Fondamenti Logici dell'Informatica

Corso di Laurea Magistrale in Informatica

Logica e Complessità

Fabio Zanasi

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Anno Accademico 2023-2024

La Complessità Computazionale

- Uno degli scopi più importanti che l'**informatica teorica** si prefigge è la classificazione dei *problemi computazionali* rispetto alla quantità di *risorse* che la loro risoluzione *richiede*.

La Complessità Computazionale

- Uno degli scopi più importanti che l'**informatica teorica** si prefigge è la classificazione dei *problemi computazionali* rispetto alla quantità di *risorse* che la loro risoluzione *richiede*.

Posso simulare Windows sul mio Mac?

La Complessità Computazionale

- Uno degli scopi più importanti che l'**informatica teorica** si prefigge è la classificazione dei *problemi computazionali* rispetto alla quantità di *risorse* che la loro risoluzione *richiede*.

Posso simulare Windows sul mio Mac?

C'è un algoritmo efficiente per la fattorizzazione in numeri primi?

La Complessità Computazionale

- Uno degli scopi più importanti che l'**informatica teorica** si prefigge è la classificazione dei *problemi computazionali* rispetto alla quantità di *risorse* che la loro risoluzione *richiede*.

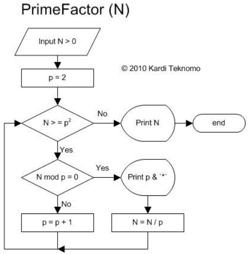
Posso simulare Windows sul mio Mac?

C'è un algoritmo efficiente per la fattorizzazione in numeri primi?

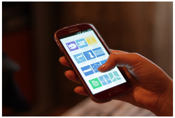
Quanto è protetto il mio conto online da un attacco informatico?

La Complessità Computazionale

- In questo modo, è possibile comprendere l'**intrinseca difficoltà** dei problemi computazionali che ci troviamo a risolvere.



- Parimenti, si può in questo modo studiare quali siano i **limiti** del calcolo con risorse *limitate*.



La Complessità Computazionale

- Per fare tutto questo, occorre però capire come formalizzare i concetti di:
    - *Problema Computazionale*;
    - *Algoritmo*;
    - *Risorse* e *limiti al loro uso*.

La Complessità Computazionale

- Per fare tutto questo, occorre però capire come formalizzare i concetti di:
  - *Problema Computazionale*;
  - *Algoritmo*;
  - *Risorse* e *limiti al loro uso*.
- La teoria risultante è la **complessità computazionale.**

Problema Computazionale

- Tradizionalmente, i problemi di cui si occupa la complessità computazionale sono il calcolo di funzioni nella forma $f : \mathbb{B} \to \mathbb{B}$ dove $\mathbb{B} = \{0, 1\}^*$ è l'insieme di tutte le *stringhe binarie*.

Problema Computazionale

- Tradizionalmente, i problemi di cui si occupa la complessità computazionale sono il calcolo di funzioni nella forma $f : \mathbb{B} \to \mathbb{B}$ dove $\mathbb{B} = \{0, 1\}^*$ è l'insieme di tutte le *stringhe binarie*.
- Perché?

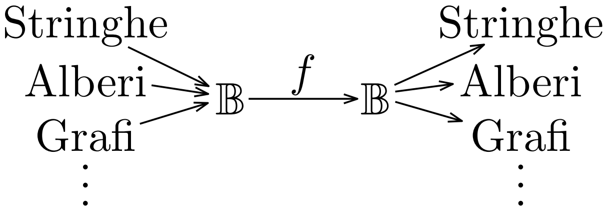
Problema Computazionale

- Tradizionalmente, i problemi di cui si occupa la complessità computazionale sono il calcolo di funzioni nella forma $f : \mathbb{B} \to \mathbb{B}$ dove $\mathbb{B} = \{0, 1\}^*$ è l'insieme di tutte le *stringhe binarie*.
- Perché?

$$\mathbb{B} \xrightarrow{f} \mathbb{B}$$

Problema Computazionale

- Tradizionalmente, i problemi di cui si occupa la complessità computazionale sono il calcolo di funzioni nella forma $f : \mathbb{B} \to \mathbb{B}$ dove $\mathbb{B} = \{0, 1\}^*$ è l'insieme di tutte le *stringhe binarie*.
- Perché?



Problema Computazionale

- Tradizionalmente, i problemi di cui si occupa la complessità computazionale sono il calcolo di funzioni nella forma $f : \mathbb{B} \to \mathbb{B}$ dove $\mathbb{B} = \{0,1\}^*$ è l'insieme di tutte le *stringhe binarie*.
- Perché?

$$\begin{tikzcd}
\text{Stringhe} \arrow[dr] & & & \text{Stringhe} \\
\text{Alberi} \arrow[r] & \mathbb{B} \arrow[r, "f"] & \mathbb{B} \arrow[ur] \arrow[r] \arrow[dr] & \text{Alberi} \\
\text{Grafi} \arrow[ur] & & & \text{Grafi} \\
\vdots & & & \vdots
\end{tikzcd}$$

- Spesso, ci si concentra sui problemi **decisionali**, cioè quelli in cui $f(\mathbb{B}) \subseteq \{0,1\}$.
  - Tali problemi sono in corrispondenza biunivoca con i *sottoinsiemi di* $\mathbb{B}$.

Dai Problemi agli Algoritmi

- Le funzioni incapsulano conoscenza *dichiarativa* (estensionale) e non *imperativa* (intensionale/algoritmica)
  - Se $f : \mathbb{B} \to \mathbb{B}$, allora $f \subseteq \mathbb{B} \times \mathbb{B}$.

Dai Problemi agli Algoritmi

- Le funzioni incapsulano conoscenza *dichiarativa* (estensionale) e non *imperativa* (intensionale/algoritmica)
  - Se $f : \mathbb{B} \to \mathbb{B}$, allora $f \subseteq \mathbb{B} \times \mathbb{B}$.
- Occorre cambiare punto di vista.

$$x \xmapsto{f} f(x)$$

Dai Problemi agli Algoritmi

- Le funzioni incapsulano conoscenza *dichiarativa* (estensionale) e non *imperativa* (intensionale/algoritmica)
  - Se $f : \mathbb{B} \rightarrow \mathbb{B}$, allora $f \subseteq \mathbb{B} \times \mathbb{B}$.
- Occorre cambiare punto di vista.

$$\begin{tikzcd} & x \arrow[r, mapsto, "f"] \arrow[dl] & f(x) \\ I(x) & & \end{tikzcd}$$

Dai Problemi agli Algoritmi

- Le funzioni incapsulano conoscenza *dichiarativa* (estensionale) e non *imperativa* (intensionale/algoritmica)
  - Se $f : \mathbb{B} \to \mathbb{B}$, allora $f \subseteq \mathbb{B} \times \mathbb{B}$.
- Occorre cambiare punto di vista.

$$\begin{tikzcd}
 & x \arrow[rr, mapsto, "f"] \arrow[dl] & & f(x) & \\
I(x) \arrow[rr] & & \cdots \arrow[rr] & & F(x) \arrow[ul]
\end{tikzcd}$$

Dai Problemi agli Algoritmi

- Le funzioni incapsulano conoscenza *dichiarativa* (estensionale) e non *imperativa* (intensionale/algoritmica)
  - Se $f : \mathbb{B} \to \mathbb{B}$, allora $f \subseteq \mathbb{B} \times \mathbb{B}$.
- Occorre cambiare punto di vista.

$$\begin{tikzcd}
 & x \arrow[r, mapsto, "f"] \arrow[dl] & f(x) & \\
I(x) \arrow[r, "\mathcal{A}"] & \cdots \arrow[r, "\mathcal{A}"] & F(x) \arrow[u] &
\end{tikzcd}$$

Dai Problemi agli Algoritmi

- Le funzioni incapsulano conoscenza *dichiarativa* (estensionale) e non *imperativa* (intensionale/algoritmica)
  - Se $f : \mathbb{B} \to \mathbb{B}$, allora $f \subseteq \mathbb{B} \times \mathbb{B}$.
- Occorre cambiare punto di vista.

$$\begin{tikzcd}
 & x \arrow[rr, mapsto, "f"] \arrow[dl] & & f(x) & \\
I(x) \arrow[rr, "\mathcal{A}"] & & \cdots \arrow[rr, "\mathcal{A}"] & & F(x) \arrow[ul]
\end{tikzcd}$$

- La trasformazione $\xrightarrow{\mathcal{A}}$ *non* deve dipendere da x, e ogni passo deve essere *elementare*.

Dai Problemi agli Algoritmi

- Le funzioni incapsulano conoscenza *dichiarativa* (estensionale) e non *imperativa* (intensionale/algoritmica)
  - Se $f : \mathbb{B} \to \mathbb{B}$, allora $f \subseteq \mathbb{B} \times \mathbb{B}$.
- Occorre cambiare punto di vista.

$$
\begin{tikzcd}
 & x \arrow[rr, mapsto, "f"] \arrow[dl] & & f(x) \\
I(x) \arrow[r, "\mathcal{A}"] & \cdots \arrow[rr, "\mathcal{A}"] & & F(x) \arrow[u]
\end{tikzcd}
$$

- La trasformazione $\xrightarrow{\mathcal{A}}$ *non* deve dipendere da x, e ogni passo deve essere *elementare*.
- In tal caso $\mathcal{A}$ *calcola* f e scriviamo $[\![\mathcal{A}]\!] = f$.
- Agli algoritmi può essere data natura formale in tanti modi diversi.
  - *Macchine di Turing.*
  - *Random Access Machines.*
  - $\ldots$

Misurare l'Uso delle Risorse

- Quali potrebbero essere le risorse di calcolo d'interesse?

Misurare l'Uso delle Risorse

- Quali potrebbero essere le risorse di calcolo d'interesse?
- **Tempo**.
  - Il tempo è modellato come il *numero* di transizioni necessarie ad arrivare a $F(x)$ partendo da $I(x)$.
  - Certamente in questo modo otteniamo un'idealizzazione del *vero* tempo di calcolo, il quale ovviamente dipende dalla complessità di ogni singolo passo di calcolo.
  - Il tempo che l'algoritmo $\mathcal{A}$ impiega sull'input x è indicato con $\mathsf{TIME}_{\mathcal{A}}(x)$

Misurare l'Uso delle Risorse

- Quali potrebbero essere le risorse di calcolo d'interesse?
- **Tempo**.
  - Il tempo è modellato come il *numero* di transizioni necessarie ad arrivare a $F(x)$ partendo da $I(x)$.
  - Certamente in questo modo otteniamo un'idealizzazione del *vero* tempo di calcolo, il quale ovviamente dipende dalla complessità di ogni singolo passo di calcolo.
  - Il tempo che l'algoritmo $\mathcal{A}$ impiega sull'input x è indicato con $\mathsf{TIME}_{\mathcal{A}}(x)$
- **Spazio**.
  - Lo spazio di calcolo è invece modellato come la *massima* lunghezza dei risultati intermedi necessari ad ottenere $F(x)$ a partire da $I(x)$.
  - Non si tiene conto né dello spazio necessario a tener traccia di x, né di quello necessario a tener traccia di $f(x)$.
  - Lo spazio che l'algoritmo $\mathcal{A}$ impiega sull'input x è indicato con $\mathsf{SPACE}_{\mathcal{A}}(x)$

Misurare l'Uso delle Risorse

- Quali potrebbero essere le risorse di calcolo d'interesse?
- **Tempo**.
  - Il tempo è modellato come il *numero* di transizioni necessarie ad arrivare a $F(x)$ partendo da $I(x)$.
  - Certamente in questo modo otteniamo un'idealizzazione del *vero* tempo di calcolo, il quale ovviamente dipende dalla complessità di ogni singolo passo di calcolo.
  - Il tempo che l'algoritmo $\mathcal{A}$ impiega sull'input x è indicato con $\mathsf{TIME}_{\mathcal{A}}(x)$
- **Spazio**.
  - Lo spazio di calcolo è invece modellato come la *massima* lunghezza dei risultati intermedi necessari ad ottenere $F(x)$ a partire da $I(x)$.
  - Non si tiene conto né dello spazio necessario a tener traccia di x, né di quello necessario a tener traccia di $f(x)$.
  - Lo spazio che l'algoritmo $\mathcal{A}$ impiega sull'input x è indicato con $\mathsf{SPACE}_{\mathcal{A}}(x)$
- La *lunghezza* di una stringa x è indicata con $|x|$.

Classi di Complessità

- Possiamo prima di tutto definire delle **classi concrete**, parametriche su una funzione $g : \mathbb{N} \to \mathbb{N}$.

$$
\begin{aligned}
DTIME(g) &= \{L \subseteq \mathbb{B} \mid \exists \mathcal{A}.[\![\mathcal{A}]\!] = L \wedge \forall x.\mathsf{TIME}_{\mathcal{A}}(x) \leq g(|x|)\};\\
DSPACE(g) &= \{L \subseteq \mathbb{B} \mid \exists \mathcal{A}.[\![\mathcal{A}]\!] = L \wedge \forall x.\mathsf{SPACE}_{\mathcal{A}}(x) \leq g(|x|)\}.
\end{aligned}
$$

- Possiamo poi definire vere e proprie **classi di complessità**, che permettono una classificazione dei problemi più informativa:

$$
\mathsf{P} = \bigcup_{g \in \mathsf{POLY}} DTIME(g) \qquad \mathsf{PSPACE} = \bigcup_{g \in \mathsf{POLY}} DSPACE(g)
$$

$$
\mathsf{L} = \bigcup_{g \in \mathsf{LOGA}} DSPACE(g)
$$

dove POLY è la classe dei polinomi e LOGA è la classe delle funzioni logaritmiche.

Nondeterminismo

- Se il problema computazionale è decisionale, ha senso pensare ad algoritmi che *non* siano deterministici.

Nondeterminismo

- Se il problema computazionale è decisionale, ha senso pensare ad algoritmi che *non* siano deterministici.
- Per esempio, potremmo avere un comportamento puramente nondeterministico.

Nondeterminismo

- Se il problema computazionale è decisionale, ha senso pensare ad algoritmi che *non* siano deterministici.
- Per esempio, potremmo avere un comportamento puramente nondeterministico.

$$x \xmapsto{f} f(x)$$

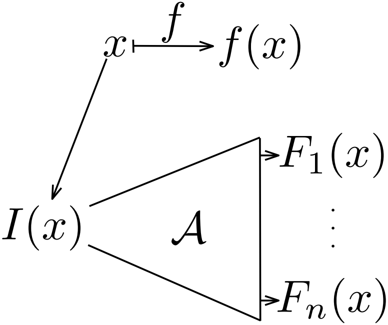
Nondeterminismo

- Se il problema computazionale è decisionale, ha senso pensare ad algoritmi che *non* siano deterministici.
- Per esempio, potremmo avere un comportamento puramente nondeterministico.

$$\begin{tikzcd} & x \arrow[r, mapsto, "f"] \arrow[dl] & f(x) \\ I(x) & & \end{tikzcd}$$

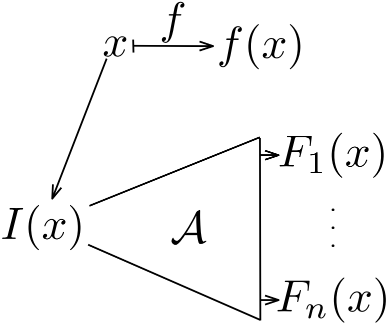
Nondeterminismo

- Se il problema computazionale è decisionale, ha senso pensare ad algoritmi che *non* siano deterministici.
- Per esempio, potremmo avere un comportamento puramente nondeterministico.



Nondeterminismo

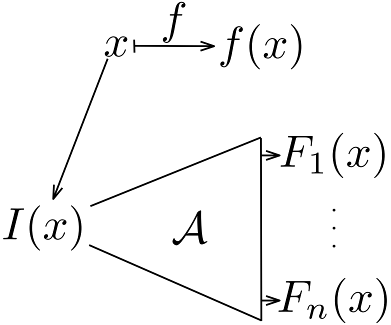
- Se il problema computazionale è decisionale, ha senso pensare ad algoritmi che *non* siano deterministici.
- Per esempio, potremmo avere un comportamento puramente nondeterministico.



$f(x) = 1$
se e solo se
esiste i dove
$F_i(x)$ è finale.

Nondeterminismo

- Se il problema computazionale è decisionale, ha senso pensare ad algoritmi che *non* siano deterministici.
- Per esempio, potremmo avere un comportamento puramente nondeterministico.

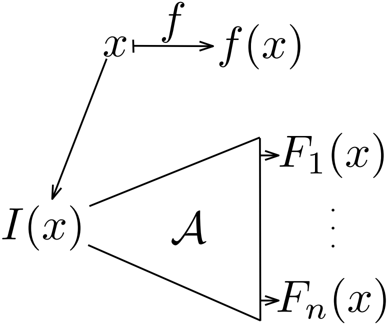


$f(x) = 1$
se e solo se
esiste i dove
$F_i(x)$ è finale.

- In tal caso $\mathcal{A}$ *decide* f e scriviamo $[\![\mathcal{A}]\!] = f$.

Nondeterminismo

- Se il problema computazionale è decisionale, ha senso pensare ad algoritmi che *non* siano deterministici.
- Per esempio, potremmo avere un comportamento puramente nondeterministico.



$f(x) = 1$
se e solo se
esiste i dove
$F_i(x)$ è finale.

- In tal caso $\mathcal{A}$ *decide* f e scriviamo $[\![\mathcal{A}]\!] = f$.

La classe NP

- Si possono definire poi $NDTIME(g)$ e NP, seguendo lo stesso schema visto precedentemente per gli algoritmi deterministici.

$$\begin{aligned} NDTIME(g) = \{L \subseteq \mathbb{B} \mid \exists \mathcal{A} \text{ non-det } . \llbracket \mathcal{A} \rrbracket = L \\ \wedge \forall x. \mathsf{TIME}_{\mathcal{A}}(x) \leq g(|x|)\} \end{aligned}$$

$$\mathsf{NP} = \bigcup_{g \in \mathsf{POLY}} NDTIME(g)$$

La classe NP

- Si possono definire poi $NDTIME(g)$ e NP, seguendo lo stesso schema visto precedentemente per gli algoritmi deterministici.

$$
\begin{aligned}
NDTIME(g) = \{L \subseteq \mathbb{B} \mid \exists \mathcal{A} \text{ non-det } . \llbracket \mathcal{A} \rrbracket = L \\
\wedge \forall x . \mathsf{TIME}_{\mathcal{A}}(x) \leq g(|x|)\}
\end{aligned}
$$

$$
\mathsf{NP} = \bigcup_{g \in \mathsf{POLY}} NDTIME(g)
$$

- Una caratterizzazione equivalente di NP è la classe dei linguaggi L per i quali una prova (computazione) di $x \in L$ può essere ispezionata da un algoritmo deterministico in tempo polinomiale.

Il modello di calcolo: ha rilevanza?

- Quando si parla di classi di complessità — nello specifico, di tempo di calcolo di un algoritmo — sembrerebbe importante specificare rispetto a quale **modello di calcolo**.

Il modello di calcolo: ha rilevanza?

- Quando si parla di classi di complessità — nello specifico, di tempo di calcolo di un algoritmo — sembrerebbe importante specificare rispetto a quale **modello di calcolo**.
- Di primo acchito, una macchina di Turing pare molto più inefficiente di un programma C eseguito su un moderno mainframe...

Il modello di calcolo: ha rilevanza?

- Quando si parla di classi di complessità — nello specifico, di tempo di calcolo di un algoritmo — sembrerebbe importante specificare rispetto a quale **modello di calcolo**.
- Di primo acchito, una macchina di Turing pare molto più inefficiente di un programma C eseguito su un moderno mainframe...
- ...invece no!

Congettura (Tesi di Church-Turing forte)

Qualsiasi modello di calcolo fisicamente realizzabile può essere simulato efficientemente (con slowdown al più polinomiale) da una macchina di Turing.

Il modello di calcolo: ha rilevanza?

- Quando si parla di classi di complessità — nello specifico, di tempo di calcolo di un algoritmo — sembrerebbe importante specificare rispetto a quale **modello di calcolo**.
- Di primo acchito, una macchina di Turing pare molto più inefficiente di un programma C eseguito su un moderno mainframe...
- ...invece no!

Congettura (Tesi di Church-Turing forte)

Qualsiasi modello di calcolo fisicamente realizzabile può essere simulato efficientemente (con slowdown al più polinomiale) da una macchina di Turing.

- La congettura suggerisce dunque che il modello di calcolo, sorprendentemente, **non conta**.

Una Teoria Giovane

- Con tecniche abbastanza semplici si riesce a dimostrare che

$$\mathsf{L} \subseteq \mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE}.$$

Una Teoria Giovane

- Con tecniche abbastanza semplici si riesce a dimostrare che

$$\mathsf{L} \subseteq \mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE}.$$

- Con tecniche altrettanto semplici, si può concludere che $\mathsf{L} \subsetneq \mathsf{PSPACE}$.

Una Teoria Giovane

- Con tecniche abbastanza semplici si riesce a dimostrare che

$$\mathsf{L} \subseteq \mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE}.$$

- Con tecniche altrettanto semplici, si può concludere che $\mathsf{L} \subsetneq \mathsf{PSPACE}$.
- A tutt'oggi, non è però chiaro **quali** tra le tre inclusioni di cui sopra siano strette. Almeno *una* deve essere stretta, ma potrebbero anche essere *tutte* strette.

Una Teoria Giovane

- Con tecniche abbastanza semplici si riesce a dimostrare che
$$\mathsf{L} \subseteq \mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE}.$$
- Con tecniche altrettanto semplici, si può concludere che $\mathsf{L} \subsetneq \mathsf{PSPACE}$.
- A tutt'oggi, non è però chiaro **quali** tra le tre inclusioni di cui sopra siano strette. Almeno *una* deve essere stretta, ma potrebbero anche essere *tutte* strette.
- Particolarmente dibattuto è: $P \neq NP$?. Possiamo dimostrare che *trovare* la soluzione ad un problema è più difficile che *verificare* se una data soluzione è corretta?

Una Teoria Giovane

- Con tecniche abbastanza semplici si riesce a dimostrare che

$$\mathsf{L} \subseteq \mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE}.$$

- Con tecniche altrettanto semplici, si può concludere che $\mathsf{L} \subsetneq \mathsf{PSPACE}$.
- A tutt'oggi, non è però chiaro **quali** tra le tre inclusioni di cui sopra siano strette. Almeno *una* deve essere stretta, ma potrebbero anche essere *tutte* strette.
- Particolarmente dibattuto è: $P \neq NP$?. Possiamo dimostrare che *trovare* la soluzione ad un problema è più difficile che *verificare* se una data soluzione è corretta?
- Più di quarant'anni di sforzi non hanno portato ad *alcun* decisivo progresso nella risoluzione di questo problema.

Una Teoria Giovane

- Con tecniche abbastanza semplici si riesce a dimostrare che
$$\mathsf{L} \subseteq \mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE}.$$
- Con tecniche altrettanto semplici, si può concludere che $\mathsf{L} \subsetneq \mathsf{PSPACE}$.
- A tutt'oggi, non è però chiaro **quali** tra le tre inclusioni di cui sopra siano strette. Almeno *una* deve essere stretta, ma potrebbero anche essere *tutte* strette.
- Particolarmente dibattuto è: $P \neq NP$?. Possiamo dimostrare che *trovare* la soluzione ad un problema è più difficile che *verificare* se una data soluzione è corretta?
- Più di quarant'anni di sforzi non hanno portato ad *alcun* decisivo progresso nella risoluzione di questo problema.
- La comunità scientifica è convinta che servano tecniche **diverse** da quelle puramente combinatorie.

Complessità Descrittiva: Idea

- Le misure di **tempo** e **spazio** nascono da esigenze di natura ingegneristica. Sono modi naturali di ragionare sulle trasformazioni che avvengono in un modello **fisico**.

Complessità Descrittiva: Idea

- Le misure di **tempo** e **spazio** nascono da esigenze di natura ingegneristica. Sono modi naturali di ragionare sulle trasformazioni che avvengono in un modello **fisico**.
- Da un punto di vista **matematico**, queste misure non sono del tutto soddisfacenti. Gli strumenti di analisi tipici della matematica astratta (logica, algebra, ...) si applicano solo in maniera limitata.

Complessità Descrittiva: Idea

- Le misure di **tempo** e **spazio** nascono da esigenze di natura ingegneristica. Sono modi naturali di ragionare sulle trasformazioni che avvengono in un modello **fisico**.
- Da un punto di vista **matematico**, queste misure non sono del tutto soddisfacenti. Gli strumenti di analisi tipici della matematica astratta (logica, algebra, ...) si applicano solo in maniera limitata.
- Da questa esigenza nasce l'idea di **caratterizzare** le classi di complessità in termini di **linguaggi logici**.

Complessità Descrittiva: Idea

- Le misure di **tempo** e **spazio** nascono da esigenze di natura ingegneristica. Sono modi naturali di ragionare sulle trasformazioni che avvengono in un modello **fisico**.
- Da un punto di vista **matematico**, queste misure non sono del tutto soddisfacenti. Gli strumenti di analisi tipici della matematica astratta (logica, algebra, ...) si applicano solo in maniera limitata.
- Da questa esigenza nasce l'idea di **caratterizzare** le classi di complessità in termini di **linguaggi logici**.
- La caratterizzazione logica consente di studiare più facilmente proprietà delle classi di complessità (ad esempio proprietà di chiusura) e di analizzare la complessità di software basati su linguaggi logici (ad esempio i linguaggi di interrogazione dei database).

- La logica predicativa offre un modo di parlare di *insiemi* di oggetti, ovvero di *linguaggi*.

- La logica predicativa offre un modo di parlare di *insiemi* di oggetti, ovvero di *linguaggi*.
- Un esempio di oggetti sono i grafi. In che modo la logica predicativa permette di ragionare sui grafi?

Logica per lo Studio di Modelli di Calcolo: Caso dei Grafi

- La logica predicativa offre un modo di parlare di *insiemi* di oggetti, ovvero di *linguaggi*.
- Un esempio di oggetti sono i grafi. In che modo la logica predicativa permette di ragionare sui grafi?
- Supponiamo di utilizzare un vocabolario costituito dal simbolo binario di uguaglianza $=$ e da un simbolo binario E.

Logica per lo Studio di Modelli di Calcolo: Caso dei Grafi

- La logica predicativa offre un modo di parlare di *insiemi* di oggetti, ovvero di *linguaggi*.
- Un esempio di oggetti sono i grafi. In che modo la logica predicativa permette di ragionare sui grafi?
- Supponiamo di utilizzare un vocabolario costituito dal simbolo binario di uguaglianza $=$ e da un simbolo binario E.
- Consideriamo universi finiti nella forma $\mathcal{A}_n = \{1, \ldots, n\}$. Un'interpretazione I per tale universo consta, fondamentalmente, di una relazione binaria E_I su $\mathcal{A}_n$.
  - L'uguaglianza è sempre interpretata con la relazione identità.

Logica per lo Studio di Modelli di Calcolo: Caso dei Grafi

- La logica predicativa offre un modo di parlare di *insiemi* di oggetti, ovvero di *linguaggi*.
- Un esempio di oggetti sono i grafi. In che modo la logica predicativa permette di ragionare sui grafi?
- Supponiamo di utilizzare un vocabolario costituito dal simbolo binario di uguaglianza $=$ e da un simbolo binario E.
- Consideriamo universi finiti nella forma $\mathcal{A}_n = \{1, \ldots, n\}$. Un'interpretazione I per tale universo consta, fondamentalmente, di una relazione binaria E_I su $\mathcal{A}_n$.
  - L'uguaglianza è sempre interpretata con la relazione identità.
- $(\mathcal{A}_n, I)$ è **un grafo**.

- Ogni formula nel vocabolario costituito da E e dall'uguaglianza può quindi essere vista come un **riconoscitore di grafi**.

Logica per lo Studio di Modelli di Calcolo: Caso dei Grafi

- Ogni formula nel vocabolario costituito da E e dall'uguaglianza può quindi essere vista come un **riconoscitore di grafi**.
- Ad esempio possiamo esprimere con la seguente formula una proprietà dei grafi:

$$
\begin{aligned}
BIN \equiv\ & \forall x.\exists y.\exists z.\forall w.(y \neq z \wedge E(x,y) \wedge E(x,z)) \\
& \wedge (E(x,w) \rightarrow w = y \vee w = z).
\end{aligned}
$$

Logica per lo Studio di Modelli di Calcolo: Caso dei Grafi

- Ogni formula nel vocabolario costituito da E e dall'uguaglianza può quindi essere vista come un **riconoscitore di grafi**.
- Ad esempio possiamo esprimere con la seguente formula una proprietà dei grafi:

$$\begin{aligned} BIN \ &\equiv \forall x. \exists y. \exists z. \forall w. (y \neq z \wedge E(x,y) \wedge E(x,z)) \\ &\qquad \wedge (E(x,w) \rightarrow w = y \vee w = z). \end{aligned}$$

- $(\mathcal{A}_n, I) \vDash BIN$ se e solo se $(\mathcal{A}_n, I)$ è un grafo binario completo.

Ritorno alla Complessità Descrittiva

Formule Logiche $\longleftrightarrow$ Linguaggi

Ritorno alla Complessità Descrittiva

$$
\begin{array}{ccc}
\text{Formule Logiche} & \longleftrightarrow & \text{Linguaggi} \\
\text{Interpretazioni} & \longleftrightarrow & \text{Stringhe}
\end{array}
$$

Ritorno alla Complessità Descrittiva

Formule Logiche $\longleftrightarrow$ Linguaggi

Interpretazioni $\longleftrightarrow$ Stringhe

- Esistono logiche che *catturano*, che *caratterizzano*, classi di complessità importanti come P o NP?
- È possibile capire qualcosa di profondo circa le classi di complessità indagandone la natura *logica*?
- Il resto di questa parte del Corso sarà dedicato a dare una risposta alla prima di queste domande.

Interpretazioni come Stringhe

- Per dare concretezza alla complessità descrittiva, occorre prima di tutto capire come le interpretazioni possano essere descritte in modo compatto.

Interpretazioni come Stringhe

- Per dare concretezza alla complessità descrittiva, occorre prima di tutto capire come le interpretazioni possano essere descritte in modo compatto.
- Supponiamo di lavorare con l'universo finito $\mathcal{A}_n = \{1, \ldots, n\}$.

Interpretazioni come Stringhe

- Per dare concretezza alla complessità descrittiva, occorre prima di tutto capire come le interpretazioni possano essere descritte in modo compatto.
- Supponiamo di lavorare con l'universo finito $\mathcal{A}_n = \{1, \ldots, n\}$.
- Supponiamo che un vocabolario sia costituito da m simboli predicativi $P_1, \ldots, P_m$ e da k simboli di funzione $f_1, \ldots, f_k$, tutti di arietà 0.

Interpretazioni come Stringhe

- Per dare concretezza alla complessità descrittiva, occorre prima di tutto capire come le interpretazioni possano essere descritte in modo compatto.
- Supponiamo di lavorare con l'universo finito $\mathcal{A}_n = \{1, \ldots, n\}$.
- Supponiamo che un vocabolario sia costituito da m simboli predicativi $P_1, \ldots, P_m$ e da k simboli di funzione $f_1, \ldots, f_k$, tutti di arietà 0.
- Una qualunque interpretazione I per tale vocabolario può essere quindi descritta da una stringa $\mathbf{bin}^n(I) \in \mathbb{B}$ dove

$$\mathbf{bin}^n(I) = \mathbf{bin}^n(P_1) \ldots \mathbf{bin}^n(P_m)\mathbf{bin}^n(f_1) \ldots \mathbf{bin}^n(f_k)$$

  e:
  - Per ogni $1 \leq i \leq m$, la stringa $\mathbf{bin}^n(P_i)$ ha lunghezza pari a $n^{\mathbf{ar}(P_i)}$, dove $\mathbf{ar}(P_i)$ è l'arietà di P_i. Tale stringa specifica quando ogni possibile tupla fa parte di $(P_i)_I$ oppure no.
  - Per ogni $1 \leq i \leq k$, la stringa $\mathbf{bin}^n(f_i)$ ha lunghezza pari a $\lceil \log_2(n) \rceil$ e specifica quale elemento di $\mathcal{A}_n$ interpreta f_i.

Formule come Funzioni

- Una formula predicativa F si dice chiusa quando nessuna variabile occorre libera in F.
- Ad ogni formula predicativa chiusa F si può far corrispondere un sottoinsieme $\mathbf{struct}(F)$ come segue:

$$\mathbf{struct}(F) = \{\mathbf{bin}^n(I) \mid (\mathcal{A}_n, I) \models F\} \subseteq \mathbb{B}.$$

- A questo punto ci si può chiedere quale sia la classe di linguaggi che una certa logica caratterizza.
- Nel caso della logica predicativa abbiamo per esempio

$$\mathsf{FO} = \{\mathbf{struct}(F) \mid F \text{ è una formula predicativa chiusa}\}$$

La Logica Predicativa è Interessante?

- Una domanda a questo punto molto interessante è la seguente: FO coincide con una classe interessante?

La Logica Predicativa è Interessante?

- Una domanda a questo punto molto interessante è la seguente: FO coincide con una classe interessante?
- Per fare in modo che la logica predicativa abbia un minimo di potere espressivo, occorre assumere che il vocabolario includa:
  - Tre simboli funzionali di arietà nulla chiamati 0,1, max, interpretati nel modo ovvio.
  - Due simboli predicativi binari $\leq$ e BIT, anch'essi interpretati nel modo ovvio.

La Logica Predicativa è Interessante?

- Una domanda a questo punto molto interessante è la seguente: FO coincide con una classe interessante?
- Per fare in modo che la logica predicativa abbia un minimo di potere espressivo, occorre assumere che il vocabolario includa:
  - Tre simboli funzionali di arietà nulla chiamati 0,1, max, interpretati nel modo ovvio.
  - Due simboli predicativi binari $\leq$ e BIT, anch'essi interpretati nel modo ovvio.

Lemma

$\mathsf{FO} \subseteq \mathsf{L}$.

Lemma

$PARITY \notin \mathsf{FO}$.

Teorema

$\mathsf{FO} \subsetneq \mathsf{L}$.

Logica Predicativa del Secondo Ordine

- Conviene a questo punto cercare una logica più espressiva.
- Una scelta naturale è quella di considerare un'estensione delle formule con *variabili* al second'ordine, che indicano una qualunque *relazione*, anziché un *oggetto*:

$$F ::= \ldots \;\Big|\; X^n(t_1, \ldots, t_n) \;\Big|\; \exists X^n.F \;\Big|\; \forall X^n.F.$$

- La semantica di queste formule segue quella della logica predicativa, ma occorre che ξ assegni una relazione n-aria ad ogni variabile X^n.
- In questo modo:

$$\begin{array}{lll}
(\mathcal{A}, I), \xi \models X^n(t_1, \ldots, t_n) & \text{sse} & ([\![t_1]\!]_\xi^{(\mathcal{A},I)}, \ldots, [\![t_n]\!]_\xi^{(\mathcal{A},I)}) \in \xi(X^n) \\
(\mathcal{A}, I), \xi \models \exists X^n.F & \text{sse} & (\mathcal{A}, I), \xi[X^n := \mathcal{R}] \models F \\
 & & \quad \text{per qualche } \mathcal{R} \subseteq \mathcal{A}^n \\
(\mathcal{A}, I), \xi \models \forall X^n.F & \text{sse} & (\mathcal{A}, I), \xi[X^n := \mathcal{R}] \models F \\
 & & \quad \text{per tutte le } \mathcal{R} \subseteq \mathcal{A}^n \\
 & \vdots &
\end{array}$$

Il Teorema di Fagin

- La logica al second'ordine è troppo potente per i nostri scopi.

Il Teorema di Fagin

- La logica al second'ordine è troppo potente per i nostri scopi.
- Ne esiste però un frammento molto interessante, che chiamiamo **logica al second'ordine esistenziale**, le cui formule sono le formule che possono essere scritte nella forma
$$\exists X^{n_1}.\ldots.\exists X^{n_m}.F$$
dove F è una formula predicativa *al prim'ordine*.

Il Teorema di Fagin

- La logica al second'ordine è troppo potente per i nostri scopi.
- Ne esiste però un frammento molto interessante, che chiamiamo **logica al second'ordine esistenziale**, le cui formule sono le formule che possono essere scritte nella forma
$$\exists X^{n_1}.\dots.\exists X^{n_m}.F$$
dove F è una formula predicativa *al prim'ordine*.
- A questo punto è facile generalizzare quanto visto in precedenza per la logica predicativa:

$$\exists SO = \{\mathbf{struct}(F) \mid F \text{ è una formula al second'ordine esistenziale}\}$$

Il Teorema di Fagin

- La logica al second'ordine è troppo potente per i nostri scopi.
- Ne esiste però un frammento molto interessante, che chiamiamo **logica al second'ordine esistenziale**, le cui formule sono le formule che possono essere scritte nella forma
$$\exists X^{n_1}.\ldots.\exists X^{n_m}.F$$
dove F è una formula predicativa *al prim'ordine*.
- A questo punto è facile generalizzare quanto visto in precedenza per la logica predicativa:
$$\exists SO = \{\mathbf{struct}(F) \mid F \text{ è una formula al second'ordine esistenziale}\}$$

Teorema (Fagin)

$\exists SO = \mathsf{NP}$

E il Tempo Polinomiale Deterministico?

- La logica predicativa è, come abbiamo visto, troppo debole per i nostri scopi.
- D'altro canto, quella al second'ordine risulta troppo potente.

E il Tempo Polinomiale Deterministico?

- La logica predicativa è, come abbiamo visto, troppo debole per i nostri scopi.
- D'altro canto, quella al second'ordine risulta troppo potente.
- Cosa serve alla logica predicativa per catturare *esattamente* il tempo polinomiale deterministico?

E il Tempo Polinomiale Deterministico?

- La logica predicativa è, come abbiamo visto, troppo debole per i nostri scopi.
- D'altro canto, quella al second'ordine risulta troppo potente.
- Cosa serve alla logica predicativa per catturare *esattamente* il tempo polinomiale deterministico?
- Consieriamo il vocabolario relativo ai grafi, ossia quello in cui l'unico simbolo relazionale E è binario e rappresenta l'adiacenza tra nodi.
- Il fatto che un nodo y sia *raggiungibile* in un numero non specificato di passi da x non è esprimibile in logica predicativa.

E il Tempo Polinomiale Deterministico?

- La logica predicativa è, come abbiamo visto, troppo debole per i nostri scopi.
- D'altro canto, quella al second'ordine risulta troppo potente.
- Cosa serve alla logica predicativa per catturare *esattamente* il tempo polinomiale deterministico?
- Consieriamo il vocabolario relativo ai grafi, ossia quello in cui l'unico simbolo relazionale E è binario e rappresenta l'adiacenza tra nodi.
- Il fatto che un nodo y sia *raggiungibile* in un numero non specificato di passi da x non è esprimibile in logica predicativa.
- Ci vorrebbe qualcosa come un nuovo predicato, cioè "il più piccolo" tra tutti quelli che soddisfano la seguente "equazione":

$$E^*(x, y) \equiv x = y \vee \exists z.(E(x, z) \wedge E^*(z, y))$$

Formule Positive e Loro Minimi Punti Fissi

- Consideriamo un formula predicativa al prim'ordine F in cui le variabili libere sono:
  - Una variabile al *second*'ordine X^m;
  - m variabili al prim'ordine $x_1, \ldots, x_m$.

Formule Positive e Loro Minimi Punti Fissi

- Consideriamo un formula predicativa al prim'ordine F in cui le variabili libere sono:
  - Una variabile al *second*'ordine X^m;
  - m variabili al prim'ordine $x_1, \ldots, x_m$.
- Tale formula predicativa si dice X^m-positiva se ogni occorrenza di X^m in F è nello scope di un numero *pari* di negazioni.

Formule Positive e Loro Minimi Punti Fissi

- Consideriamo un formula predicativa al prim'ordine F in cui le variabili libere sono:
  - Una variabile al *second*'ordine X^m;
  - m variabili al prim'ordine $x_1, \ldots, x_m$.
- Tale formula predicativa si dice X^m-positiva se ogni occorrenza di X^m in F è nello scope di un numero *pari* di negazioni.
- Data un'interpretazione I per $\mathcal{A}_n$, possiamo pensare a F come ad un funzionale F^I su $\mathcal{P}(\mathcal{A}_n^m)$, ossia il seguente:

$$D \longmapsto \{(a_1, \ldots, a_m) \in \mathcal{A}_n^m \mid (\mathcal{A}_n, I), \xi \models F, \\ \text{dove } \xi(X^m) = D \text{ e } \xi(x_i) = a_i\}$$

Lemma

Per ogni F che sia X^m-positiva, il funzionale F^I è monotono, e ammette quindi un minimo punto fisso $\mu^I X^m(x_1, \ldots, x_m).F$.

Logica Predicativa e Minimi Punti Fissi

- Le formule della *logica predicativa con minimi punti fissi* sono le stesse della logica predicativa al prim'ordine, ma tra i simboli predicativi ve ne sono anche nella forma $LFP(X^m, x_1, \ldots, x_m, F)$ dove F è una formula X^m-positiva.

Logica Predicativa e Minimi Punti Fissi

- Le formule della *logica predicativa con minimi punti fissi* sono le stesse della logica predicativa al prim'ordine, ma tra i simboli predicativi ve ne sono anche nella forma $LFP(X^m, x_1, \ldots, x_m, F)$ dove F è una formula X^m-positiva.
- Alle nuove formule si può dare semantica nel modo seguente:

$$\begin{aligned}(\mathcal{A}, I), \xi \models\ & LFP(X^m, x_1, \ldots, x_m, F)(t_1, \ldots, t_m) \\ & \text{sse} \qquad ([\![t_1]\!]_\xi^{(\mathcal{A},I)}, \ldots, [\![t_m]\!]_\xi^{(\mathcal{A},I)}) \in \mu^I X^m(x_1, \ldots, x_m).F.\end{aligned}$$

Logica Predicativa e Minimi Punti Fissi

- Le formule della *logica predicativa con minimi punti fissi* sono le stesse della logica predicativa al prim'ordine, ma tra i simboli predicativi ve ne sono anche nella forma $LFP(X^m, x_1, \ldots, x_m, F)$ dove F è una formula X^m-positiva.
- Alle nuove formule si può dare semantica nel modo seguente:

$$
\begin{aligned}
(\mathcal{A}, I),\xi \models {} & LFP(X^m, x_1, \ldots, x_m, F)(t_1, \ldots, t_m) \\
& \text{sse} \qquad (\llbracket t_1 \rrbracket_\xi^{(\mathcal{A},I)}, \ldots, \llbracket t_m \rrbracket_\xi^{(\mathcal{A},I)}) \in \mu^I X^m(x_1, \ldots, x_m).F.
\end{aligned}
$$

- A questo punto

$$
\begin{aligned}
FO(LFP) = \{\mathbf{struct}(F) \mid {} & F \text{ è una formula} \\
& \text{predicativa con minimi punti fissi}\}
\end{aligned}
$$

Logica Predicativa e Minimi Punti Fissi

- Le formule della *logica predicativa con minimi punti fissi* sono le stesse della logica predicativa al prim'ordine, ma tra i simboli predicativi ve ne sono anche nella forma $LFP(X^m, x_1, \ldots, x_m, F)$ dove F è una formula X^m-positiva.
- Alle nuove formule si può dare semantica nel modo seguente:

$$\begin{aligned}(\mathcal{A}, I), \xi \models\ & LFP(X^m, x_1, \ldots, x_m, F)(t_1, \ldots, t_m) \\ & \text{sse} \qquad ([\![t_1]\!]_\xi^{(\mathcal{A},I)}, \ldots, [\![t_m]\!]_\xi^{(\mathcal{A},I)}) \in \mu^I X^m(x_1, \ldots, x_m).F.\end{aligned}$$

- A questo punto

$$FO(LFP) = \{\mathbf{struct}(F) \mid F \text{ è una formula predicativa con minimi punti fissi}\}$$

Teorema (Immerman, Vardi)

$FO(LFP) = \mathsf{P}$

In Conclusione

Corollary

$\mathsf{P} = \mathsf{NP}$ *se e solo se* $FO(LFP) = \exists SO$.

In Conclusione

Corollary

$\mathsf{P} = \mathsf{NP}$ *se e solo se* $FO(LFP) = \exists SO$.

- In questo modo, un problema di complessità può essere ridotto ad un problema riguardante l'espressività di due logiche.

In Conclusione

Corollary

P = NP *se e solo se* $FO(LFP) = \exists SO$.

- In questo modo, un problema di complessità può essere ridotto ad un problema riguardante l'espressività di due logiche.
- Se si riuscisse a dimostrare che la quantificazione esistenziale al second'ordine **non** può essere espressa con i punti fissi. . .

In Conclusione

Corollary

$\mathsf{P} = \mathsf{NP}$ *se e solo se* $FO(LFP) = \exists SO$.

- In questo modo, un problema di complessità può essere ridotto ad un problema riguardante l'espressività di due logiche.
- Se si riuscisse a dimostrare che la quantificazione esistenziale al second'ordine **non** può essere espressa con i punti fissi…
- A tutt'oggi, non si è ancora riusciti a dimostrare alcunché, purtroppo.

Tutto Qui?

- Le interazioni fruttuose tra logica e complessità sono limitate alla complessità descrittiva?

Tutto Qui?

- Le interazioni fruttuose tra logica e complessità sono limitate alla complessità descrittiva?
  - Assolutamente **NO**! Vale in realtà il contrario.

Tutto Qui?

- Le interazioni fruttuose tra logica e complessità sono limitate alla complessità descrittiva?
  - Assolutamente **NO**! Vale in realtà il contrario.
- Alcuni altri modi in cui i due mondi possono interagire:
  - La cosiddetta **proof complexity**, in cui viene studiata l'esitenza di sistemi di prova che producano prove *compatte* per tutte le tautologie.

Tutto Qui?

- Le interazioni fruttuose tra logica e complessità sono limitate alla complessità descrittiva?
  - Assolutamente **NO**! Vale in realtà il contrario.
- Alcuni altri modi in cui i due mondi possono interagire:
  - La cosiddetta **proof complexity**, in cui viene studiata l'esitenza di sistemi di prova che producano prove *compatte* per tutte le tautologie.
  - La **implicit computational complexity**, che studia *caratterizzazioni* di classi di complessità dati in termine di linguaggi di programmazione e sistemi di prova.