



UNIVERSITÀ
DEGLI STUDI DI TRIESTE

Computabilità, Complessità e Logica

Prof. Adriano Peron

Complessità: Spazio

Complessità computazionale

- ▶ **Contesto**
- ▶ La prova della decidibilità di un problema è un passo preliminare indispensabile per dimostrare la trattabilità di un problema
- ▶ Dal punto di vista pratico la decidibilità non è spesso una proprietà sufficiente
- ▶ E indispensabile valutare se la quantità di risorse necessaria per la risoluzione del problema è affrontabile
- ▶ Le due dimensioni principali per l'analisi della complessità sono
 - ▶ **Tempo** (numero di passi elementari)
 - ▶ **Spazio** (memoria necessaria)

Complessità computazionale

- ▶ **Contesto**
- ▶ L'appartenenza di un problema a una classe di complessità (come già la decidibilità/indecidibilità) sono **proprietà intrinseche** dei problemi
- ▶ Il modello adottato per la definizione di computabilità (le Macchine di Turing) è anche lo strumento per definire le classi di complessità.
- ▶ **Lo spazio richiesto per l'esecuzione** di una MdT è il numero di celle della MdT utilizzate per la computazione
- ▶ Lo spazio di esecuzione è valutato in funzione di un unico parametro: la **dimensione dell'input sul nastro all'inizio della computazione**
- ▶ Il valore di spazio considerato è **il valore maggiore** rispetto a tutte le computazioni che hanno la stessa dimensione dell'input

Classi di complessità spaziale

► Definizione

Sia $t: N \rightarrow R^+$ una funzione.

La classe di complessità **SPACE(t(n))** è l'insieme dei linguaggi che possono essere decisi mediante **MdT** deterministiche che operano in spazio **O(t(n))**.

La classe di complessità **NSPACE(t(n))** è l'insieme dei linguaggi che possono essere decisi mediante **MdT non-deterministiche** che operano in spazio **O(t(n))**.

Esempio. Si consideri il linguaggio $L = \{a^k b^k : k \geq 0\}$

Sia M una MdT che decide L

1. M controlla che non ci siano b prima di a (una scansione $2n$ passi)
2. M itera la marcatura di una a e una b
3. La MdT usa solo la porzione di nastro riservata all'input e dunque usa spazio $O(n)$.
4. **$L \in \text{SPACE}(n)$.**

Classi di complessità spaziale

Esempio. Si consideri il linguaggio L_{SAT}

1. Sappiamo che è un problema NP-completo e che (se $P \neq NP$) non esiste algoritmo in tempo polinomiale per decidere il linguaggio.

Una MdT non-deterministica

1. Inizia la computazione con la formula booleana sul nastro (input)
2. Genera casualmente una valutazione per le variabili booleane (spazio lineare nel numero delle variabili, non superiore allo spazio della formula che rappresenta l'input)
3. Verifica che la valutazione delle variabili soddisfi la formula (può essere fatto nello stesso spazio della formula)
4. $L_{SAT} \in \text{NSPACE}(n)$.

La classe di complessità PSPACE e NPSPACE

- ▶ La classe di complessità PSPACE include tutti i linguaggi/problemi che siano decisi da una MdT deterministica che decida in spazio polinomiale
- ▶ **DEFINIZIONE.** PSPACE è la classe dei linguaggi che possono essere decisi da una MdT deterministica in spazio polinomiale

$$PSPACE = \bigcup_k SPACE(n^k)$$

- ▶ La classe di complessità NPSPACE include tutti i linguaggi/problemi che siano decisi da una MdT non-deterministica che decida in spazio polinomiale
- ▶ **DEFINIZIONE.** NPSPACE è la classe dei linguaggi che possono essere decisi da una MdT deterministica in spazio polinomiale

$$NPSPACE = \bigcup_k NPSPACE(n^k)$$

Classi di complessità temporale

- ▶ **Considerazioni.**
- ▶ **Nel determinare la decidibilità di un linguaggio/problema è irrilevante il modello computazionale**
- ▶ Il risultato non dipende da determinismo/non determinismo o dall'uso di più nastri
- ▶ **Nel determinare la classe di complessità temporale di un linguaggio/problema è rilevante il modello di MdT utilizzato.**
- ▶ L'uso di MdT deterministiche e non deterministiche porta alla definizione di classi di complessità distinte (nel caso in cui $P \neq NP$).
- ▶ **Cosa si può dire per la complessità spaziale?**

Complessità spaziale e non-determinismo

- ▶ Quanto può incidere l'uso del non-determinismo nel determinare la complessità spaziale del problema?

- ▶ **TEOREMA (Savitch).**

Sia $f: N \rightarrow R^+$ una funzione tale che $f(n) \geq n$.

$$NSPACE(f(n)) \subseteq SPACE(f^2(n)).$$

- ▶ Ogni linguaggio L deciso in spazio $O(f(n))$ da una MdT non-deterministica può essere deciso in spazio $O(f^2(n))$.

- ▶ **COROLLARIO**

$$PSPACE = NPSPACE$$

Confronto tra complessità temporale e spaziale

- ▶ Ogni MdT che decide in tempo polinomiale decide necessariamente anche in spazio polinomiale
- ▶ (Per ogni passo può usare una cella del nastro)

$$\mathbf{NP} \subseteq \mathbf{NPSPACE} = \mathbf{PSPACE}$$

Segue la seguente gerarchia di classi di complessità

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{NPSPACE} = \mathbf{PSPACE}$$

Per nessuna delle inclusioni della gerarchia **esiste una prova che l'inclusione sia propria.**

Confronto tra complessità temporale e spaziale

- ▶ Una classe di complessità temporale che includa **PSPACE**.
- ▶ Si assuma che una MdT decida un linguaggio con spazio $f(n)$.
- ▶ Quante sono le possibili configurazioni lunghezza $f(n)$ di una MdT con alfabeto interno Γ con $|\Gamma| = d$ e stati di controllo Q con $|Q| = m$
- ▶ Una configurazione include
 - ▶ Uno stato di controllo (m possibilità)
 - ▶ Il contenuto del nastro di $(d^{f(n)})$ possibilità
 - ▶ La posizione della testina sul nastro ($f(n)$ possibilità)
- ▶ Complessivamente
- ▶ $mf(n)d^{f(n)}$ di upper bound $f(n)2^{O(f(n))}$
- ▶ Una MdT per decidere non ha bisogno di visitare due volte la stessa configurazione
- ▶ **Una MdT che decide in spazio $f(n)$ decide in tempo $f(n)2^{O(f(n))}$**

Confronto tra complessità temporale e spaziale

- ▶ Una MdT che decide in spazio $f(n)$ decide in tempo $f(n)2^{O(f(n))}$
- ▶ Se un linguaggio L è deciso in spazio $O(n^k)$ ($L \in PSPACE$) allora L può essere deciso in tempo $O(2^{n^k})$

Sia la classe **EXPTIME** la classe dei problemi decisi da una MdT deterministica in tempo esponenziale, vale a dire

$$\text{EXPTIME} = \bigcup_k \text{TIME}(2^{n^k})$$

Vale dunque

$$NPSPACE = PSPACE \subseteq \text{EXPTIME}$$

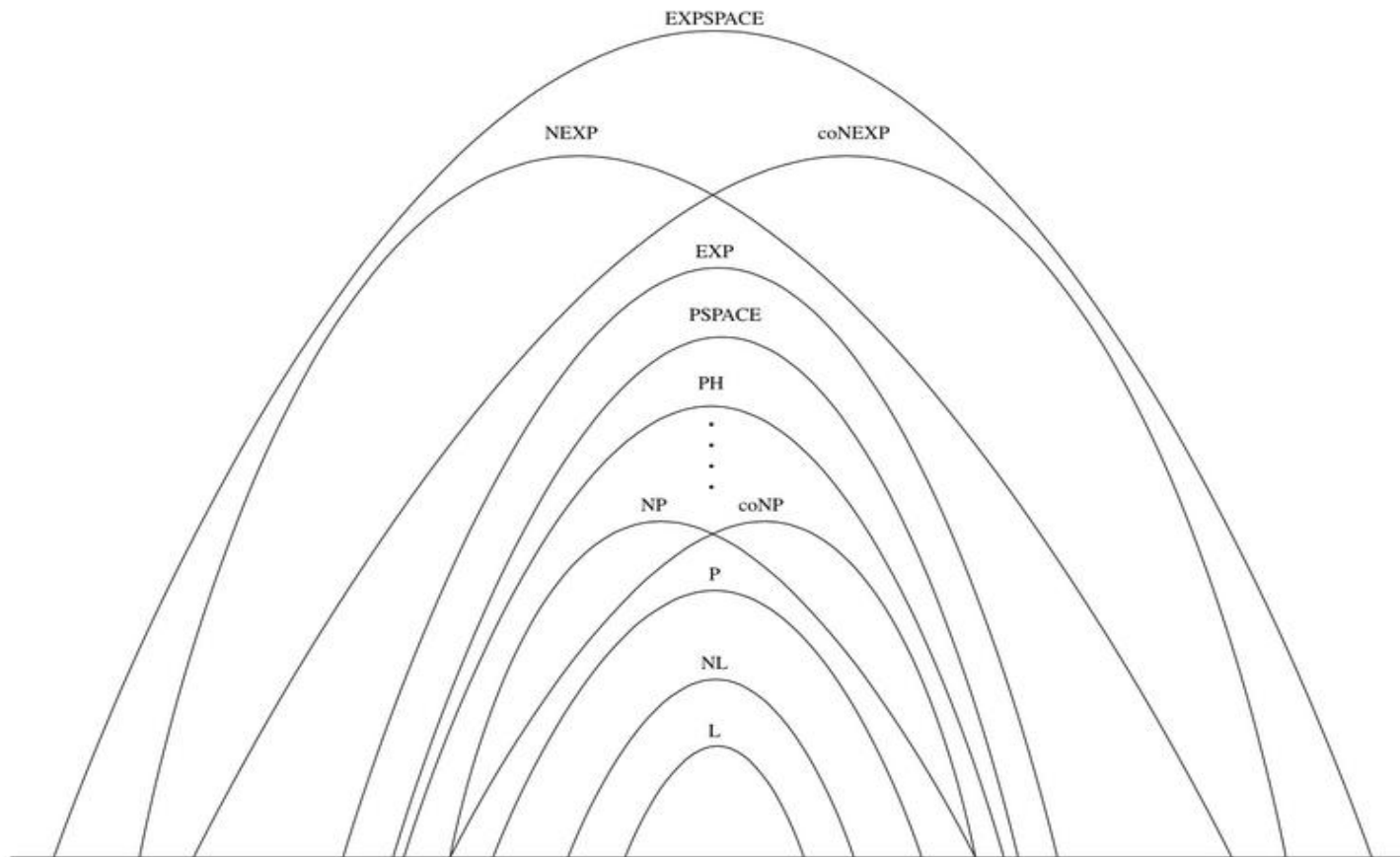
Confronto tra complessità temporale e spaziale

Segue la seguente gerarchia di classi di complessità

$$P \subseteq NP \subseteq NPSPACE = PSPACE \subseteq EXPTIME$$

- ▶ Per nessuna delle inclusioni della gerarchia **esiste una prova che l'inclusione sia propria.**
- ▶ E' solo noto che vale $P \subset EXPTIME$
- ▶ Una delle inclusioni è stretta ma non è noto quale sia stretta.

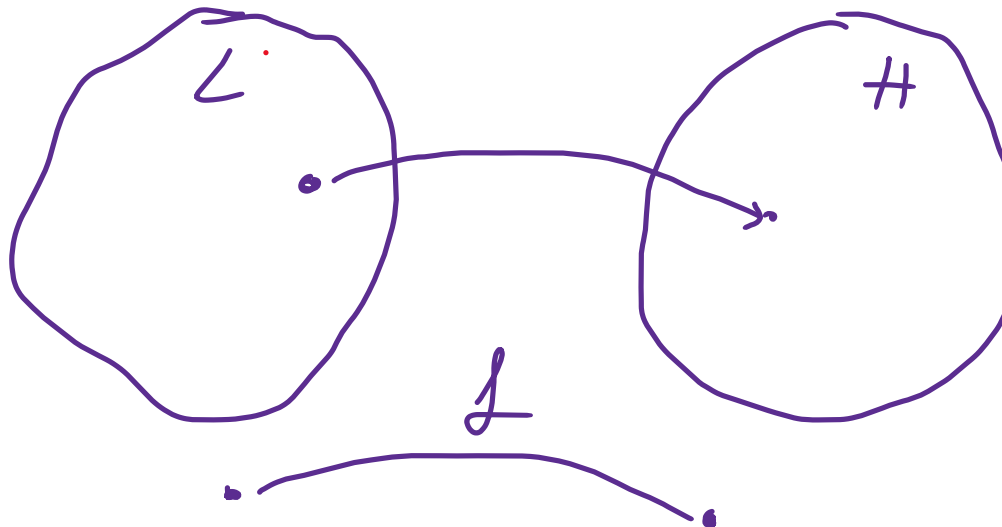
Gerarchie di complessità



PSPACE-completezza

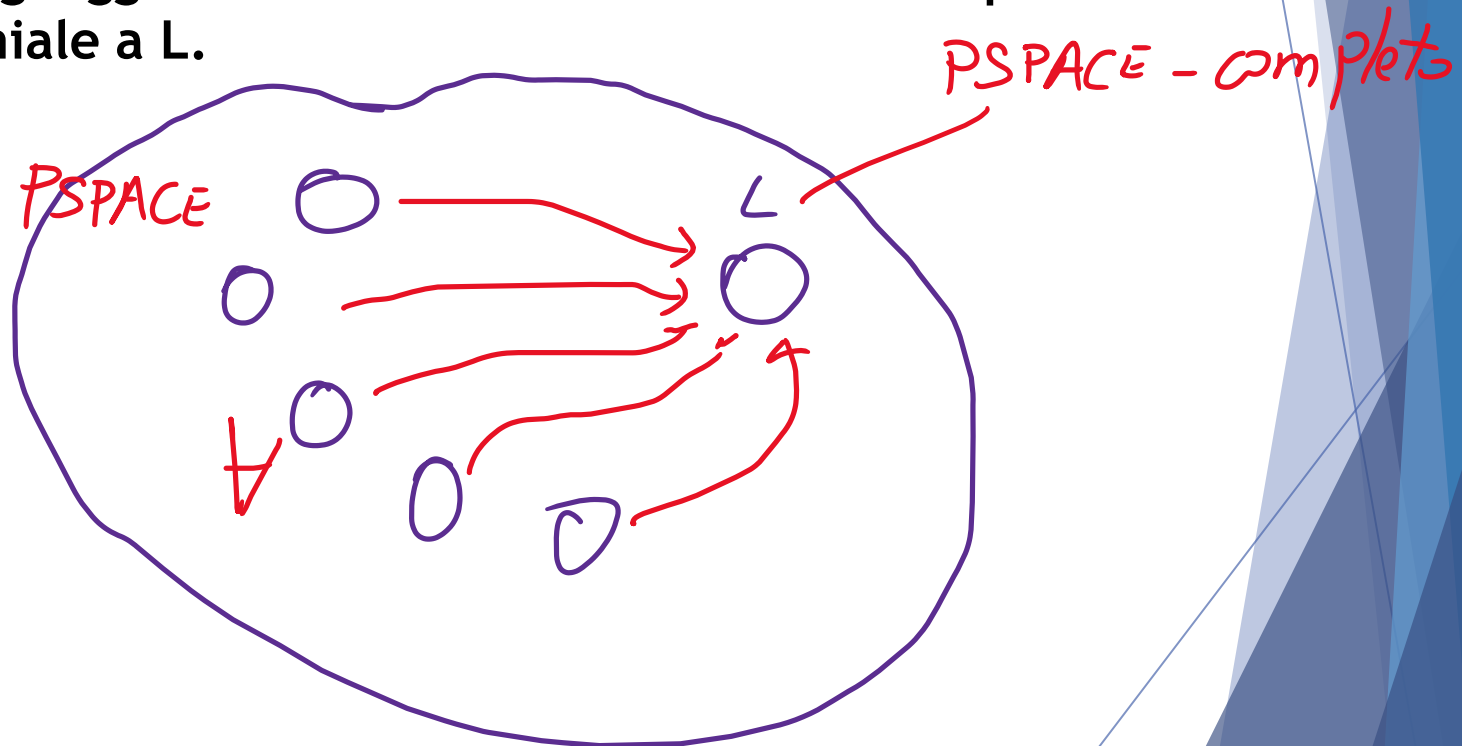
- **Riducibilità in tempo polinomiale:** Un linguaggio L è riducibile in tempo polinomiale ad un linguaggio H se esiste una funzione $f: \Sigma^* \rightarrow \Sigma^*$ computabile in tempo polinomiale tale che per ogni $w \in \Sigma^*$

$$w \in L \Leftrightarrow f(w) \in H$$



PSPACE-completezza

- **PSPACE-completezza:** Un linguaggio L è PSPACE-completo se
 1. $L \in PSPACE$;
 2. Ogni linguaggio $H \in PSPACE$ è riducibile in tempo polinomiale a L .



Un problema PSPACE-completo

- ▶ Formule booleane quantificate
- ▶ Sintassi
- ▶ $\varphi ::= TRUE \mid FALSE \mid x \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists x. \varphi \mid \forall x. \varphi$
- ▶ La valutazione della formula booleana richiede l'assegnazione di un valore di verità 0/1 (0 = False, 1=True) alle variabili
- ▶ Sia $\alpha: Var \rightarrow \{0,1\}$ la funzione che assegna un valore di verità alle variabili booleane
- ▶ La relazione $\alpha \models \varphi$ denota la soddisfacibilità di una formula rispetto alla valutazione delle variabili ed è definita come segue.
- ▶ $\alpha \models True$
- ▶ $\alpha \not\models False$
- ▶ $\alpha \models x$ se $\alpha(x) = 1$
- ▶ $\alpha \models \neg\varphi$ se $\alpha \not\models \varphi$
- ▶ $\alpha \models \varphi' \wedge \varphi''$ se $\alpha \models \varphi'$ e $\alpha \models \varphi''$
- ▶ $\alpha \models \varphi' \vee \varphi''$ se $\alpha \models \varphi'$ o $\alpha \models \varphi''$
- ▶ $\alpha \models \exists x. \varphi$ se o $\alpha[x \leftarrow 0] \models \varphi$ o $\alpha[x \leftarrow 1] \models \varphi$
- ▶ $\alpha \models \forall x. \varphi$ se $\alpha[x \leftarrow 0] \models \varphi$ e $\alpha[x \leftarrow 1] \models \varphi$

Un problema PSPACE-completo

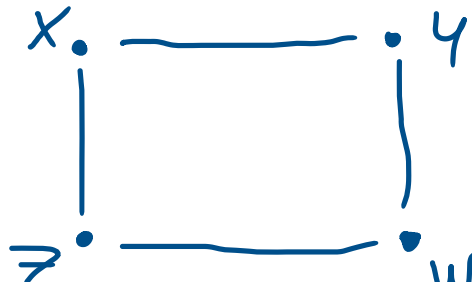
- ▶ **Formule booleane quantificate**
- ▶ Un'occorrenza della variabile x è **legata** se compare nello scopo di una quantificazione $\exists x$ o $\forall x$
- ▶ *esempio* $x \wedge \exists y \cdot y \wedge x$ ha x libera e y legata
- ▶ Una formula senza occorrenze di variabili libere è detta **sentenza**.
- ▶ **Osservazione.**
- ▶ La valutazione α serve solo a determinare la valutazione delle variabili libere.
- ▶ Se tutte le variabili sono legate da un quantificatore (è una sentenza) la valutazione della formula è indipendente dalla valutazione α usata per assegnare un valore di verità alle variabili.
- ▶ Ad una sentenza è vera o falsa indipendentemente da una funzione α di valutazione delle variabili.

$L_{TQBF} = \{ \langle \varphi \rangle : \varphi \text{ una sentenza vera della logica booleana quantificata} \}$

TEOREMA. Il linguaggio L_{TQBF} è PSPACE-completo.

Formule booleane quantificate

- **Esempio.** Dato il grafo di seguito si vuole colorare i nodi con 0 e 1 in modo che nodi adiacenti siano colorati diversamente.


$$\varphi = ((x \wedge \bar{y} \wedge \bar{z}) \vee (\bar{x} \wedge y \wedge z)) \wedge ((y \wedge \bar{w}) \vee (\bar{y} \wedge w)) \wedge ((z \wedge \bar{w}) \vee (\bar{z} \wedge w))$$

$\exists x. \exists z. \exists y. \exists w. \varphi$: esiste una colorazione

$\forall x. \exists z. \exists y. \exists w. \varphi$: per ogni colorazione di x esiste una colorazione degli altri nodi

Sono entrambe True

Formule booleane quantificate

- **Esempio.** Dato il grafo di seguito si vuole colorare i nodi con 0 e 1 in modo che nodi adiacenti siano colorati diversamente.



$$\varphi = ((x \wedge \bar{y} \wedge \bar{z}) \vee (\bar{x} \wedge y \wedge z)) \wedge ((y \wedge \bar{w}) \vee (\bar{y} \wedge w)) \wedge ((z \wedge \bar{w}) \vee (\bar{z} \wedge w))$$

$\forall x. \exists z. \exists y. \forall w. \varphi : \text{True} ?$

$\forall x. \exists z. \forall y. \exists w. \varphi : \text{True} ?$