

Complessità computazionale



La gerarchia polinomiale e PSPACE

Piero A. Bonatti

Università di Napoli Federico II

Laurea Magistrale in Informatica

Tema della lezione I

- La vera classe dei comuni problemi di ottimizzazione: **DP**
- Generalizzare **DP**
 - MdT con oracolo
 - la gerarchia polinomiale (**PH**)
- **PSPACE**

Classificazione dei problemi di ottimizzazione

- Riconsideriamo ad esempio TSP (problema di minimizzazione). In realtà ha 2 versioni decisionali
 - TSP(D): dato $B \in \mathbb{N}$, dire se esiste un tour di costo $\leq B$
 - EXACT TSP: dato $B \in \mathbb{N}$, dire se il tour ottimo ha costo $= B$
- Anche EXACT TSP è **NP-hard** (dimostrazione simile a TSP(D))
 - stessa riduzione da HAMILTON PATH: $B = n$ e

$$d_{ij} = \begin{cases} 1 & \text{se } [i, j] \text{ appartiene al grafo originale} \\ 2 & \text{altrimenti} \end{cases}$$

Classificazione dei problemi di ottimizzazione

- Ma $\text{EXACT TSP} \in \mathbf{NP}$? Che certificato succinto posso avere?
 - specialmente per mostrare che *nessun* altro tour ha costo inferiore a B
 - significa parlare di un numero esponenziale di altri tour
- Similmente, $\text{EXACT TSP} \in \mathbf{coNP}$? Che disqualifier succinto posso avere?
 - se il costo ottimo è $< B$ allora è facile: basta mostrare 1 tour di costo $< B$
 - se il costo ottimo è $> B$ allora è difficile quanto prima: serve una evidenza compatta che *nessun* tour ha costo $\leq B$
- Vedremo che se fosse $\text{EXACT TSP} \in \mathbf{NP} \cup \mathbf{coNP}$ allora le conseguenze sarebbero eccezionali!

Analisi di EXACT TSP

- EXACT TSP è l'intersezione di due linguaggi
 - L_1 : grafi il cui tour ottimo ha costo $\leq B$ ($\overline{\text{TSP(D)}}$)
 - L_2 : grafi il cui tour ottimo ha costo $\geq B$ ($\overline{\text{TSP(D)}}$)
- tali che $L_1 \in \mathbf{NP}$ e $L_2 \in \mathbf{coNP}$
- Quindi decidere se $x \in \text{EXACT TSP}$ equivale a risolvere
 - sia un problema **NP**-completo ($x \in L_1?$)
 - sia un problema **coNP**-completo ($x \in L_2?$)
- Ci fa pensare che EXACT TSP sia *più difficile* di entrambi i problemi
 - lo stesso vale per le versioni EXACT degli altri problemi di ottimizzazione **NP**-completi (INDEPENDENT SET, KNAPSACK, NODE COVER ecc.)

La classe DP

Definizione di DP

L appartiene a **DP** se e solo se esistono $L_1 \in \mathbf{NP}$ e $L_2 \in \mathbf{coNP}$ tali che $L = L_1 \cap L_2$

- **ATTENZIONE:** non confondere **DP** con $\mathbf{NP} \cap \mathbf{coNP}$
 - $L \in \mathbf{DP}$ significa che le *istanze* “yes” di L sono in $L_1 \cap L_2$
 - $L \in \mathbf{NP} \cap \mathbf{coNP}$ significa che L stesso appartiene a $\mathbf{NP} \cap \mathbf{coNP}$
 - nel primo caso devo risolvere 2 sottoproblemi ($x \in L_1, x \in L_2$)
 - il primo richiede un certificato succinto
 - il secondo l'assenza di disqualifier succinti
 - nel secondo basta risolvere 1 problema (trovando o un certificato, se l'istanza è “yes”, o un disqualifier, se è “no”)
- In realtà è plausibile che esistano $L \in \mathbf{DP}$ tali che $L \notin \mathbf{NP} \cup \mathbf{coNP}$

La classe DP

- Chiaramente, le versioni esatte dei problemi di ottimizzazione
 - EXACT INDEPENDENT SET
 - EXACT KNAPSACK
 - EXACT MAX2SAT
 - EXACT ...
- appartengono tutte a **DP**
- Quali sono i problemi **DP**-completi?

SAT-UNSAT è DP-completo

Definizione

Date due espressioni booleane ϕ e ϕ'
dire se ϕ è soddisfacibile e ϕ' non lo è

- Quindi SAT-UNSAT combina due classici problemi completi per **NP** e **coNP** (SAT e $\overline{\text{SAT}}$)

SAT-UNSAT è DP-completo

Appartenenza a DP

- Per mostrare che $\text{SAT-UNSAT} \in \mathbf{DP}$ bisogna riformularlo come intersezione di due linguaggi in \mathbf{NP} e \mathbf{coNP}
- Semplice:
 - $L_1 = \{(\phi, \phi') \mid \phi \text{ è soddisfacibile}\}$
 - $L_2 = \{(\phi, \phi') \mid \phi' \text{ è insoddisfacibile}\}$
- L_1 è chiaramente equivalente a SAT (ϕ' viene ignorata)
- L_2 è chiaramente equivalente a $\overline{\text{SAT}}$ (ϕ viene ignorata)
- Ovviamente $\text{SAT-UNSAT} = L_1 \cap L_2$

SAT-UNSAT è DP-completo

DP-hardness

- Sia dato un qualunque $L \in \mathbf{DP}$
 - quindi $L = L_1 \cap L_2$ per qualche $L_1 \in \mathbf{NP}$ e $L_2 \in \mathbf{coNP}$
dobbiamo ridurre L a SAT-UNSAT
- Esistono due riduzioni $R_1 : L_1 \rightarrow \text{SAT}$ e $R_2 : L_2 \rightarrow \overline{\text{SAT}}$
 - (perchè SAT è **NP**-completo e $\overline{\text{SAT}}$ **coNP**-completo)Porre $R(x) = (R_1(x), R_2(x))$ – chiaramente è in \mathbf{L}
- Per le proprietà di R_1, R_2 , abbiamo la correttezza:

$$R(x) \in \text{SAT-UNSAT} \Leftrightarrow R_1(x) \in \text{SAT} \wedge R_2(x) \in \overline{\text{SAT}}$$

$$\Leftrightarrow x \in L_1 \wedge x \in L_2 \Leftrightarrow x \in L$$

QED

Altri problemi DP-completi

- EXACT TSP
 - Abbiamo già mostrato l'appartenenza, la hardness si può ottenere con una riduzione da SAT-UNSAT
- CRITICAL SAT
 - Data ϕ in CNF, dire se è insoddisfacibile ma basta togliere una qualunque clausola per renderla soddisfacibile
- CRITICAL HAMILTON PATH
 - Il grafo dato non ha cicli hamiltoniani, ma aggiungendo qualunque arco se ne crea uno
- CRITICAL 3-COLORING
 - Il grafo dato non può essere colorato con 3 colori ma togliendo un qualunque nodo lo diventa

Un problema in DP

- UNIQUE SAT
 - L'espressione ϕ data, è soddisfatta da esattamente un truth assignment?
- Appartiene a DP
- Non è noto se sia DP-completo, nè se appartenga a una classe più piccola

Generalizzando DP

- **DP** è la classe di problemi cui si può rispondere risolvendo due sottoproblemi “equivalenti a SAT”
 - uno con risposta “yes”, uno con risposta “no”
- Che dire della classe di problemi che si risolvono
 - con un qualunque numero *polinomiale* di sottoproblemi **NP**-completi
 - e con arbitrarie combinazioni di risposte “yes” e “no” ?
- Come li modelliamo?
 - astraendo la soluzione del sottoproblema
 - facendola apparire come una operazione atomica a costo unitario
 - mostrando solo l’algoritmo che sceglie i sottoproblemi e “combina” tra loro i risultati

MdT con oracolo

Definizione di MdT con oracolo

È una MdT multinastro $M^?$

- con un nastro particolare, detto di *query*
 - con 3 stati speciali, $q_?$, q_{YES} , q_{NO}
-
- $M^?$ è come un *template*:
 - Dato un problema L , otteniamo M^L
 - dove i 3 stati speciali dicono in 1 passo se la stringa sul nastro di query appartiene a L o no
 - $q_?$ = “chiamata dell’oracolo per L ”
 - q_{YES} , q_{NO} = possibili risposte dell’oracolo
 - non determinate dal programma! l’oracolo è una “componente esterna”

MdT con oracolo

Definizione formale di computazione

- Alle transizioni delle MdT multinastro già viste aggiungere le seguenti:

$$(q?, w_1, u_1, \dots, w_k, u_k) \xrightarrow{M} (q_{ans}, w_1, u_1, \dots, w_k, u_k)$$

dove, se il nastro di query è l' i -esimo e $w_i u_i = \triangleright x$

- $q_{ans} = q_{YES}$ sse $x \in L$
- $q_{ans} = q_{NO}$ sse $x \notin L$
- Nota: le risorse necessarie per risolvere L sono completamente mascherate
 - la risposta arriva sempre in 1 passo
 - lo spazio richiesto è solo quello per scrivere la query x

Misure di tempo e spazio per MdT con oracolo

- Esattamente come per le MdT senza oracolo:
 - Tempo: numero di transizioni da stato iniziale a finale
 - Spazio: somma delle lunghezze dei nastri¹
- Così il costo della soluzione di L viene completamente scorporato
 - a parte la scrittura della query

¹È discutibile se considerare anche il nastro di query, vedere nota 14.5.8 del Papadimitriou

Esempio di MdT con oracolo per SAT-UNSAT

Fare per esercizio

- Oracolo: SAT (**NP**-completo)
- 2 nastri (per input e query)
- Dato l'input $\phi; \phi'$
 - 1 copia ϕ sul nastro di query e va in $q?$
 - 2 se la risposta è q_{NO} termina con "no"
 - 3 altrimenti copia ϕ' sul nastro di query e va in $q?$
 - 4 se la risposta è q_{YES} termina con "no"
 - 5 altrimenti termina con "yes"
- Problema: le transizioni in 2-3 e 4-5 sono diverse, a parità di q_{ans}
- Soluzione: salvare su di un nastro ausiliario la fase in cui ci si trova: inizialmente 1, poi al passo 3 sovrascrivere 2
 - le istruzioni dei passi 2 e 4 leggono anche la fase per decidere cosa fare

Esempio di MdT per SAT-UNSAT

Fare per esercizio

- Misura delle risorse:
- Tempo: $O(n)$ (copia di input su query)
- Spazio: $O(n)$ (nastro di query)
- Il costo della soluzione dei due sottoproblemi risulta incorporato

Classi con oracolo

Definizione di \mathcal{C}^L

Se \mathcal{C} è la classe di problemi che si possono riconoscere in un certo tempo con MdT di un certo tipo (deterministiche o no)

\mathcal{C}^L è la classe di problemi che si possono riconoscere con gli stessi tipi di MdT e limiti di tempo *usando un oracolo per L*

- Esempi: \mathbf{P}^{SAT} , $\mathbf{NP}^{\text{TSP(D)}}$, $\mathbf{EXP}^{\text{SAT}}$, ...

Classi con oracolo

Non conta L ma la sua classe di complessità. Per esempio:

Proposizione

Per qualunque L **NP**-completo, $\mathbf{P}^L = \mathbf{P}^{\text{SAT}}$

- Prova: data una MdT M^L se ne ottiene una equivalente M'^{SAT} traducendo le query con la riduzione $L \rightarrow \text{SAT}$
 - costo aggiuntivo: polinomiale, sia nel numero di traduzioni che nel costo di ciascuna (resta in $\mathbf{P}^?$)

Similmente per ogni M'^{SAT} se ne ottiene una equivalente M^L

QED

Classi con oracolo

Non conta L ma la sua classe di complessità

- La stessa prova funziona per tutte le classi \mathcal{C}^L dove
 - le risorse richieste da \mathcal{C} dominano quelle per le riduzioni (spazio logaritmico, tempo polinomiale)
 - gli oracoli sono riducibili l'uno all'altro (hanno la stessa complessità)
- Quindi ogni specifico oracolo L si può sostituire direttamente con la sua classe di complessità:
 - $\mathbf{P}^{\mathbf{NP}}$, $\mathbf{NP}^{\mathbf{NP}}$, $\mathbf{EXP}^{\mathbf{NP}}$, invece di $\mathbf{P}^{\mathbf{SAT}}$, $\mathbf{NP}^{\mathbf{TSP(D)}}$, $\mathbf{EXP}^{\mathbf{CLIQUE}}$, ...

Classi con oracolo

I complementari non contano

Proposizione

$$\mathcal{C}^{\mathcal{D}} = \mathcal{C}^{\text{co}\mathcal{D}}$$

- Prova: ogni MdT $M^{\mathcal{D}}$ si trasforma in una $M'^{\text{co}\mathcal{D}}$ equivalente (e viceversa) invertendo q_{YES} e q_{NO} nella relazione di transizione

QED

- È un effetto del vedere l'oracolo come una "black box" deterministica
- Ecco perchè non si usano \mathbf{P}^{coNP} , $\mathbf{NP}^{\text{coNP}}$, $\mathbf{EXP}^{\text{coNP}}$, ...

Prima generalizzazione di DP

- P^{NP} è proprio la generalizzazione di **DP** di cui parlavamo
 - Classe dei problemi risolvibili con una combinazione polinomiale di chiamate a problemi in **NP** e **coNP**
 - il numero di chiamate all'oracolo è polinomiale
 - la costruzione delle query è polinomiale
 - la combinazione dei risultati è polinomiale
- Chiaramente $DP \subseteq P^{NP}$

Oracoli inutili (?)

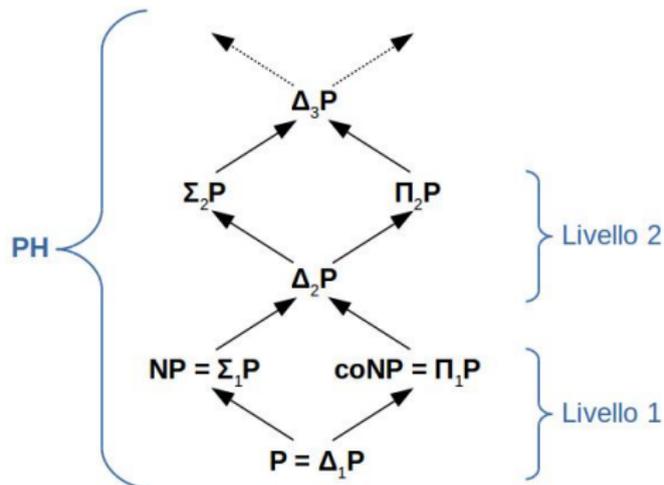
- **$P = P^P$?**
 - Sì! Ogni chiamata all'oracolo può essere espansa *in line*
 - numero di chiamate: polinomiale
 - costo di ogni chiamata: polinomiale
 - quindi: aumento di tempo polinomiale (restiamo in **P**)
- **$NP = NP^P$?**
 - Sì! Per la stessa ragione
- **$NP = NP^{NP}$?**
 - ah, saperlo...

La gerarchia polinomiale (Polynomial Hierarchy, **PH**)

La gerarchia polinomiale (PH)

Definizione: $\Delta_i\mathbf{P}$, $\Sigma_i\mathbf{P}$, $\Pi_i\mathbf{P}$ e PH

- $\Delta_0\mathbf{P} = \Sigma_0\mathbf{P} = \Pi_0\mathbf{P} = \mathbf{P}$
- $\Delta_{i+1}\mathbf{P} = \mathbf{P}^{\Sigma_i\mathbf{P}}$
- $\Sigma_{i+1}\mathbf{P} = \mathbf{NP}^{\Sigma_i\mathbf{P}}$
- $\Pi_{i+1}\mathbf{P} = \mathbf{coNP}^{\Sigma_i\mathbf{P}}$
- $\mathbf{PH} = \bigcup_{i \geq 0} \Sigma_i\mathbf{P}$



- Qualche testo usa Δ_i^P , Σ_i^P , Π_i^P

Struttura dei problemi in PH

Teorema 17.8

Per ogni $i \geq 1$, $L \in \Sigma_i \mathbf{P}$ sse esiste una R polynomially balanced tale che

- $\{x; y \mid (x, y) \in R\}$ appartiene a $\Pi_{i-1} \mathbf{P}$ e
- $L = \{x \mid \text{esiste } y \text{ tale che } (x, y) \in R\}$

- **Prova:** per induzione su i
- **Caso base:** Notare che per $i = 1$ il teorema si riduce alle relazioni polynomially balanced e polynomially decidable che caratterizzano **NP**

(segue)

Struttura dei problemi in PH

Struttura dei $L \in \Sigma_i \mathbf{P}$

- **Passo induttivo (se):** Prima assumiamo che R esista e mostriamo che $L \in \Sigma_i \mathbf{P}$
 - producendo una MdT nondeterministica che decide L in tempo polinomiale con oracolo in $\Sigma_{i-1} \mathbf{P}$
- Basta prendere una MdT il cui oracolo è il complemento di $\{x; y \mid (x, y) \in R\}$ (che appartiene a $\Sigma_{i-1} \mathbf{P}$) che
 - 1 genera nondeterministicamente y in tempo polinomiale (perchè?)
 - 2 verifica se $(x, y) \in R$ chiamando l'oracolo per $\Sigma_{i-1} \mathbf{P}$ e invertendo la risposta

(segue)

Struttura dei problemi in PH

- **Passo induttivo (solo se):** Ora assumiamo che $L \in \Sigma_i \mathbf{P}$ e mostriamo che R esiste
- Sia $M^?$ la MdT nondeterministica che riconosce L in tempo polinomiale usando un oracolo $K \in \Sigma_{i-1} \mathbf{P}$
 - per ipotesi di induzione, esiste una relazione S polynomially balanced che caratterizza K , decidibile in $\Pi_{i-2} \mathbf{P}$
- Nelle $(x, y) \in R$, y rappresenta una computazione di $M^K(x)$ mediante sequenze di interi $\leq d$ che rappresentano le scelte nondeterministiche...
- ...e mediante i certificati w_i delle chiamate z_i all'oracolo, con risposta "yes", ad es.

$$y = 12(z_1, w_1)231(z_2, w_2)571$$

dove $(z_i, w_i) \in S$ e $|w_i| \leq |z_i|^k$

(segue)

Struttura dei problemi in PH

Struttura dei $L \in \Sigma_i \mathbf{P}$

- (segue) Resta da dimostrare che $\{x; y \mid (x, y) \in R\}$ è in $\Pi_{i-1} \mathbf{P}$
 - verificando se la computazione di $M^K(x)$ descritta da y è di accettazione
- Simulare $M^K(x)$ usando un oracolo Σ_{i-2} -completo e:
 - gli interi di y per operare le scelte nondeterministiche
 - l'oracolo per verificare che non sia $(z_i, w_i) \notin S$
 - fin qui, computazioni polinomiali con oracolo in Σ_{i-2} , quindi ricadono in $\Pi_{i-1} \mathbf{P}$
 - per le query "no", decidere se $z_i \notin K$ è in $\Pi_{i-1} \mathbf{P}$, quindi si può programmare la MdT per verificarlo rimanendo in $\Pi_{i-1} \mathbf{P}$

QED

Struttura dei problemi in PH

Corollario

Per ogni $i \geq 1$, $L \in \Pi_i \mathbf{P}$ sse esiste una R polynomially balanced (con esponente k) tale che

- $\{x; y \mid (x, y) \in R\}$ è in $\Sigma_{i-1} \mathbf{P}$
 - $L = \{x \mid \text{per ogni } y \text{ t.c. } |y| \leq |x|^k, (x, y) \in R\}$
-
- Si dimostra sfruttando $\Pi_i \mathbf{P} = \mathbf{co}\Sigma_i \mathbf{P}$
 - R è il complemento della relazione polynomially balanced \bar{R} che esiste per $\bar{L} \in \Sigma_i \mathbf{P}$

Struttura dei problemi in PH

Rimuovere la ricorsione

Definizione

Una relazione $R \subseteq (\Sigma^*)^{i+1}$ è *polynomially balanced* sse per ogni $(x, y_1, \dots, y_i) \in R$, $|y_j| \leq |x|^k$ ($j = 1, \dots, i$)

Corollario

Per ogni $i \leq 1$, $L \in \Sigma_i \mathbf{P}$ sse esiste una relazione polynomially balanced R calcolabile in tempo polinomiale tale che

$$L = \{x \mid \exists y_1 \forall y_2 \exists y_3 \dots Q y_i \text{ tale che } (x, y_1, \dots, y_i) \in R\}$$

dove Q è \exists se i è dispari e \forall se i è pari

- Prova: applicare ricorsivamente i risultati precedenti per sostituire i $\Sigma_i \mathbf{P}$ e $\Pi_i \mathbf{P}$ con i loro certificati i -ari

Struttura di PH

- Non è noto se i livelli di **PH** siano distinti
- nè se $\Sigma_i\mathbf{P} \neq \Pi_i\mathbf{P}$
 - si pensa di sì
- Come sempre si può ragionare su cosa succederebbe se fossero uguali

Struttura di PH

Teorema 17.9

Se $\Sigma_i \mathbf{P} = \Pi_i \mathbf{P}$ allora per ogni $j > i$,

$$\Sigma_j \mathbf{P} = \Pi_j \mathbf{P} = \Delta_j \mathbf{P} = \Sigma_i \mathbf{P}$$

- **Prova:** Basta mostrare che $\Sigma_{i+1} \mathbf{P} = \Sigma_i \mathbf{P}$
 - segue che $\Pi_{i+1} \mathbf{P} = \Pi_i \mathbf{P} = \Sigma_i \mathbf{P} = \Sigma_{i+1} \mathbf{P}$
 - poi si riapplica l'uguaglianza ai livelli successivi

(segue)

Struttura di PH

Collasso di PH a livello i

- (segue): per dimostrare $\Sigma_{i+1}\mathbf{P} = \Sigma_i\mathbf{P}$, prendiamo un $L \in \Sigma_{i+1}\mathbf{P}$
- Per il Teorema 17.8 esiste una R polynomially balanced in $\Pi_i\mathbf{P}^2$ t.c.

$$L = \{x \mid \text{esiste } y \text{ t.c. } (x, y) \in R\}$$

- Poichè $\Sigma_i\mathbf{P} = \Pi_i\mathbf{P}$, R è anche in $\Sigma_i\mathbf{P}$. Quindi per 17.8 esiste S in $\Pi_{i-1}\mathbf{P}$ tale che

$$(x, y) \in R \Leftrightarrow (x; y, z) \in S$$

Chiaramente anche $S' = \{(x, y; z) \mid (x; y, z) \in S\}$ è in $\Pi_{i-1}\mathbf{P}$

- Inoltre $|y| \leq |x|^{k_1}$ e $|z| \leq |x; y|^{k_2}$, quindi $|y; z| = O(|x|^{k_1 k_2})$
- Quindi S' (polynomially balanced e in $\Pi_{i-1}\mathbf{P}$) prova che $L \in \Sigma_i\mathbf{P}$

QED

²Più precisamente, il linguaggio $\{x; y \mid (x, y) \in R\}$ è in $\Pi_i\mathbf{P}$

Terminologia e corollari

- Quando per ogni $j > i$ abbiamo $\Sigma_j\mathbf{P} = \Pi_j\mathbf{P} = \Delta_j\mathbf{P} = \Sigma_i\mathbf{P}$ diciamo
PH collassa all' i -esimo livello

Corollari

Se $\mathbf{P} = \mathbf{NP}$ o se $\mathbf{NP} = \mathbf{coNP}$
allora **PH** collassa al suo primo livello

Problemi completi per **PH** e i suoi livelli

STRATEGIC COMPANIES è $\Sigma_2\mathbf{P}$ -completo

Definizione di STRATEGIC COMPANIES

Dati una collezione di società $C = \{c_1, \dots, c_m\}$ possedute da una holding, un insieme di prodotti (*goods*) $G = \{g_1, \dots, g_n\}$ e per ogni società $c \in C$

- un insieme $G(c) \subseteq G$ di merci prodotte da c
- k insiemi $O_1(c) \dots O_k(c) \subseteq C$ di società che controllano (*own*) c

Dire se una società c data è *strategica*, cioè appartiene a un insieme minimo $C' \subseteq C$ tale che:

- $\bigcup_{c \in C'} G(c) = G$ (C' produce tutte le merci della holding)
- se qualche $O_i(c) \subseteq C'$ allora $c \in C'$ (*)

- Le compagnie non strategiche possono essere vendute
- (*) assicura che questo non richieda di vendere anche qualche società essenziale per mantenere la produzione

Commonsense reasoning e Nonmonotonic logics

- Le logiche nonmonotone furono introdotte in AI per permettere *in pratica* il ragionamento in condizioni di conoscenza incompleta
 - facendo assunzioni plausibili che potrebbero rivelarsi false in un secondo momento
- Esempi di logiche nonmonotone comprendono
 - Circumscription
 - Default logic
 - Answer Set Programming (ASP)
- Il ragionamento nelle versioni proposizionali di queste logiche si colloca al 2° livello della **PH**

Motivazioni: The Yale Shooting Problem

Reasoning about actions & change

Descrizione dello scenario

Se si spara a un tacchino con una pistola carica, il tacchino muore.

Si vuole derivare che:

Dopo aver caricato la pistola e dopo aver sparato (nell'ordine), il tacchino è morto.

- Azioni possibili al tempo t : $load_t$, $unload_t$, $shoot_t$

Motivazioni: The Yale Shooting Problem

Versione 1

- Modellazione in logica proposizionale – primo tentativo

- $load_t \rightarrow loaded_{t+1}$
- $unload_t \rightarrow \neg loaded_{t+1}$
- $loaded_t \wedge shoot_t \rightarrow dead_{t+1}$
- $load_0$
- $shoot_1$

- Possiamo concludere $dead_2$?

- Sì, è una *conseguenza logica* degli assiomi qui sopra

$$\begin{array}{c}
 load_0 \quad load_0 \rightarrow loaded_1 \\
 \hline
 loaded_1 \quad shoot_1 \quad loaded_1 \wedge shoot_1 \rightarrow dead_2 \\
 \hline
 dead_2
 \end{array}$$

Motivazioni: The Yale Shooting Problem

Versione 2

- Modellazione in logica proposizionale – primo tentativo, secondo caso
 - $load_t \rightarrow loaded_{t+1}$
 - $unload_t \rightarrow \neg loaded_{t+1}$
 - $loaded_t \wedge shoot_t \rightarrow dead_{t+1}$
 - $load_0$
 - $shoot_2$
- Possiamo concludere $dead_3$?
 - NO: nulla implica $loaded_1 \rightarrow loaded_2$
 - così non possiamo applicare $loaded_2 \wedge shoot_2 \rightarrow dead_3$
 - Informalmente: nulla dice che l'arma non venga scaricata al tempo 1

Motivazioni: The Yale Shooting Problem

Versione 3

- Modellazione in logica proposizionale – correzione sbagliata
 - $load_t \rightarrow loaded_{t+1}$
 - $unload_t \rightarrow \neg loaded_{t+1}$
 - $loaded_t \wedge shoot_t \rightarrow dead_{t+1}$
 - $loaded_t \rightarrow loaded_{t+1}$ (la pistola resta carica)
 - $load_0$
 - $unload_1$
 - $shoot_2$
- Possiamo concludere $dead_3$?
 - Sì, ma la modellazione è sbagliata:
 - se la pistola viene scaricata al tempo 1 otteniamo una contraddizione: $loaded_2 \wedge \neg loaded_2$

Motivazioni: The Yale Shooting Problem

Versione 4

- Modellazione in logica proposizionale – correzione della correzione
 - $load_t \rightarrow loaded_{t+1}$
 - $unload_t \rightarrow \neg loaded_{t+1}$
 - $loaded_t \wedge shoot_t \rightarrow dead_{t+1}$
 - $loaded_t \wedge \neg unload_t \rightarrow loaded_{t+1}$
 - $load_0$
 - $shoot_2$
- Possiamo concludere $dead_3$?
 - NO: nulla dice che $\neg unload_1$
 - in logica (diversamente dai database) ciò che manca non è falso, ma sconosciuto
 - ...e siamo tornati al problema iniziale

Motivazioni: The Yale Shooting Problem

Dove sta il problema

- Non possiamo assumere che le cose *restino come sono in assenza di indicazioni contrarie*
- Possiamo solo inferire conclusioni condizionali del tipo *se al tempo 1 non succede questo e al tempo 2 non succede quest'altro e ... allora al tempo t abbiamo...*
- Bisognerebbe citare esplicitamente tutte le cose previste e impreviste che possono succedere nell'intervallo di tempo $[0, t]$
 - ingestibile in pratica
- Il commonsense reasoning si propone di supportare una rappresentazione più gestibile in pratica mediante **logiche nonmonotone**

Lo Yale Shooting Problem è nonmonotono

- Sia KB l'insieme degli assiomi dello YSP 2^a versione (con $load_0$ e $shoot_2$)
- Vogliamo concludere $dead_3$ da KB (assumendo che l'arma non venga scaricata se non viene detto esplicitamente)...
- Ma **non da** $KB \cup \{unload_1\}$
- In questo senso la logica non è monotona: l'insieme delle conseguenze non aumenta al crescere degli assiomi
- Limite della logica classica: è **monotona**
 - ogni dimostrazione a partire da *qualunque* KB è valida per *qualunque* $KB' \supseteq KB$
 - ovvero $KB \subseteq KB'$ e $KB \models \phi$ implicano $KB' \models \phi$
 - è vincolata per definizione alle sole inferenze *valide*, che non possono essere invalidate da nessuna nuova informazione

Circumscription

- Specifica condizioni di anomalità
 - con *abnormality predicates*
- Nel ragionamento considera solo truth assignment che sono *massimamente normali*
 - questo rende il ragionamento nonmonotono
- Così combina SAT con un problema di massimizzazione
 - e questo fa salire la complessità del ragionamento dal primo al secondo livello

Circumscription

Preferenze sui modelli (normalità)

- Sia M l'insieme di proposizioni che rappresentano situazioni anormali
 - da rendere “più false possibile”
- un truth assignment T è preferito a (più normale di) T' sse
 - per ogni simbolo proposizionale $p \in M$
 - $T'(p) = \text{false} \Rightarrow T(p) = \text{false}$
 - (T è almeno tanto normale quanto T')

In questo caso scriviamo $T \preceq_M T'$

- i *modelli* di un insieme di assiomi KB (in Circumscription) sono
 - i truth assignment T che soddisfano KB ($T \models KB$) tali che
 - per ogni T' che soddisfa KB , $T' \preceq_M T \Rightarrow T \preceq_M T'$
- Se ϕ è vera in tutti i modelli di KB , scriviamo $KB \models_M \phi$
 - e diciamo che ϕ è una conseguenza di KB

Lo YSP in Circumscription

- $M = \{ab_t \mid t \in \mathbb{N}\}$. Assiomi (KB):

- $load_t \rightarrow loaded_{t+1}$ (1)

- $unload_t \rightarrow \neg loaded_{t+1}$ (2)

- $loaded_t \wedge shoot_t \rightarrow dead_{t+1}$ (3)

- $loaded_t \wedge \neg ab_t \rightarrow loaded_{t+1}$

- $load_0$

- $shoot_2$

- I modelli qui sono completamente normali: tutti gli ab_t sono falsi

- possiamo derivare $dead_3$ ($KB \models_M dead_3$)

$$load_0 \quad (1)$$

$$loaded_1 \quad \text{assume } \neg ab_1 \quad loaded_1 \wedge \neg ab_1 \rightarrow loaded_2$$

$$loaded_2$$

$$shoot_2 \quad (3)$$

$$dead_3$$

Lo YSP in Circumscription – scenario 2

- $M = \{ab_t \mid t \in \mathbb{N}\}$. Assiomi (KB):

- $load_t \rightarrow loaded_{t+1}$ (1)

- $unload_t \rightarrow \neg loaded_{t+1}$ (2)

- $loaded_t \wedge shoot_t \rightarrow dead_{t+1}$ (3)

- $loaded_t \wedge \neg ab_t \rightarrow loaded_{t+1}$

- $load_0, unload_1, shoot_2$

- Qui $unload_1$ rende ab_1 vero in tutti i modelli di KB (per evitare la contraddizione $loaded_2 \wedge \neg loaded_2$)

- quindi – giustamente – non deriviamo $dead_3$

$$load_0 \quad (1)$$

$$loaded_1 \quad \text{assume } \neg ab_1 \quad loaded_1 \wedge \neg ab_1 \rightarrow loaded_2 \quad unload_1 \quad (2)$$

$$loaded_2$$

$$\neg loaded_2$$

$$loaded_2 \wedge \neg loaded_2$$

Lo YSP in Circumscription – Diagnosi

- La logica è molto flessibile, accetta imprevisti di ogni tipo:
 - $load_t \rightarrow loaded_{t+1}$ (1)
 - $unload_t \rightarrow \neg loaded_{t+1}$ (2)
 - $loaded_t \wedge shoot_t \rightarrow dead_{t+1}$ (3)
 - $loaded_t \wedge \neg ab_t \rightarrow loaded_{t+1}$
 - $load_0, shoot_2$ e $\neg dead_3$ (osservazione)
- Qui $\neg dead_3$ rende ab_1 vero in tutti i modelli di KB (per evitare la contraddizione $dead_3 \wedge \neg dead_3$)
 - possiamo derivare la diagnosi $KB \models_M ab_1$ (qualcosa ha scaricato la pistola al tempo $t = 1$)

$$\frac{}{load_0} \quad (1)$$

$$\frac{loaded_1 \quad \text{assume } \neg ab_1 \quad loaded_1 \wedge \neg ab_1 \rightarrow loaded_2 \quad unload_1}{loaded_2 \quad \neg loaded_2} \quad (2)$$

$$loaded_2$$

$$\neg loaded_2$$

$$loaded_2 \wedge \neg loaded_2$$

Complessità intrinseca della Circumscription

Teorema (CIRC SAT e CIRC VALIDITY)

Decidere se un modello di KB soddisfa ϕ è $\Sigma_2\mathbf{P}$ -completo.

Decidere se $KB \models_M \phi$ è $\Pi_2\mathbf{P}$ -completo.

- Notare che $\text{CIRC SAT} \approx \text{SAT} + \text{“minimizzazione degli abnormality predicates”}$
- Come spesso accade, una ottimizzazione aumenta la complessità di un livello

Esercizio

- Dati $M = \{ab\}$ e la KB

- $bird \wedge \neg ab \rightarrow fly$

(birds normally fly)

- $penguin \rightarrow bird$

(penguins are birds)

- $penguin \rightarrow \neg fly$

(penguins can't fly)

- dire se

- 1 $KB \cup \{bird\} \models fly$

- 2 $KB \cup \{penguin\} \models \neg fly$

- 3 $KB \cup \{bird\} \models_M fly$

- 4 $KB \cup \{penguin\} \models_M \neg fly$

giustificando la risposta

Quantified Boolean Formulae (QBF)

Una fonte di problemi completi per tutta la PH

- Le QBF sono il frammento proposizionale della *logica del 2° ordine*
 - dove si può quantificare sui *predicati*
- Esempi di QBF vere:
 - $\forall p \exists q. \neg p \vee q$: per ogni valore di p (vero o falso) esiste un valore di q (vero o falso) tale che $\neg p \vee q$ è vera
 - $\forall p \forall q. \neg p \vee p \vee q$: ovvero $\neg p \vee p \vee q$ è valida
 - $\exists p \exists q. p \wedge q$: ovvero $p \wedge q$ è soddisfacibile
- Esempi di QBF false:
 - $\forall p \exists q. \neg p \vee (q \wedge \neg q)$: se p è vera, nessun valore di q può soddisfare $\neg p \vee (q \wedge \neg q)$
 - $\forall p \forall q. p \vee q$: $p \vee q$ non è valida
 - $\exists p \exists q. p \wedge \neg p \wedge q$: ovvero $p \wedge \neg p \wedge q$ non è soddisfacibile

Quantified Boolean Formulae (QBF)

QBF vs. Circumscription

- Oltre a SAT e VALIDITY (1° livello di **PH**), le QBF possono codificare anche la Circumscription (2° livello)
- Data una formula $\phi(\vec{p}, \vec{ab})$ con predicati $p_1, \dots, p_n, ab_1, \dots, ab_n$ esprimere che $\psi(\vec{p}, \vec{ab})$ è soddisfatta da un modello di $\phi(\vec{p}, \vec{ab})$
 - $\exists \vec{p}, \vec{ab}. \phi(\vec{p}, \vec{ab}) \wedge \psi(\vec{p}, \vec{ab}) \wedge$ “non esiste un truth assignment (\vec{p}', \vec{ab}') più normale che soddisfa ϕ ”
 - $\exists \vec{p}, \vec{ab}. \phi(\vec{p}, \vec{ab}) \wedge \psi(\vec{p}, \vec{ab}) \wedge \forall \vec{p}', \vec{ab}'. [\phi(\vec{p}', \vec{ab}') \rightarrow \neg \bigwedge_{i=1}^n (ab'_i \rightarrow ab_i) \vee \bigwedge_{i=1}^n (ab_i \rightarrow ab'_i)]$
 - $\exists \vec{p}, \vec{ab} \forall \vec{p}', \vec{ab}'. \phi(\vec{p}, \vec{ab}) \wedge \psi(\vec{p}, \vec{ab}) \wedge [\phi(\vec{p}', \vec{ab}') \rightarrow \neg \bigwedge_{i=1}^n (ab'_i \rightarrow ab_i) \vee \bigwedge_{i=1}^n (ab_i \rightarrow ab'_i)]$
- Questa QBF è vera sse ψ è soddisfatta in un modello di ϕ

Quantified Boolean Formulae (QBF)

QBF vs. Circumscription

- Data una formula $\phi(\vec{p}, \vec{ab})$ con predicati $p_1, \dots, p_n, ab_1, \dots, ab_n$ possiamo codificare $\phi(\vec{p}, \vec{ab}) \models_M \psi(\vec{p}, \vec{ab})$
 - $\forall \vec{p}, \vec{ab}$, “se $\phi(\vec{p}, \vec{ab})$ è vera e $\psi(\vec{p}, \vec{ab})$ falsa allora il truth assignment rappresentato dai valori di (\vec{p}, \vec{ab}) non è un modello (non è \leq_M -minimo)”
 - $\forall \vec{p}, \vec{ab}$. $[\phi(\vec{p}, \vec{ab}) \wedge \neg\psi(\vec{p}, \vec{ab})] \rightarrow$ “esiste un truth assignment (\vec{p}', \vec{ab}') più normale”
 - $\forall \vec{p}, \vec{ab}$. $[\phi(\vec{p}, \vec{ab}) \wedge \neg\psi(\vec{p}, \vec{ab})] \rightarrow$
 $\exists \vec{p}', \vec{ab}'$. $\phi(\vec{p}', \vec{ab}') \wedge \bigwedge_{i=1}^n (ab'_i \rightarrow ab_i) \wedge \neg \bigwedge_{i=1}^n (ab_i \rightarrow ab'_i)$
 - $\forall \vec{p}, \vec{ab} \exists \vec{p}', \vec{ab}'$. $[\phi(\vec{p}, \vec{ab}) \wedge \neg\psi(\vec{p}, \vec{ab})] \rightarrow$
 $\phi(\vec{p}', \vec{ab}') \wedge \bigwedge_{i=1}^n (ab'_i \rightarrow ab_i) \wedge \neg \bigwedge_{i=1}^n (ab_i \rightarrow ab'_i)$

Quantified Boolean Formulae (QBF)

QBF vs. Circumscription

- Conseguenze:
 - Le QBF della forma $\exists \vec{x} \forall \vec{y}. \phi$ sono hard per $\Sigma_2\mathbf{P}$ (2° livello di **PH**)
 - Le QBF della forma $\forall \vec{x} \exists \vec{y}. \phi$ sono hard per $\Pi_2\mathbf{P}$ (2° livello di **PH**)
- Confrontare con quelle hard per il 1° livello:
 - $\exists \vec{p}. \phi$ ($\Sigma_1\mathbf{P}$) e $\forall \vec{p}. \phi$ ($\Pi_1\mathbf{P}$)
- Il numero di alternanze tra i quantificatori corrisponde ai livelli. Questo NON è un caso

QSAT_i is Σ_i P-complete

Definizione di QSAT_i

Data una QBF con i alternanze di quantificatori

$$\exists p_{1,1} \dots \exists p_{1,n_1} \forall p_{2,1} \dots \forall p_{2,n_2} \exists p_{3,1} \dots \phi$$

rappresentata più concisamente come

$$\exists \vec{p}_1 \forall \vec{p}_2 \exists \vec{p}_3 \dots Q \vec{p}_i. \phi$$

dove Q è \exists se i è dispari e \forall se i è pari, e dove le variabili in $\vec{p}_1 \dots \vec{p}_i$ coprono tutte quelle di ϕ ,
dire se la QBF è vera.

QSAT_i is $\Sigma_i\mathbf{P}$ -complete

Teorema 17.10

QSAT_i è $\Sigma_i\mathbf{P}$ -completo.

- Quindi decidere la verità delle formule

$$\forall \vec{p}_1 \exists \vec{p}_2 \forall \vec{p}_3 \dots Q \vec{p}_i. \phi$$

è completo per $\Pi_i\mathbf{P}$, perchè è il complemento di QSAT_i

- $\forall \vec{p}_1 \exists \vec{p}_2 \forall \vec{p}_3 \dots Q \vec{p}_i. \phi$ è vera sse $\exists \vec{p}_1 \forall \vec{p}_2 \exists \vec{p}_3 \dots Q \vec{p}_i. \neg \phi$ è falsa
- Quindi è il complemento di un problema completo per $\Sigma_i\mathbf{P}$
 - dunque completo per $\Pi_i\mathbf{P} = \text{co}\Sigma_i\mathbf{P}$

QSAT_i is $\Sigma_i\mathbf{P}$ -complete

- **Prova (appartenenza):** Ricordare il 2° corollario del Teorema 17.8: QSAT_i $\in \Sigma_i\mathbf{P}$ sse esiste R polynomially balanced e decidibile in tempo polinomiale tale che

$$\text{QSAT}_i = \{x \mid \exists y_1 \forall y_2 \exists y_3 \dots Q y_i \text{ tale che } (x, y_1, \dots, y_i) \in R\}$$

dove le x hanno la forma $\exists \vec{p}_1 \forall \vec{p}_2 \exists \vec{p}_3 \dots Q \vec{p}_i. \phi$

- Scegliamo come y_j un assegnamento di verità alle variabili \vec{p}_j
 - y_1, \dots, y_i è un assegnamento di verità per tutte le variabili di ϕ
 - definiamo $(x, y_1, \dots, y_i) \in R$ sse y_1, \dots, y_i soddisfa ϕ (si può verificare in tempo polinomiale)
- Chiaramente x è vera sse $\exists y_1 \forall y_2 \exists y_3 \dots Q y_i$ tali che $(x, y_1, \dots, y_i) \in R$
 - quindi per il Corollario QSAT_i è in $\Sigma_i\mathbf{P}$

QSAT_i is $\Sigma_i\mathbf{P}$ -complete

- **Prova (hardness)** Dimostriamo simultaneamente che decidere la verità di formule della forma:
 - $\exists \vec{p}_1 \forall \vec{p}_2 \dots Q \vec{p}_i. \phi$ (QSAT_i) è completo per $\Sigma_i\mathbf{P}$.
 - $\forall \vec{p}_1 \exists \vec{p}_2 \dots Q \vec{p}_i. \phi$ ($\overline{\text{QSAT}}_i$) è completo per $\Pi_i\mathbf{P}$.
- Consideriamo prima i dispari. Dato un qualunque $L \in \Sigma_i\mathbf{P}$, dobbiamo ridurre L a QSAT_i. Per il 2° Corollario del 17.8

$$L = \{x \mid \exists y_1 \forall y_2 \dots Q y_i \text{ tali che } (x, y_1, \dots, y_i) \in R\}$$

per qualche R polynomially balanced e decidibile in tempo polinomiale

- Consideriamo la MdT M_R che decide R

(segue)

QSAT_i is Σ_i P-complete

Prova di hardness (segue)

- Le computazioni di M_R possono essere codificate con una espressione booleana ϕ in tempo $O(\log n)$
 - come nel Teorema di Cook [lezione-6.pdf, slides 46 e segg.]
- Le proposizioni $P_{s,1}^i$ descrivono il nastro di input
 - per $s = 1, \dots, |x|$ descrivono l'istanza x
 - valore di verità fissato come in [lezione-6.pdf, slide 50]
 - segue un gruppo di proposizioni Y_1 che rappresenta y_1
($s = |x| + 2, \dots, |x| + 1 + |y_1|$)
 - segue un gruppo di proposizioni Y_2 che rappresenta y_2
 - così via fino a Y_i che rappresenta y_i
- Sia V l'insieme di tutte le altre proposizioni ($P_{s,t}^i$ con $t > 1$, Q_t^i , $S_{s,t}$)

(segue)

QSAT_i is Σ_i P-complete

Prova di hardness (segue)

- Per decidere se $x \in L$ dobbiamo verificare se M_R accetta (x, y_1, \dots, y_i) (equivalentemente, ϕ è soddisfacibile)
 - per qualche y_1 (cioè qualche valore delle proposizioni in Y_1)
 - per ogni y_2 (quindi per ogni valore delle proposizioni in Y_2)
 - e via così fino a $\exists y_i$ (i è dispari)
 - per ogni valore di y_1, \dots, y_i così determinato deve esistere un assegnamento alle variabili in V che soddisfa ϕ
- Questo equivale a verificare se è vera la QBF

$$\exists Y_1 \forall Y_2 \dots \exists Y_i \forall V. \phi$$

che è una istanza di QSAT_i

(segue)

QSAT_i is $\Sigma_i\mathbf{P}$ -complete

Prova di hardness (segue)

- La completezza di $\overline{\text{QSAT}}_i$ per $\Pi_i\mathbf{P}$ per i dispari segue dalla completezza di QSAT_i per $\Sigma_i\mathbf{P}$
 - in quanto $\Pi_i\mathbf{P} = \text{co}\Sigma_i\mathbf{P}$
- Per i pari, iniziamo col mostrare la completezza di $\overline{\text{QSAT}}_i$ per $\Pi_i\mathbf{P}$
 - la completezza di QSAT_i per $\Sigma_i\mathbf{P}$ segue come sopra
- Con lo stesso metodo usato per i dispari possiamo ridurre ogni $L \in \Pi_i\mathbf{P}$ al controllo di verità su una QBF della forma

$$\forall Y_1 \exists Y_2 \dots \exists Y_i \forall V. \phi$$

QED

Esercizi

Dire se esistono riduzioni tra i seguenti problemi, motivando la risposta

- $QSAT_2$ e $QSAT_4$
- $QSAT_i$ e \overline{QSAT}_i
- CLIQUE e $QSAT_1$
- CLIQUE e $QSAT_2$
- CLIQUE e \overline{QSAT}_1
- CLIQUE e \overline{QSAT}_2
- SAT-UNSAT e $QSAT_3$
- SAT-UNSAT e \overline{QSAT}_3

Considerare entrambe le direzioni

Risposte possibili: “sì”, “no”, “solo se (indicare le conseguenze)”, “non si sa”

Esercizi

Supponendo che i livelli di **PH** siano tutti distinti, dire per quali i esistono riduzioni

- da $QSAT_i$ a $QSAT_{i+1}$
- da $QSAT_i$ a \overline{QSAT}_{i+1}
- da $QSAT_i$ a \overline{QSAT}_i
- da \overline{QSAT}_i a $QSAT_{i+1}$
- da $QSAT_i$ a $TSP(D)$
- da $QSAT_i$ a $VALIDITY$

CIRC SAT è $\Sigma_2\mathbf{P}$ -completo

Hardness: riduzione da QSAT_2

Teorema

Date KB e ϕ , decidere se ϕ è vera in qualche modello di KB (secondo la circumscription) è $\Sigma_2\mathbf{P}$ -completo

- **Prova:** Iniziamo con la hardness: data $\Phi = \exists x_1 \dots \exists x_n \forall y_1 \dots \forall y_m \cdot \phi$ mostriamo una riduzione a CIRC SAT:
- Definiamo KB come l'insieme di formule ($1 \leq i \leq n$, $1 \leq j \leq m$):

$$x_i \vee x'_i \quad y_j \vee y'_j \quad \phi \rightarrow y_j \wedge y'_j$$

dove le x'_i e le y'_j sono nuove proposizioni. La query è ϕ

- Mettiamo in M (proposizioni da "minimizzare") tutte le proposizioni
 - i modelli massimamente normali di KB saranno quelli in cui quante più proposizioni possibili sono false (segue)

CIRC SAT è $\Sigma_2\mathbf{P}$ -completo

Hardness: riduzione da QSAT_2 – segue

- Proprietà di qualunque modello T di KB (assegnamento di verità \preceq_M -minimo tra quelli che soddisfano KB)
 - 1 Soddisfa solo x_i o solo x'_i . Almeno uno per soddisfare $x_i \vee x'_i$. Solo uno sennò T non sarebbe \preceq_M -minimo
 - verificare che se li soddisfacesse tutti e due il T' uguale a T salvo che per $T'(x'_i) = \text{false}$ soddisfa ancora KB e $T' \prec_M T$
 - 2 Se $T \models \phi$ allora $T \models y_i \wedge y'_i$ grazie a $\phi \rightarrow y_i \wedge y'_i$ in KB

(segue)

CIRC SAT è $\Sigma_2\mathbf{P}$ -completo

Hardness: riduzione da QSAT_2 – segue

- Ora mostriamo che Φ è vera sse ψ è vera in qualche modello (massimamente normale) di KB
 - Iniziamo supponendo che Φ sia vera.
 - Sia T_x un assegnamento alle x_i tale che per ogni sua estensione T_{xy} alle y_1, \dots, y_m rende vera ϕ (*)
 - Estendiamo T_{xy} a un T per tutte le proposizioni di KB ponendo
 - $T(x'_i) = \neg T(x_i)$, $T(y'_i) = T(y_i) = \text{true}$
- Verificare che $T \models KB$

(segue)

CIRC SAT è Σ_2 P-completo

Hardness: riduzione da QSAT₂ – segue

- Mostriamo che T è \preceq_M -minimo provando che per ogni $T' \preceq_M T$ tale che $T' \models KB$, $T = T'$
- T e T' devono concordare sulle x_i e x'_i
 - prop. 1: T soddisfa solo x_i o x'_i , quindi T' non può migliorare T falsificando l'unico rimasto perchè falsificherebbe $x_i \vee x'_i \in KB$
- Quindi per costruzione di T_x , qualunque siano i valori $T'(y_i)$, $T' \models \phi$
- Per le $\phi \rightarrow y_i \wedge y'_i$ in KB , T' soddisfa tutte le y_i e y'_i , quindi $T' = T$
- Concludiamo che T è un modello massimamente normale di KB , e che soddisfa ϕ

(segue)

CIRC SAT è $\Sigma_2\mathbf{P}$ -completo

Hardness: riduzione da QSAT_2 – segue

- Resta da dimostrare che se ϕ è vera in qualche modello (massimamente normale) T di KB allora Φ è vera
- Indichiamo con T_x la restrizione di T alle x_i .
- Per mostrare che Φ è vera, basta provare che ogni estensione di T_x alle y_i soddisfa ϕ
- Per assurdo, supponiamo che una di queste estensioni, T_{xy} , falsifichi ϕ (otterremo una contraddizione)
- Verificare che T_{xy} può essere esteso a un assignment T' che soddisfa KB :
 - porre $T'(x'_i) = \neg T_{xy}(x_i)$, $T'(y'_i) = \neg T_{xy}(y_i)$, $T'(w) = \text{false}$
- Verificare che $T' \prec_M T$ (assurdo, perchè T è \preceq_M -minimo!)

(fine della prova di hardness)

CIRC SAT è $\Sigma_2\mathbf{P}$ -completo

Appartenenza: algoritmo nondeterministico polinomiale con oracolo in **NP**

- Oracolo: data KB e un truth assignment T dire se T non è un modello \preceq_M -minimo di KB
- L'oracolo è in **NP**
 - 1 generare nondeterministicamente un T'
 - 2 verificare in tempo polinomiale se $T' \models KB$ e se $T' \prec_M T$
- Algoritmo nondeterministico polinomiale con oracolo per CIRC SAT:
 - 1 generare nondeterministicamente un T
 - 2 verificare se T soddisfa la query ϕ in tempo polinomiale
 - 3 verificare con l'oracolo se T è un modello \preceq_M -minimo di KB

QED

STRATEGIC COMPANIES è $\Sigma_2\mathbf{P}$ -completo

Prova di hardness – riduzione da QSAT_2

- Dobbiamo ridurre una qualunque QBF $\Phi = \exists x_1 \dots \exists x_n \forall y_1 \dots \forall y_m \cdot \phi$ a STRATEGIC COMPANIES
 - possiamo assumere che ϕ sia una disgiunzione di k congiunzioni di letterali $L_{i1} \wedge L_{i2} \wedge L_{i3}$ – il problema resta $\Sigma_2\mathbf{P}$ -hard
- Riduzione: $C = \{x_1, \dots, x_n, \neg x_1, \dots, \neg x_n, y_1, \dots, y_m, \neg y_1, \dots, \neg y_m, w\}$,
 - $G = \{gx_1, \dots, gx_n, gy_1, \dots, gy_m\}$
 - $G(x_i) = G(\neg x_i) = \{gx_i\}$, $G(y_i) = G(\neg y_i) = \{gy_i\}$,
 - $O(x_i) = \{\{x_i\}\}$, $O(\neg x_i) = \{\{\neg x_i\}\}$,
 $O(y_i) = O(\neg y_i) = \{\{w\}\}$,
 $O(w) = \{\{L_{i1}, L_{i2}, L_{i3}\} \mid i = 1 \dots k\}$

(segue)

STRATEGIC COMPANIES è $\Sigma_2\mathbf{P}$ -completo

Prova di hardness – riduzione da QSAT_2 – segue

- (Correttezza) vogliamo dimostrare che Φ è vera sse w è strategica
 - Supponiamo prima che Φ sia vera e mostriamo che w è strategica
 - Dato un qualunque assignment T per x_1, \dots, x_n che rende vera Φ , sia $C' \subseteq C$ il seguente insieme:
 - contiene x_i sse $T(x_i) = \text{true}$ e $\neg x_i$ sse $T(x_i) = \text{false}$
 - contiene $y_1, \dots, y_m, \neg y_1, \dots, \neg y_m, w$
 - Questo C' contiene w e ha le seguenti proprietà
 - $\bigcup_{c \in C'} G(c) = G$ e per ogni $O_i \in O(c)$, $O_i \subseteq C' \Rightarrow c \in C'$
- quindi per mostrare che w è strategica resta solo da mostrare che C' è \subseteq -minimo tra quelli con le suddette proprietà

(segue)

STRATEGIC COMPANIES è $\Sigma_2\mathbf{P}$ -completo

Prova di hardness – riduzione da QSAT_2 – segue

- (Correttezza – segue) Supponiamo quindi per assurdo che esista $C'' \subset C'$ minimo, con le stesse proprietà
- C'' deve contenere le stesse x_i e $\neg x_i$ di C'
 - togliendone una perdiamo un bene, violando la 1^a condizione
 - infatti per def. C' contiene 1 elemento per ogni coppia $x_i, \neg x_i$
 - e $g x_i$ è prodotto solo da quelle due compagnie
- quindi C'' deve perdere qualche elemento tra i $y_i, \neg y_i$
 - al massimo 1 per coppia, senno' perdiamo un bene
- questi dipendono tutti da w quindi – per rispettare la proprietà di chiusura – deve essere $w \notin C''$

(segue)

STRATEGIC COMPANIES è $\Sigma_2\mathbf{P}$ -completo

Prova di hardness – riduzione da QSAT_2 – segue

- (Correttezza – segue) Ora, per ogni assegnamento di verità alle y_i che estende T , ϕ è vera
 - quindi qualche disgiunto $L_1 \wedge L_2 \wedge L_3$ di ϕ è vero
- Questo vale anche per tutti gli assegnamenti “contenuti” in C'' :
 - se $T(y_i) = \text{true}$ allora $y_i \in C''$
 - se $T(y_i) = \text{false}$ allora $\neg y_i \in C''$
- Ma allora qualche tripla $\{L_1, L_2, L_3\}$ è contenuta in C''
- E siccome sono tutte in $O(w)$, ma $w \notin C''$, abbiamo che C'' viola la condizione di chiusura (assurdo)
 - Questo dimostra che C' è minimo, quindi w è strategico
 - Dunque se Φ è vera, allora w è strategico

(segue)

STRATEGIC COMPANIES è $\Sigma_2\mathbf{P}$ -completo

Prova di hardness – riduzione da QSAT_2 – segue

- (Correttezza – direzione opposta) Ora assumendo w strategico, ovvero appartenente a un C' minimo che soddisfa le 2 proprietà
 - $\bigcup_{c \in C'} G(c) = G$ e per ogni $O_i \in O(c)$, $O_i \subseteq C' \Rightarrow c \in C'$dobbiamo mostrare che Φ è vera per assurdo
- Definiamo $T(x_i) = \text{true}$ sse $x_i \in C'$ e supponiamo che una sua estensione a y_1, \dots, y_m non soddisfi ϕ
- Usando T , definiremo $C'' \subset C'$ e mostreremo che soddisfa le 2 proprietà
 - quindi C' non è minimo \Rightarrow assurdo!

(segue)

STRATEGIC COMPANIES è $\Sigma_2\mathbf{P}$ -completo

Prova di hardness – riduzione da QSAT_2 – segue

- (Correttezza – segue) Definiamo
 - $C'' = \{v_i \mid T(v_i) = \text{true}\} \cup \{\neg v_i \mid T(v_i) = \text{false}\}$ ($w \notin C''$)
- Verificare che T è “contenuto” in C'' nel senso di prima:
 - se $T(v_i) = \text{true}$ allora $y_i \in C''$
 - se $T(v_i) = \text{false}$ allora $\neg y_i \in C''$
- Analogamente a quanto visto prima, T “contenuto” in C'' e “ ϕ non soddisfatta da T ” implicano
 - per ogni disgiunto $L_1 \wedge L_2 \wedge L_3$ in ϕ , $\{L_1, L_2, L_3\} \not\subseteq C''$
 - verificare che questo garantisce la proprietà di chiusura per w e gli $y_i, \neg y_i$
- Inoltre per ogni coppia $x_i, \neg x_i$ e $y_i, \neg y_i$ C'' contiene 1 elemento, quindi tutti i beni sono prodotti
- Concludiamo che C'' soddisfa le due proprietà (segue)

STRATEGIC COMPANIES è $\Sigma_2\mathbf{P}$ -completo

Prova di hardness – riduzione da QSAT_2 – segue

■ Conseguenze:

- ⇒ C' non è minimo (assurdo)
- ⇒ l'assegnamento T alle variabili x_i non può essere esteso alle y_i in modo da falsificare ϕ
- ⇒ Φ è vera

QED

STRATEGIC COMPANIES è $\Sigma_2\mathbf{P}$ -completo

Prova di membership

- Verificare se un dato $C' \subseteq C$ non è un insieme minimo che soddisfa le 2 condizioni è in **NP**:
 - 1 Verificare se C' soddisfa le 2 condizioni (in **P**). Se si:
 - 2 generare nondeterministicamente $C'' \subset C'$
 - 3 verificare se C'' soddisfa le 2 condizioni.
 - 4 Se si, restituire “yes”, in tutti gli altri casi “no”
- Risolvere STRATEGIC COMPANIES così:
 - 1 Generare nondeterministicamente $C' \subset C$
 - 2 Verificare se è minimo usando l'oracolo qui sopra
 - 3 Se si, verificare se la compagnia data appartiene a C'

Questo algoritmo rientra in $\mathbf{NP}^{\mathbf{NP}} = \Sigma_2\mathbf{P}$

QED

E i problemi completi per **PH**?

- Abbiamo problemi completi per ogni livello di **PH**
 - Ma non abbiamo ancora problemi completi per *l'intera* **PH**
 - Forse l'insieme di *tutte* le formule QBF ?
 - con qualunque alternanza di quantificatori
- NO! Vedremo che quelle sono complete per **PSPACE**
- In realtà non sono noti problemi completi per **PH** e se ci fossero...

E i problemi completi per **PH**?

Teorema 17.11

Se esiste un problema **PH**-completo, allora **PH** collassa ad un suo livello finito

- **Prova:** Supponiamo che L sia **PH**-completo
- Poichè $L \in \mathbf{PH}$ e $\mathbf{PH} = \bigcup_i \Sigma_i \mathbf{P}$, deve esistere $k \geq 0$ tale che

$$L \in \Sigma_k \mathbf{P}$$

- Ma ogni $L' \in \Sigma_{k+1} \mathbf{P}$ si riduce a L (che è **PH**-completo)
- Quindi (siccome tutti i livelli sono chiusi rispetto alle riduzioni)
 $L' \in \Sigma_k \mathbf{P}$
- Ne segue che $\Sigma_k \mathbf{P} = \Sigma_{k+1} \mathbf{P}$ e per 17.9 **PH** collassa al livello k

QED

PH e PSPACE

Proposizione 17.1

PH \subseteq PSPACE

- **Prova (cenni):** Per ogni $i \geq 0$ e ogni $L \in \Sigma_i \mathbf{P}$

$$L = \{x \mid \exists y_1 \forall y_2 \dots Q y_i \text{ tali che } (x, y_1, \dots, y_i) \in R\}$$

per una opportuna R calcolabile in tempo polinomiale

- R è polynomially balanced, quindi $|y_1, \dots, y_i|$ è polinomiale in x
- Facile vedere che la enumerazione dei y_1, \dots, y_i rientra in spazio polinomiale
 - simile all'incremento di un intero
 - con un nastro lungo i si ricorda, per ogni j , se deve provare tutti i possibili y_j o trovarne uno "giusto"
- Lo spazio per verificare l'appartenenza a R è limitato dal tempo (polinomiale)

QED

PH e PSPACE

Altra versione ricorsiva (in pseudocodice) che usa spazio polinomiale

algorithm *Truth*

input una QBF $Q_1 p_1 \dots Q_n p_n \cdot \phi(p_1, \dots, p_n)$

output il valore di verità della QBF

if $n = 0$ restituire il valore di ϕ // non ci sono proposizioni

$b_0 = \text{Truth}(Q_2 p_2 \dots Q_n p_n, \phi(0, p_2, \dots, p_n))$

$b_1 = \text{Truth}(Q_2 p_2 \dots Q_n p_n, \phi(1, p_2, \dots, p_n))$ // riutilizza lo spazio

if $Q_1 = \exists$ **then return** $b_0 \vee b_1$ **else return** $b_0 \wedge b_1$

end

- Max livello di ricorsione: n
- Spazio occupato da ogni “record di attivazione”: 2 booleani e il parametro ($O(n)$)
- Totale: $O(n^2)$

PH e PSPACE

Corollario di 17.11

Se $\text{PH} = \text{PSPACE}$ allora PH collassa a qualche suo livello finito

- **Prova:** Premessa: **PSPACE** ha problemi completi (slide successiva)
- Quindi se $\text{PH} = \text{PSPACE}$ anche PH avrebbe problemi completi
- Di conseguenza, per il Teorema 17.11, **PH** collaserebbe a qualche suo livello finito.

QED

Siccome si pensa che i livelli siano distinti, si pensa anche che
PH \neq PSPACE

QSAT (senza limiti di alternanza) è **PSPACE**-completo

Definizione di QSAT

Data una QBF qualunque dire se è vera

Teorema

QSAT è **PSPACE**-completo

- **Prova:** L'appartenenza l'abbiamo già dimostrata (per ogni QSAT_i).
Resta solo la hardness

(segue)

QSAT (senza limiti di alternanza) è **PSPACE**-completo

Prova di hardness

- Sia L un qualunque linguaggio in **PSPACE** ed M la MdT che lo decide in spazio $O(n^c)$, per qualche costante c
 - dobbiamo ridurre L a QSAT
- Strategia:
 - 1 costruire il grafo delle configurazioni di M
 - 2 ridurre accettazione di M a reachability
 - 3 applicare algoritmo ricorsivo “alla Savitch” per ottenere una QBF “piccola” che simula reachability
- Sia m la lunghezza max delle configurazioni ($O(n^c)$) espresse in binario
- Il numero di configurazioni (nodi) è 2^m
- Quindi anche il cammino dalla configurazione iniziale a quella di accettazione è lungo al max 2^m

(segue)

QSAT (senza limiti di alternanza) è **PSPACE**-completo

Prova di hardness – segue

- Definiamo una sequenza di QBF $\psi_i(A, B)$ ($i = 1, \dots, m$) che sono *true* sse le variabili A e B codificano due configurazioni a e b connesse da un cammino lungo 2^i
 - simili al predicato $PATH(a, b, i)$ del Teorema di Savitch
- Per $i = 0$, $\psi_0(X, Y)$ è una espressione booleana vera sse le sue variabili libere X, Y codificano due configurazioni x, y uguali o tali che $x \xrightarrow{M} y$
 - codifica delle configurazioni e verifica simili al Teorema di Cook ristretto a una matrice di 2 sole righe

(segue)

QSAT (senza limiti di alternanza) è PSPACE-completo

Prova di hardness – segue

- Primo tentativo di costruzione induttiva:
 - Per $i + 1$, $\psi_{i+1}(X, Y) = \exists Z. [\psi_i(X, Z) \wedge \psi_i(Z, Y)]$
 - Ma così le ψ_i crescono esponenzialmente con i !
- Costruzione che rispetta i limiti di spazio e tempo delle riduzioni

$$\begin{aligned} \psi_{i+1}(X, Y) = \\ \exists Z_{i+1} \forall X_{i+1} \forall Y_{i+1}. [\\ ((X_{i+1} = X \wedge Y_{i+1} = Z_{i+1}) \vee (X_{i+1} = Z_{i+1} \wedge Y_{i+1} = Y)) \\ \rightarrow \psi_i(X_{i+1}, Y_{i+1})] \end{aligned}$$

- Nota: $\psi_{i+1}(X, Y)$ è vera sse per qualche valore di Z_{i+1} ,
 $\psi_i(X, Z_{i+1}) \wedge \psi_i(Z_{i+1}, Y)$ è vera (segue)

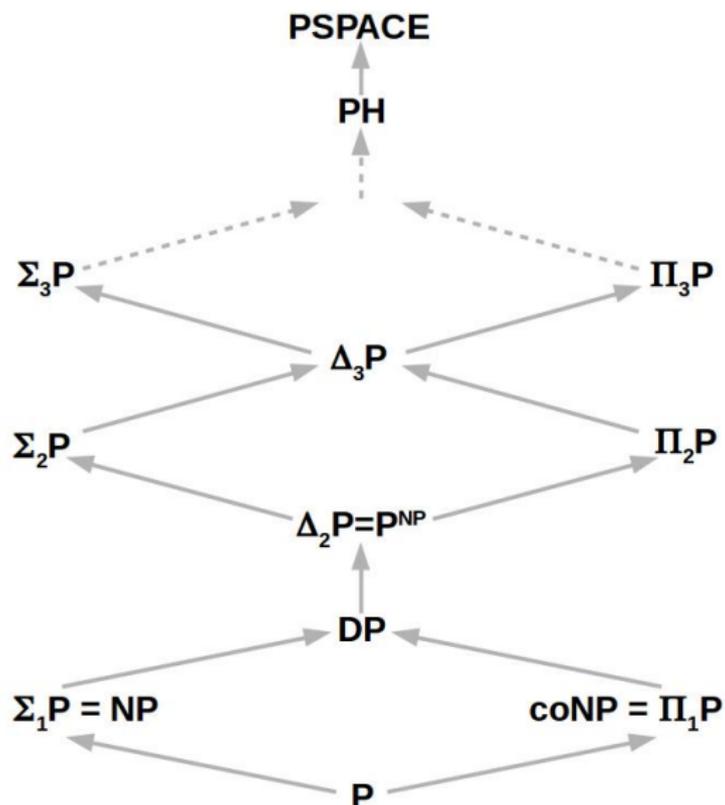
QSAT (senza limiti di alternanza) è **PSPACE**-completo

Prova di hardness – segue

- Calcolo delle $\psi_i(X, Y)$ in spazio $O(\log n)$:
- gli indici delle proposizioni possono essere rappresentati in spazio $O(\log m) = O(\log n^c) = O(n)$

QED

Riassunto



Esercizi

Supponendo che **PH** non abbia problemi completi dire se esistono riduzioni tra i seguenti problemi, motivando la risposta

- $QSAT_1$ e $QSAT_4$
- $QSAT_i$ e $\overline{QSAT_i}$
- CLIQUE e $QSAT_1$
- CLIQUE e CIRC SAT
- CLIQUE e $\overline{QSAT_1}$
- CLIQUE e $\overline{STRATEGIC COMPANIES}$
- $QSAT$ e $QSAT_3$
- $QSAT$ e $\overline{HAMILTON PATH}$

Considerare entrambe le direzioni

Risposte possibili: “sì”, “no”, “solo se (indicare le conseguenze)”, “non si sa”

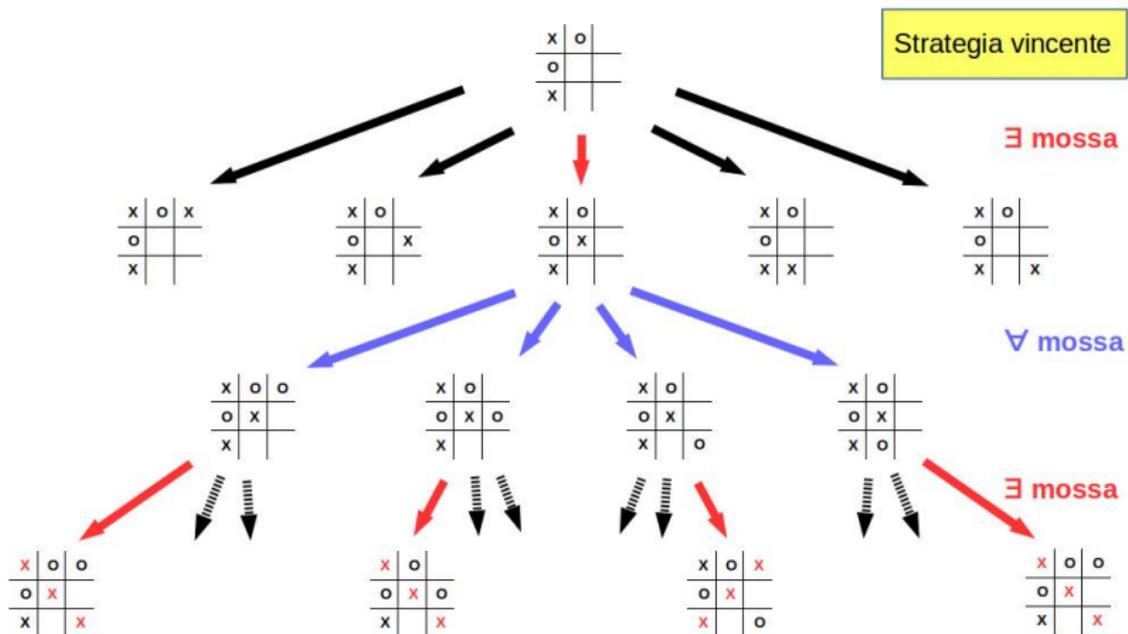
Alternanza e giochi

Giochi

- In termini matematici un gioco è essenzialmente un grafo
 - rappresentato esplicitamente o implicitamente
 - spesso in forma di albero
- I nodi sono analoghi alle *configurazioni* di una MdT
 - rappresentano uno stato
 - ad es. pezzi su di una scacchiera
- Gli archi sono analoghi alle *transizioni* di una MdT nondeterministica
 - rappresentano le *possibili* mosse di un giocatore
- Nei giochi a 2 giocatori classici i turni si alternano

Giochi

Esempio tris (a.k.a. tic-tac-toe)



Giochi e PSPACE

- È chiara l'analogia tra lo studio delle strategie di gioco e l'alternanza di quantificatori delle QBF
 - che caratterizzano **PSPACE**
- Giochi con num. costante di configurazioni: “poco interessanti”
 - il loro grafo di configurazioni *in teoria* può essere completamente esplorato in tempo costante ($O(1)$)
- Considerate però gli scacchi...
 - numero di configurazioni confrontabile col numero stimato di atomi nell'universo
 - attualmente non è noto se esista una strategia vincente

Giochi e PSPACE

- Nella teoria della complessità siamo più interessati ai giochi a dimensione variabile
 - magari generalizzazioni di quelli noti a scacchiere più grandi
- Qui consideriamo giochi
 - a 2 giocatori, turni alterni
 - dove il numero di mosse è polinomiale nella dimensione della configurazione
- Aspetto interessante: cos'è una *soluzione*?
 - deve rappresentare in modo compatto la strategia
 - cioè cosa fare in ciascuna di un numero esponenziale di situazioni

The game of Geography

- Il primo player indica una città – diciamo Roma
- Il secondo deve rispondere con una città il cui nome inizia per 'a' (ad es. Alessandria) e così via
- Non si può citare 2 volte la stessa città. Perde chi non ha più città disponibili

Generalizzazione del gioco

Dato un grafo $G = (V, E)$ ($V \approx$ città; $(x, y) \in E$ se y può essere nominata dopo x)

i giocatori scelgono alternativamente un nodo collegato al precedente e non già menzionato;

perde chi non può ulteriormente estendere il cammino.

The game of Geography

Definizione di GEOGRAPHY

Dato un grafo G esiste una strategia vincente per il 1° giocatore a partire dal nodo 1?

Teorema 19.3

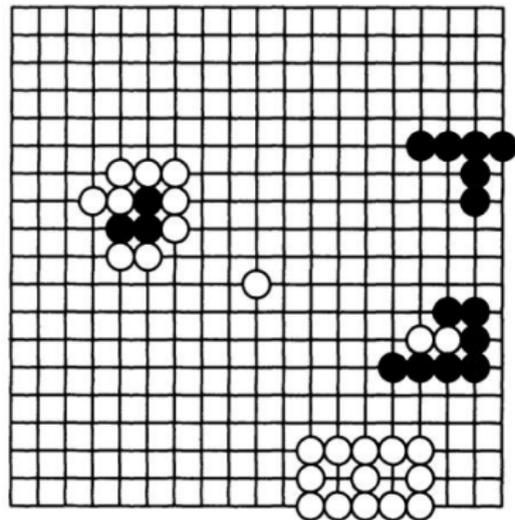
GEOGRAPHY è **PSPACE** completo

Go

- Scacchiera 19×19
- I due giocatori alternativamente mettono pedine bianche e nere sulla scacchiera
- cercando di catturare le pedine dell'avversario
 - circondandole
 - vengono rimosse
- vince chi al termine ha più pedine sulla scacchiera

Go

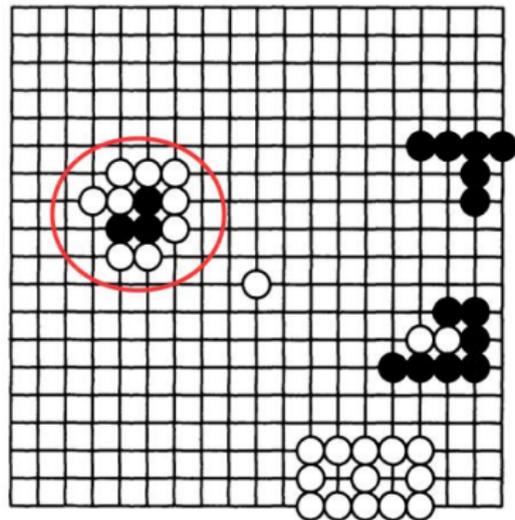
Esempio



- Esempio di configurazione

Go

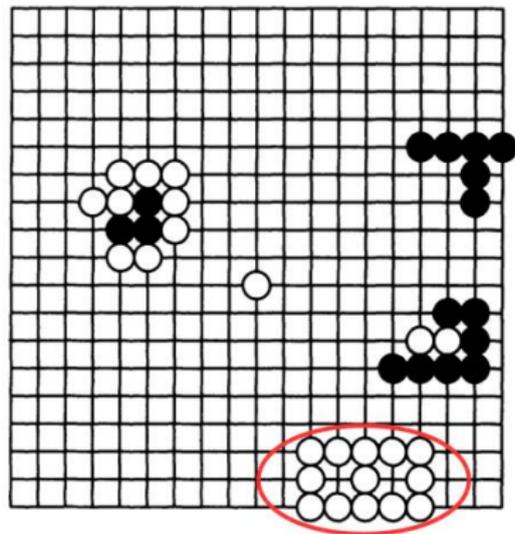
Esempio



- Gruppo nero a rischio

Go

Esempio



- Gruppo bianco intoccabile

Go

Generalizzazione di GO

Data una scacchiera $n \times n$ e una configurazione, dire se il nero ha una strategia vincente

Teorema 19.4

GO è **PSPACE**-completo

Esercizio

Dire se

- 1 GEOGRAPHY può essere ridotto al calcolo del valore di verità di una QBF Φ
- 2 se sì, dire se esiste un unico intero k che limita l'alternanza di quantificatori in Φ per tutte le istanze di GEOGRAPHY

Risposte possibili: “sì”, “no”, “solo se ⟨conseguenze⟩”, “non si sa”

Motivare la risposta

Materiale di riferimento

- Papadimitriou: Parte 5, Capitolo 17, paragrafi 1 e 2 + Capitolo 19 paragrafo 1 (solo le parti menzionate nelle slides)
- Dispense lec14.pdf: paragrafi 1 e 2 tranne Def. 2 e Corollario 3
- Al momento non c'è materiale di riferimento "pronto all'uso" per Circumscription e Strategic Companies
 - contattare il docente per ogni necessità di chiarimento