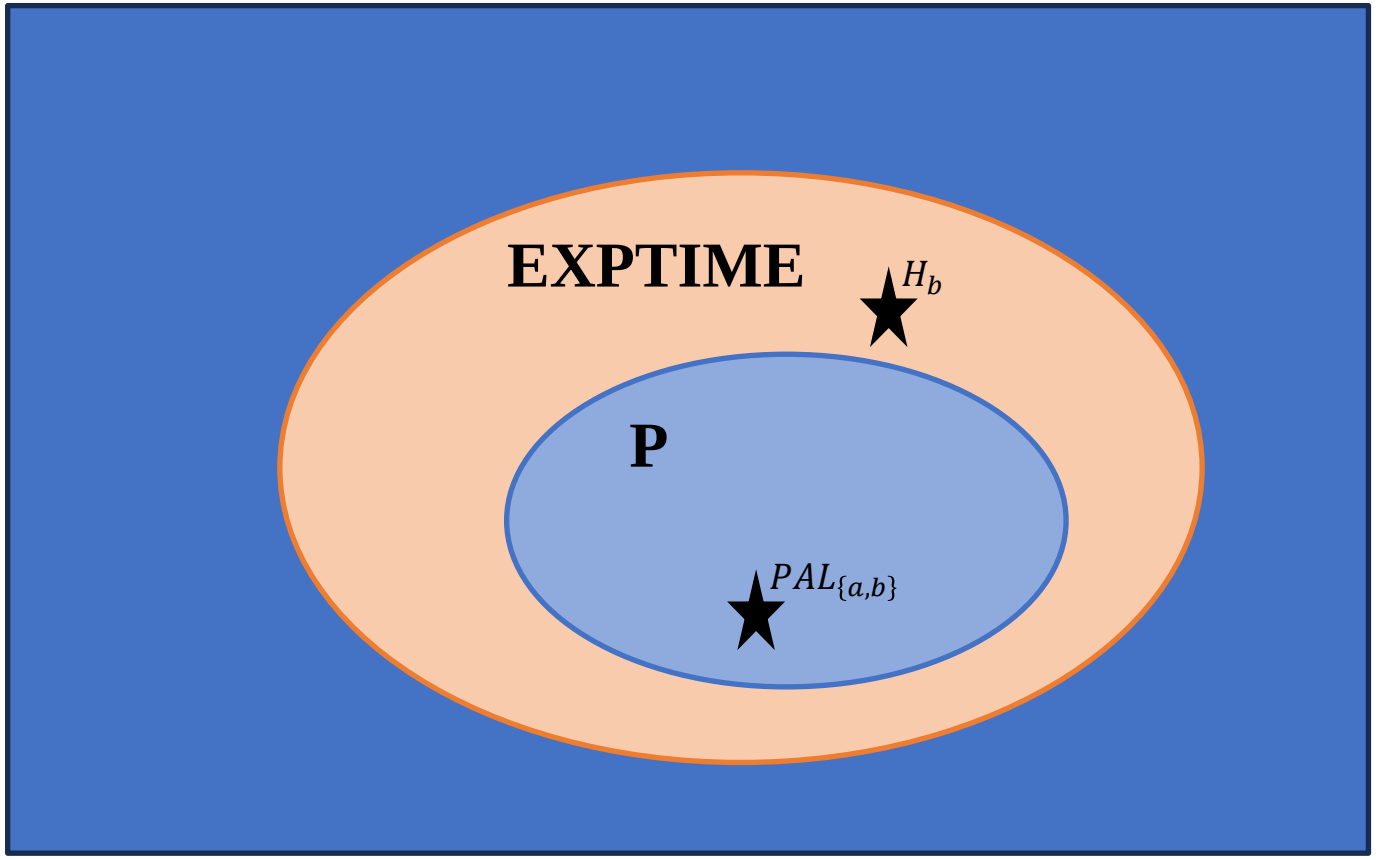
- Esistono dei linguaggi in $\mathbf{EXPTIME}$ per cui siamo riusciti a dimostrare che non esiste una Macchina di Turing che li decide e il cui tempo di esecuzione è polinomiale
  - Sono ovviamente complessi anche solo da definire …

- Il seguente problema (abbastanza intuitivo nella definizione) è uno di questi: data una Macchina di Turing M, un input x per M e un interno $k \in \mathbb{N}$, è vero che M termina in al più k passi?

- **Definizione**. $H_b = \{ (e(M), e(\sigma),\ e(n)) \mid M\ termina\ dopo\ n\ passi\ su\ input\ \sigma\}$ dove $e(M)$ è l'encoding di una Turing Machine, $M(\sigma)$ è l'encoding dell'input di M e $e(n)$ è l'Encoding binario di n

- **Proposizione**. Il linguaggio H_b è in $\mathbf{EXPTIME}$ ma non in $\mathbf{P}$
- **Dimostrazione (intuizione)**. Dobbiamo simulare la macchina per al più $n = O(2^{|e(n)|})$ passi

La Classe di Complessità **EXPTIME**



Complessità Non-Deterministiche e la Classe NP

Fra EXPTIME e P

- La distinzione fra la classe $\mathbf{P}$ e la classe $\mathbf{EXPTIME}$ è molto netta
  - $\mathbf{EXPTIME}$ contiente $\mathbf{P}$ ma anche problemi molto più complessi
  - Alcuni di questi problemi non possono essere risolti in pratica
  - **Esempio**: Per controllare se$(e(M), e(\sigma),\ e(n)) \in H_b$ servono $2^{|e(n)|}$ passi
  - **Esempio**: Per $|e(n)| = 100$ abbiamo $2^{|e(n)|} \geq 10^{80}$ (stima degli atomi nell'universo)

- Se le nostre analisi si riducessero a dire se un problema è in $\mathbf{P}$ o in $\mathbf{EXPTIME}$ ma non in $\mathbf{P}$ non sarebbero particolarmente utili

- Esistono delle classi di problemi intermedie?
  - Non necessariamente contenute in $\mathbf{P}$ **ma non complesse come** $\mathbf{EXPTIME}$

Macchina di Turing Non-Deterministica – Intuizione

- Per ottenere analisi più raffinate, introduciamo la nozione di MdT non deterministiche

- **Intuizione**. Come per gli ASF, la prossima configurazione di una MdT non deterministica è definita da un **insieme di transizioni possibili** invece di una singola transizione
  - La funzione di transizione collega una configurazione ad un insieme di possibili transizioni
  - La macchina accetta se esiste almeno una configurazione accettante che può essere raggiunta in questo modo

- **Intuizione 1.** Dal punto di vista della **computabilità**, le macchine non-deterministiche sono identiche a quelle deterministiche
  - Tesi di Church e Turing

- **Intuizione 2.** Dal punto di vista della **complessità**, le macchine non-deterministiche definiscono classi differenti che ci consentono di eseguire analisi più accurate

Macchina di Turing Non Deterministica – Definizione Formale

Definizione. Una **macchina di Turing non deterministica** (**MTND**) M è una 7-upla $M = \langle \Sigma, \Gamma, Q, \delta, q_0, q_{yes}, q_{no} \rangle$ tale che:

- Σ è un insieme finito di simboli, detto **insieme dei simboli di input** che **non include** il **simbolo blank** $\sqcup$
- Γ con $\Sigma \subseteq \Gamma$ è un insieme finito di simboli, detto **insieme dei simboli del nastro**, che **include** il **simbolo blank** $\sqcup$
- Q è un insieme finito e non vuoto di **stati** **tali che**:
  - $q_0 \in Q$ è lo **stato iniziale**
  - $q_{yes} \in Q$ è lo **stato accettante**
  - $q_{no} \in Q$ è lo **stato rifiutante**
- $Q' = Q \setminus \{q_{yes}, q_{no}\}$
- δ è la **funzione di transizione**; ovvero, una funzione totale definita come segue

$$\delta: Q' \times \Gamma \to \mathrm{P}(\mathrm{Q} \times \Gamma \times \{\leftarrow, \rightarrow, -\})$$

Macchina di Turing Non-Deterministica – Configurazioni

- Sia $M = < \Sigma, \Gamma, Q, \delta, q_0, q_{yes}, q_{no} >$ una **macchina di Turing non deterministica**
- **Definizione**. Una **configurazione** di *M* è una 3-upla $C = (\sigma, \mathrm{q}, \tau)$ tale che:
  - σ è una **stringa sull'alfabeto** Γ (nastro a sinistra della testina)
  - $\tau = a\tau'$ è una **stringa sull'alfabeto** Γ (nastro a destra della testina e a è il simbolo corrente)
  - $q \in Q$ è uno **stato** di M (stato corrente della configurazione)
- **Definizione**. Una **configurazione** $C = (\sigma, \mathrm{q}, \tau)$ di *M* è detta:
  - **Accettante** (**finale**) se $q = q_{yes}$
  - **Rifiutante** (**finale**) se $q = q_{no}$
  - **Iniziale** se se $q = q_0$, $\sigma = \epsilon$ (stringa vuota) e $\tau \in \Sigma^*$ (τ è una stringa dell'alfabeto dell'input)
- **Nota. Le definizioni di configurazione sono identiche a quelle per la machina standard**

Macchina di Turing Non-Deterministica – Esecuzioni

- Sia $M = < \Sigma, \Gamma, Q, \delta, q_0, q_{yes}, q_{no} >$ una **macchina di Turing non deterministica**

- **Definizione**. Una **configurazione C genera** una **configurazione D** in M ($C \Rightarrow_M D$) se
  - $C = (\sigma y,\ q,\ x\tau)$, $D = (\sigma,\ q',\ yz\tau)$ e $\delta(q,x) \in (\mathrm{q}', \mathrm{z}, \leftarrow)$ **e $\sigma \neq \epsilon$** (**movimento a sinistra**)
  - $C = (\epsilon, q,\ x\tau)$, $D = (\epsilon,\ q',\ z\tau)$ e $\delta(q,x) \in (\mathrm{q}', \mathrm{z}, \leftarrow)$ (**movimento a sinistra bloccato**)
  - $C = (\sigma,\ q,\ xy\tau)$, $D = (\sigma z, q', y\tau)$ e $\delta(q,x) \in (\mathrm{q}', \mathrm{z}, \rightarrow)$ **e $\tau \neq \epsilon$** (**movimento a destra**)
  - $C = (\sigma,\ q,\ \epsilon)$, $D = (\sigma z, q',\ \epsilon)$ e $\delta(q,\sqcup) \in (\mathrm{q}', \mathrm{z}, \rightarrow)$ (**movimento a destra oltre il nastro corrente**)
  - $C = (\sigma,\ q,\ x\tau)$, $D = (\sigma, q', z\tau)$ e $\delta(q,x) \in (\mathrm{q}', \mathrm{z}, -)$ (**nessun movimento**)

Nota. Una configurazione C potrebbe generare più di una configurazione D

Macchina di Turing Non-Deterministica – Esecuzioni

- **Definizione**. La macchina M **accetta l'input** $\sigma \in \Sigma^*$ se esiste una **sequenza finita** di configurazioni $C_1, C_2, \dots, C_n$ di M tale che le seguenti proprietà sono soddisfatte
  - $C_1 = (\epsilon, q_0, \sigma)$;
  - C_n è una configurazione **accettante**;
  - $C_i \Rightarrow_M C_{i+1}$ per ogni $i = 1, \dots, n-1$

- **Definizione**. La macchina M **rifiuta l'input** $\sigma \in \Sigma^*$ se ogni **sequenza finita** di configurazioni $C_1, C_2, \dots, C_n$ di M tale $C_1 = (\epsilon, q_0, \sigma)$ **e** $C_i \Rightarrow_M C_{i+1}$ per ogni $i = 1, \dots, n-1$ è tale che C_n **è rifiutante**

Macchina di Turing Non-Deterministica – Esempio

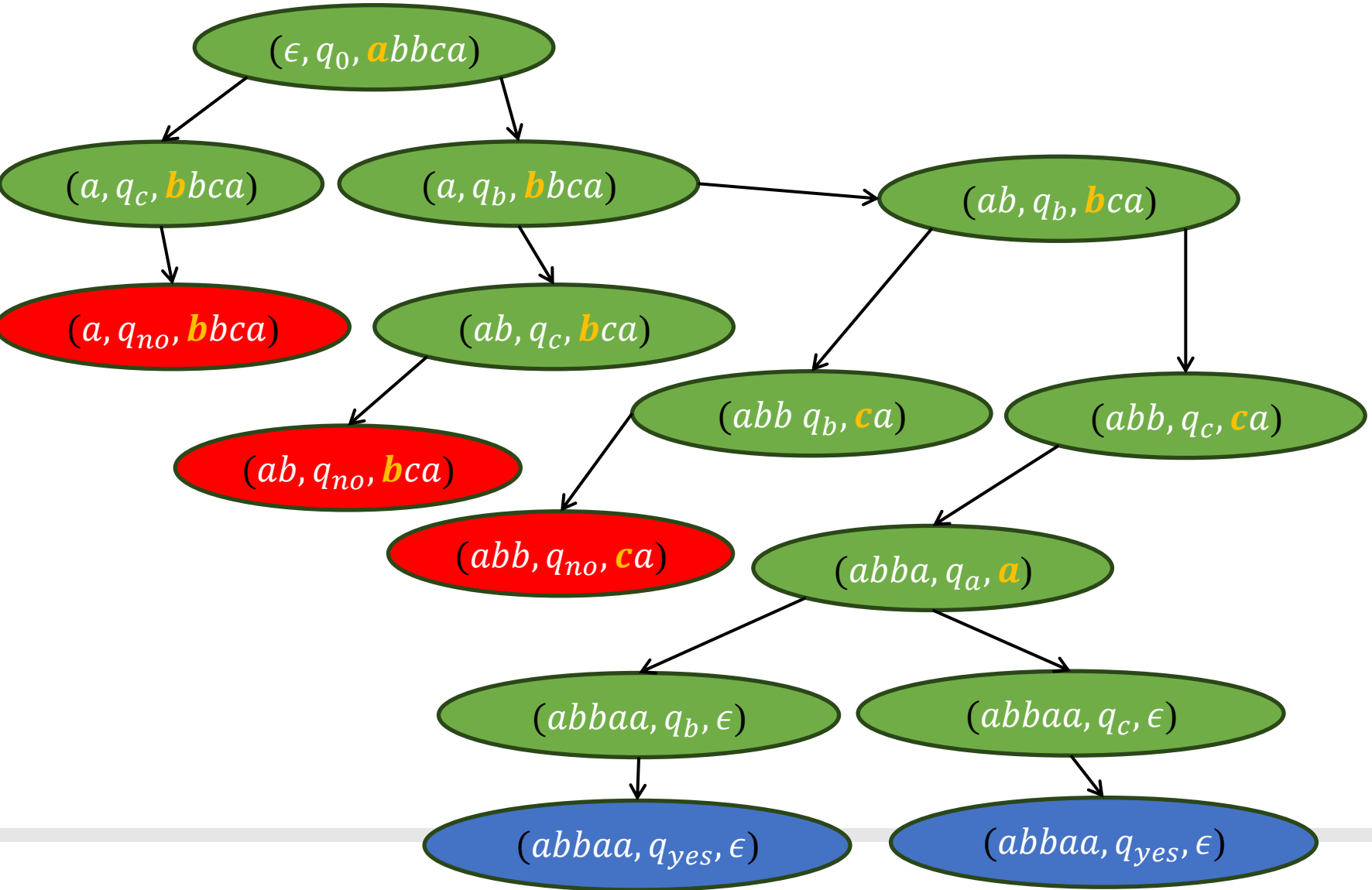
- Sia $\mathcal{L} \subseteq \{a, b\}^*$ il linguaggio delle stringhe s tale che
  - Ogni stringa inizia con a
  - Ogni a è seguita da b o da c
  - Ogni b è seguita da b o da c
  - Ogni c è seguita da a

- La MTND definita dalla tabella riconosce $\mathcal{L}$ ed è terminante
  - $\Sigma = \{a, b\}; \Gamma = \Sigma \cup \{\sqcup\}; Q = \{q_0, q_1, q_2, q_{yes}, q_{no}\}$

(q, x)	$\delta(q, x)$
(q_0, a)	$\{(q_b, a, \rightarrow), (q_c, a, \rightarrow)\}$
(q_0, b)	$\{(q_{no}, b, -)\}$
(q_0, c)	$\{(q_{no}, c, -)\}$
(q_a, a)	$\{(q_b, a, \rightarrow), (q_c, a, \rightarrow)\}$
(q_a, b)	$\{(q_{no}, b, -)\}$
(q_a, c)	$\{(q_{no}, c, -)\}$
(q_b, a)	$\{(q_{no}, b, -)\}$
(q_b, b)	$\{(q_b, b, \rightarrow), (q_c, b, \rightarrow)\}$
(q_b, c)	$\{(q_{no}, c, -)\}$
(q_c, a)	$\{(q_{no}, a, -)\}$
(q_c, b)	$\{(q_{no}, b, -)\}$
(q_c, c)	$\{(q_a, c, \rightarrow)\}$
$(*, \sqcup)$	$\{(q_{yes}, \sqcup, -)\}$

MTDN – Configurazioni

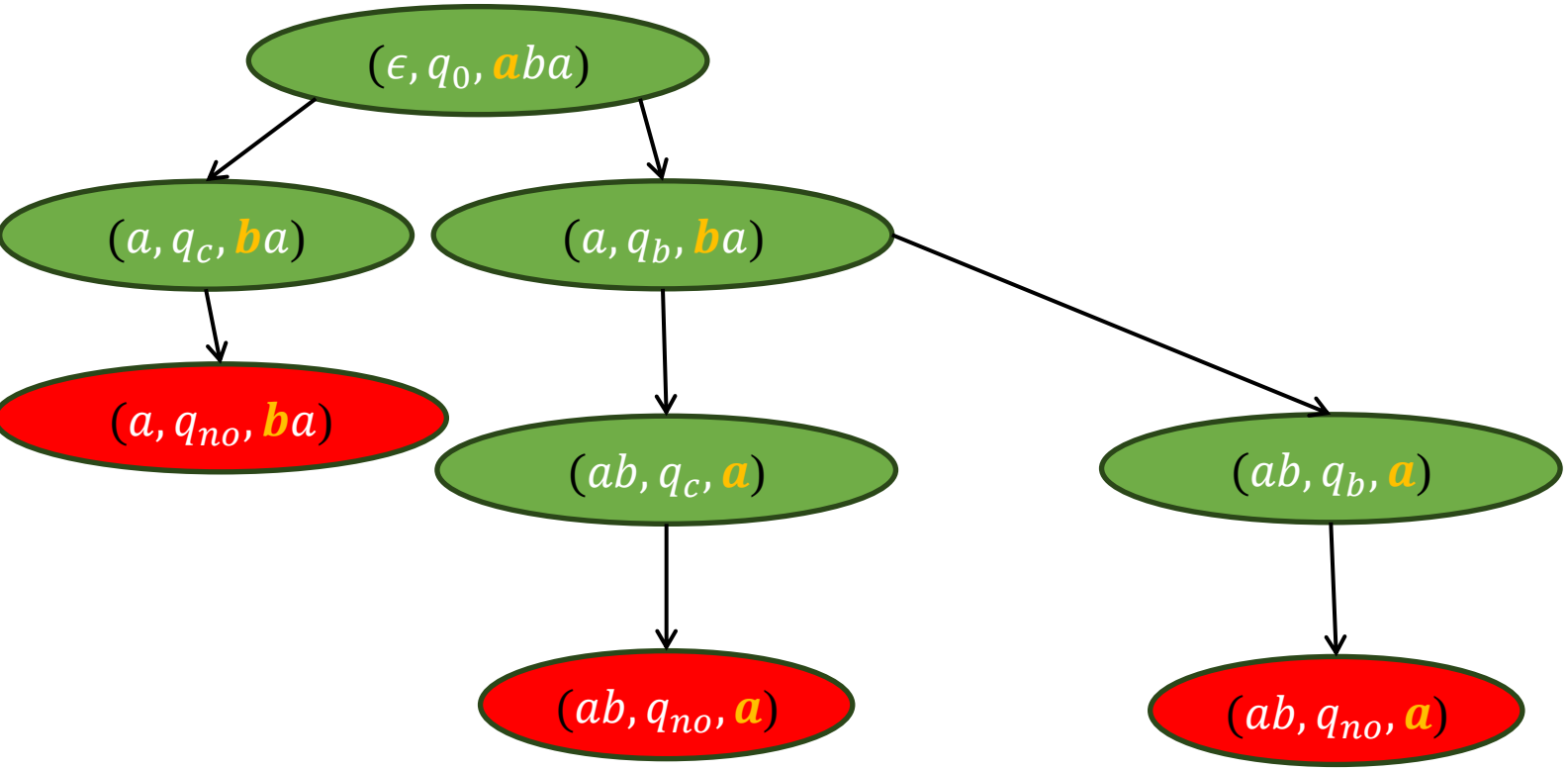
- Configurazioni generate con input $\sigma = abbca$



(q, x)	$\delta(q, x)$
(q_0, a)	$\{(q_b, a, \rightarrow), (q_c, a, \rightarrow)\}$
(q_0, b)	$\{(q_{no}, b, -)\}$
(q_0, c)	$\{(q_{no}, c, -)\}$
(q_a, a)	$\{(q_b, a, \rightarrow), (q_c, a, \rightarrow)\}$
(q_a, b)	$\{(q_{no}, b, -)\}$
(q_a, c)	$\{(q_{no}, c, -)\}$
(q_b, a)	$\{(q_{no}, b, -)\}$
(q_b, b)	$\{(q_b, b, \rightarrow), (q_c, b, \rightarrow)\}$
(q_b, c)	$\{(q_{no}, c, -)\}$
(q_c, a)	$\{(q_{no}, a, -)\}$
(q_c, b)	$\{(q_{no}, b, -)\}$
(q_c, c)	$\{(q_a, c, \rightarrow)\}$
$(*, \sqcup)$	$\{(q_{yes}, \sqcup, -)\}$

MTDN – Configurazioni

- Configurazioni generate con input $\sigma = aba$



(q, x)	$\delta(q, x)$
(q_0, a)	$\{(q_b, a, \rightarrow), (q_c, a, \rightarrow)\}$
(q_0, b)	$\{(q_{no}, b, -)\}$
(q_0, c)	$\{(q_{no}, c, -)\}$
(q_a, a)	$\{(q_b, a, \rightarrow), (q_c, a, \rightarrow)\}$
(q_a, b)	$\{(q_{no}, b, -)\}$
(q_a, c)	$\{(q_{no}, c, -)\}$
(q_b, a)	$\{(q_{no}, b, -)\}$
(q_b, b)	$\{(q_b, b, \rightarrow), (q_c, b, \rightarrow)\}$
(q_b, c)	$\{(q_{no}, c, -)\}$
(q_c, a)	$\{(q_{no}, a, -)\}$
(q_c, b)	$\{(q_{no}, b, -)\}$
(q_c, c)	$\{(q_a, c, \rightarrow)\}$
$(*, \sqcup)$	$\{(q_{yes}, \sqcup, -)\}$

Macchina di Turing Non-Deterministica – Equivalenza

- **Definizione**. L'insieme $L(M)$ delle stringhe **che M riconosce** è detto **il linguaggio riconosciuto da M**

- **Definizione**. Una MTND M è **terminante** se per ogni stringa $\sigma \in \Sigma$ M **accetta** o **rifiuta** σ

- **Teorema**. Sia $\mathcal{L}$ un linguaggio di stringhe.
  1. $\mathcal{L}$ è **riconosciuto da un MTND** se e solo se $\mathcal{L}$ è **Turing Riconoscibile**
  2. $\mathcal{L}$ è **riconosciuto da un MTND** se e solo se $\mathcal{L}$ è **Turing Decidibile**

- **Dimostrazione**. La dimostrazione procede sfruttando una serie di proprietà delle MTND
  1. Le configurazioni generate da M con input σ definiscono un albero
  2. Esplorando tale albero con una ricerca in ampiezza possiamo
     1. trovare la configurazione accettante se esiste oppure
     2. Rifiutare, se tale configurazione non esiste e la macchina termina
  3. La ricerca in ampiezza si può implementare un un 3-TM
     1. Che è equivalente a una Macchina di Turing standard

Complessità Temporale Non-Deterministica

- Definizioni simili a quelle che abbiamo dato per le macchine deterministiche possono essere date per quelle non-deterministiche

- Sia $M = \langle \Sigma, \Gamma, Q, \delta, q_0, q_{yes}, q_{no} \rangle$ una Macchina di Turing non-deterministica terminante e $x \in \Sigma^*$

- **Definizione**. **Una esecuzione di M** con input x è una sequenza di configurazioni di M $C_1, C_2, \dots, C_n$ tale che
  - C_1 è la configurazione iniziale di M con input x
  - C_n è una configurazione **accettante\rifiutante**
  - $C_i \Rightarrow_M C_{i+1}$ per ogni $i = 1, \dots, n-1$

- **Definizione**. Utilizziamo $E_M(x)$ per l'insieme di tutte le esecuzioni di M con input x

- **Definizione**. La **complessità temporale** $f_M: \mathbb{N} \to \mathbb{N}$ di M è la funzione $f_M(n) = \max_{\{x \in \Sigma^* \mid |x| = n\}} |E_M(x)|$
  - $f_M(n)$ è la lunghezza della più lunga esecuzione di M con input σ tale che $|\sigma| = n$

Nondeterminismo come modello computazionale - II

- **Definizione**. Sia $g: \mathbb{N} \to \mathbb{N}$ una funzione. $NTIME(g)$ è la famiglia dei linguaggi che possono essere decisi da una Macchina di Turing non-deterministica M la cui **complessità temporale** f_M è un O-grande di g

- **Definizione**. L'unione di tutti i linguaggi che sono in $NTIME(n^k)$ per un qualche $k \in N$ viene denotata con $\mathbf{NP}$ (**Non-Deterministic Polynomial Time**); ovvero

$$\mathbf{NP} = \bigcup_{k \in N} NTIME(n^k)$$

- **Proposizione.** Ogni problema in $\mathbf{P}$ è anche in $\mathbf{NP}$; ovvero, $\mathbf{P} \subseteq \mathbf{NP}$

- La classe NP gioca un ruolo importante nella teoria della complessità
  - Moltissimi problemi pratici sono in questa classe
  - Una macchina non-deterministica è molto potente anche se è vincolata a tempo di esecuzione polinomiale

Esempi di Linguaggi in NP - SAT

- Fissiamo un alfabeto di variabili proposizionali $\mathbf{V}$ e un Encoding ragionevole $\boldsymbol{e}$ per le formule proposizionali su $\boldsymbol{V}$
  - Possiamo utilizzare i simboli stessi delle formule nella macchina

- **Definizione.** L_{SAT} è il linguaggio di stringhe definito come segue

  $\{e(\varphi) \mid \varphi$ è una formula proposizionale **soddisfacibile** su $\boldsymbol{V}\}$

- **Proposizione.** L_{SAT} è in $\mathbf{NP}$

Esempi di Linguaggi in NP - SAT

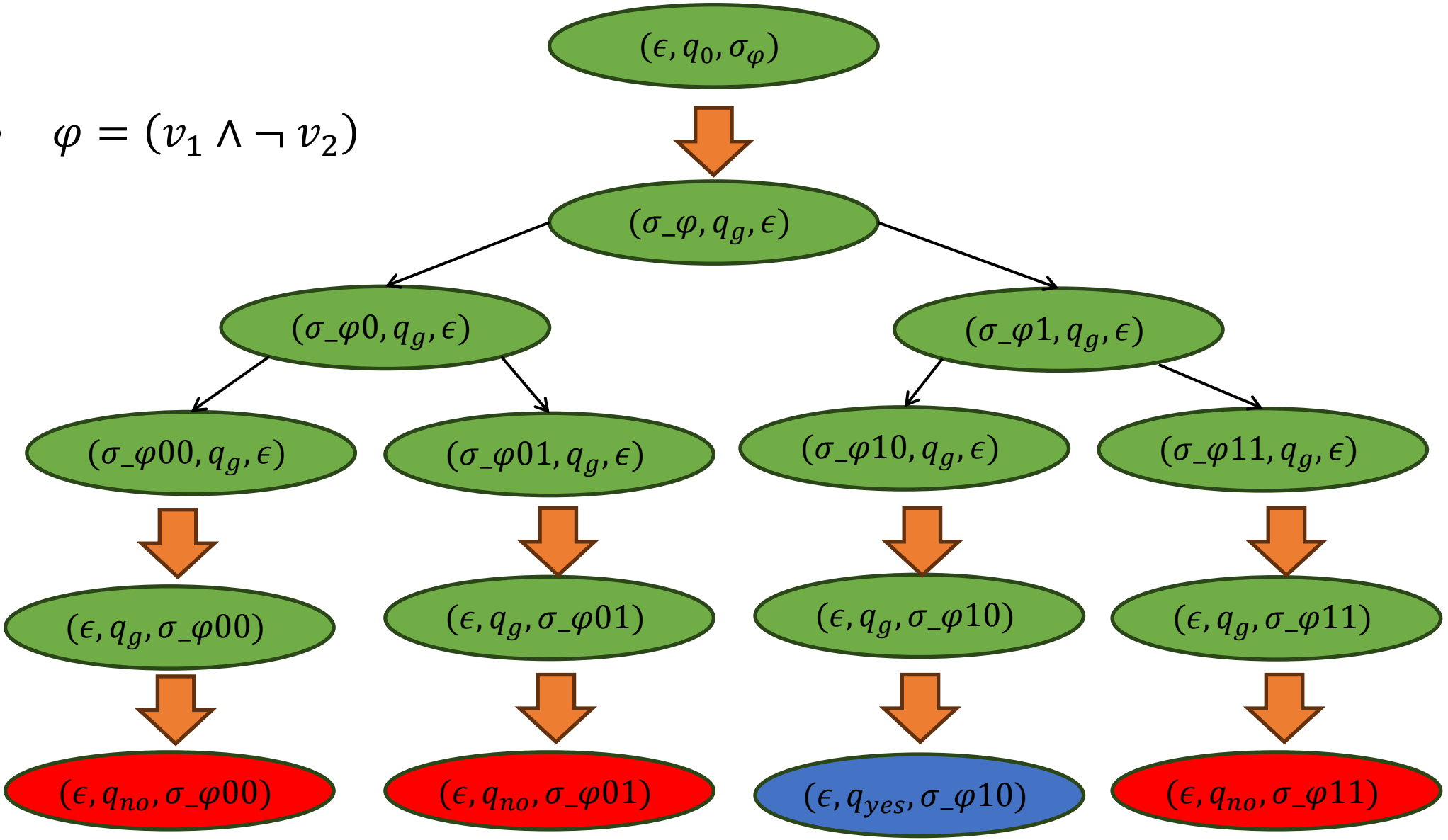
- **Proposizione.** L_{SAT} è in **NP**

- **Dimostrazione**. Per decidere se $\varphi \in L_{SAT}$ una macchina M può eseguire due fasi

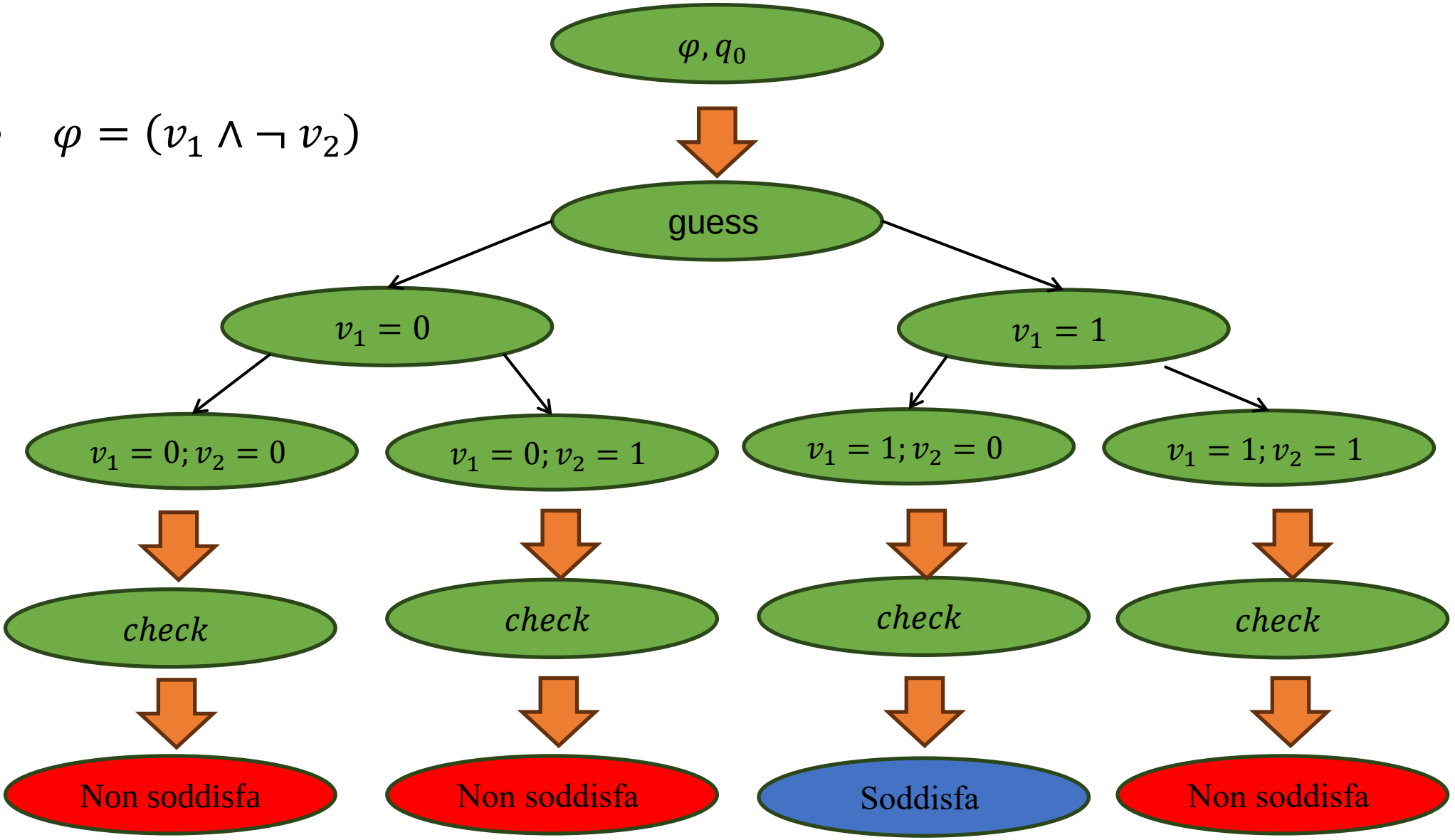
1. M genera non-deterministicamente una interpretazione $\mathcal{I}$ per φ **(GUESS)**
   - Per ogni variabile v M sceglie non-deterministicamente fra $v = 1$ oppure $v = 0$
   - Ci sono un numero di variabili in φ al più pari alla dimensione di φ quindi questa fase può essere seguita in un numero polinomiale di passi della macchina

2. M verifica se $\mathcal{I}$ è un modello per φ **(CHECK)**
   - È possible farlo in tempo polinomiale (applicando le regole delle tabelle di verità)

- $\varphi = (v_1 \wedge \neg\, v_2)$



- $\varphi = (v_1 \wedge \neg\, v_2)$



Esempi di Linguaggi in NP – Hamiltonian Path

- **Definizione.** Un grafo è una coppia $\langle V, E \rangle$ dove V è un insieme (nodi) e E è un insieme di coppie non ordinate di elementi di V (archi)

- **Definizione.** Un cammino (path) p in un grafo G è una sequenza di archi $(e_1, e_2, \dots, e_n)$ di G tale che $e_i \cap e_{i+1} \neq \emptyset$, per $i = 1, \dots, n$ e **ogni nodo** compare al più una volta nella sequenza associata.

- **Definizione. .** Un cammino (path) p in un grafo G è Hamiltoniano se ogni nodo di G compare almeno (esattamente) una volta nella sequenza.

- Fissiamo un Encoding ragionevole $\boldsymbol{e}$ per i grafi non orientati
  - Simboli per i nodi e coppie per gli archi oppure matrice di incidenza

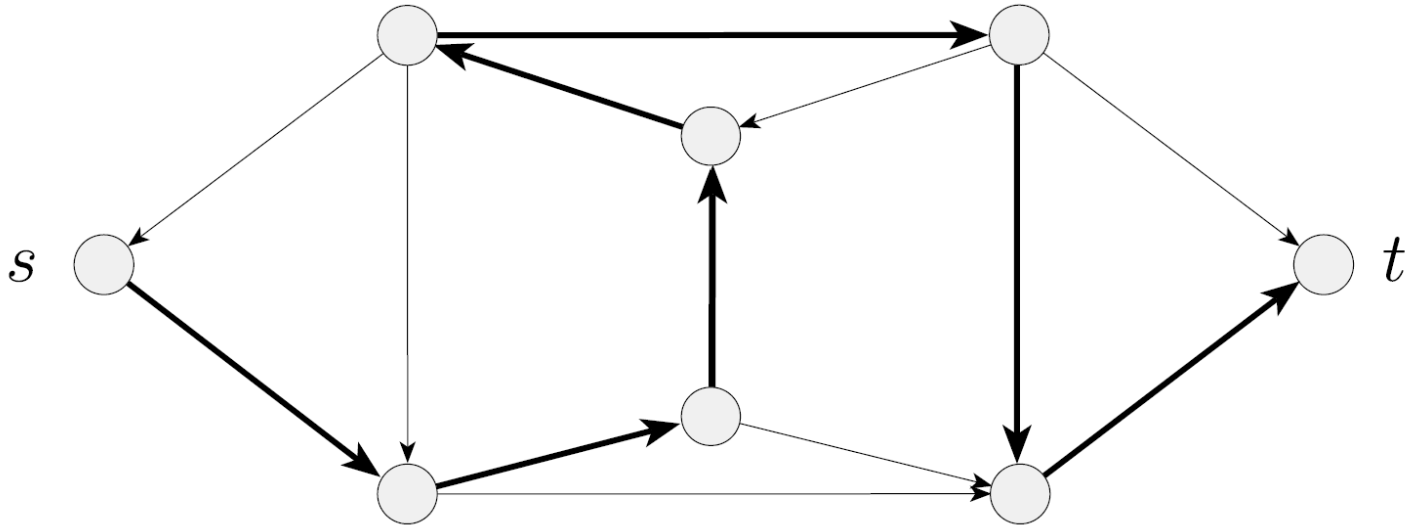
- **Definizione.** Il linguaggio di stringhe P_H è definito come segue

$$\{e(G) \mid G \; contiene \; un \; cammino \; hamiltoniano\}$$

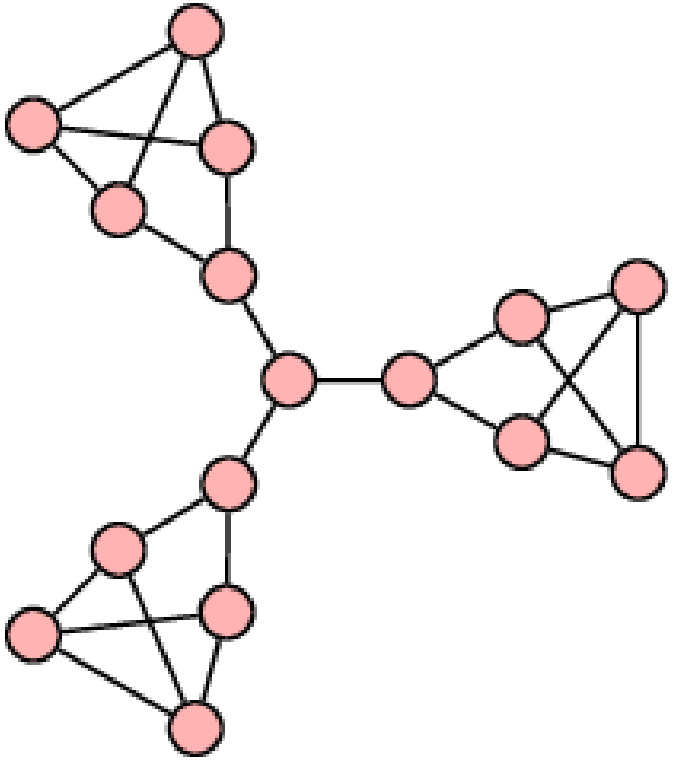
- **Proposizione.** P_H è in **NP**

Esempi di Linguaggi in NP – Hamiltonian Path

Esiste un Hamiltonian Path



NON Esiste un Hamiltonian Path



Esempi di Linguaggi in NP – Hamiltonian Path

- **Proposizione.** P_H è in **NP**

- **Dimostrazione**. Per decidere se $\sigma \in P_H$ una macchina M può eseguire due fasi

1. M genera non-deterministicamente una sequenza di archi $\mathcal{A}$ di G **(GUESS)**
   - Per ogni arco a M sceglie non-deterministicamente se $a \in \mathcal{A}$ oppure $a \notin \mathcal{A}$
   - Ci sono un numero di archi in G al più pari alla dimensione di G quindi questa fase può essere seguita in un numero polinomiale di passi della macchina

2. M verifica se$\mathcal{A}$ è un cammino Hamiltoniano per G **(CHECK)**
   - Verifichiamo che la sequenza sia un cammino scorrendola (passi polinomiali)
   - Verifichiamo che il cammino tocca tutti I nodi del grafo (passi polinomiali)

Nondeterminismo come modello computazionale - III

- **Definizione**. L'unione di tutti i linguaggi che sono in $NTIME(2^{k \cdot n})$ per un qualche $k \in N$ viene denotata con $\mathbf{NEXPTIME}$ (**Non-Deterministic Exponential Time**); ovvero

$$\mathbf{NP} = \bigcup_{k \in N} NTIME(2^{k \cdot n})$$

- **Proposizione.** Ogni problema in $\mathbf{NP}$ è anche in $\mathbf{NEXPTIME}$; ovvero, $\mathbf{NP} \subseteq \mathbf{NEXPTIME}$

Relazioni tra classi di complessità

Relazione tra le classi di complessità - Tempo

- E’ ovvio che **P** ⊆ **NP** e che **EXPTIME** ⊆ **NEXPTIME**
- E’ anche ovvio che **(N)P** ⊆ **(N)EXPTIME** (ogni polinomio f è tale che $f(n) = O(2^n)$)

- Meno ovvio è il fatto che **NP** ⊆ **EXPTIME**,
  - Ma può essere dimostrato

- **Proposizione**. Se $L \in NTIME(f(n))$ allora $L \in TIME(2^{f(n)})$
  - Se un linguaggio L è riconosciuto da una macchina di Turing non-deterministica M in tempo $f(n)$, allora è possibile costruire una macchina di Turing multi-nastro M' che riconosce L e che opera in tempo $O(2^{f(n)})$

Recap e Vari Problemi Aperti

- Il seguente risultato è una conseguenza delle definizioni che abbiamo dato
- **Teorema 1.** $\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{EXPTIME} \subseteq \mathbf{NEXPTIME}$

- La comunità scientifica congettura che tutte inclusioni siano in realtà **strette**. ($\subset$)
  - Esiste un problema in **Y** che non è in **X** per ogni $\mathbf{X} \subseteq \mathbf{Y}$ **con** $\mathbf{X,Y} \in \mathbf{\{P,NP,EXPTIME,NEXPTIME\}}$

- Tuttavia, l'unica cosa che sappiamo è che:
  - $\mathbf{P} \subset \mathbf{EXPTIME}$
    - Esite un problema in **EXPTIME** ce non è in **P**
  - $\mathbf{NP} \subset \mathbf{NEXPTIME}$
    - Esite un problema in **NEXPTIME** ce non è in **NP**

- Tra questi problemi ancora aperti c'è problema $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$, che è uno dei sette problemi del Millennio
  - Il premio per la risoluzione del problema è di 1 milione di dollari!