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

- 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 EXACT TSP $\in$ 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, EXACT TSP $\in$ 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 EXACT TSP $\in$ NP $\cup$ coNP allora le conseguenze sarebbero eccezionali!

Classificazione dei problemi di ottimizzazione

- Ma EXACT TSP $\in$ 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, EXACT TSP $\in$ 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 EXACT TSP $\in$ NP $\cup$ coNP allora le conseguenze sarebbero eccezionali!

Classificazione dei problemi di ottimizzazione

- Ma EXACT TSP $\in$ 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, EXACT TSP $\in$ 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 EXACT TSP $\in$ NP $\cup$ coNP allora le conseguenze sarebbero eccezionali!

Analisi di EXACT TSP

- EXACT TSP è l'<u>intersezione</u> di due linguaggi
  - L_1: grafi il cui tour ottimo ha costo $\leq B$
  - L_2: grafi il cui tour ottimo ha costo $\geq B$

Analisi di EXACT TSP

- EXACT TSP è l'<u>intersezione</u> di due linguaggi
  - L_1: grafi il cui tour ottimo ha costo $\leq B$ (TSP(D))
  - L_2: grafi il cui tour ottimo ha costo $\geq B$ ($\overline{\text{TSP(D)}}$)
- tali che $L_1 \in$ NP e $L_2 \in$ coNP
- Quindi decidere se $x \in$ 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.)

Analisi di EXACT TSP

- EXACT TSP è l'<u>intersezione</u> di due linguaggi
  - L_1: grafi il cui tour ottimo ha costo $\leq B$ (TSP(D))
  - L_2: grafi il cui tour ottimo ha costo $\geq B$ $(\overline{\text{TSP(D)}})$
- tali che $L_1 \in \text{NP}$ e $L_2 \in \text{coNP}$
- Quindi decidere se $x \in$ 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.)

Analisi di EXACT TSP

- EXACT TSP è l'intersezione di due linguaggi
  - L_1: grafi il cui tour ottimo ha costo $\leq B$ (TSP(D))
  - L_2: grafi il cui tour ottimo ha costo $\geq B$ ($\overline{\text{TSP(D)}}$)
- tali che $L_1 \in$ NP e $L_2 \in$ coNP
- Quindi decidere se $x \in$ 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 $L_1 \in$ NP e $L_2 \in$ coNP tali che $L = L_1 \cap L_2$

- ATTENZIONE: non confondere DP con NP $\cap$ coNP
  - $L \in$ DP significa che le *istanze "yes"* di L sono in $L_1 \cap L_2$
  - $L \in$ NP $\cap$ coNP significa che L *stesso* appartiene a NP $\cap$ 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 $L \in$ DP tali che $L \notin$ NP $\cup$ coNP

La classe DP

Definizione di DP

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

- **ATTENZIONE**: non confondere DP con NP $\cap$ coNP
  - $L \in$ DP significa che le *istanze "yes"* di L sono in $L_1 \cap L_2$
  - $L \in$ NP $\cap$ coNP significa che L *stesso* appartiene a NP $\cap$ 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")

La classe DP

Definizione di DP

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

- **ATTENZIONE**: non confondere DP con $\text{NP} \cap \text{coNP}$
  - $L \in \text{DP}$ significa che le *istanze "yes"* di L sono in $L_1 \cap L_2$
  - $L \in \text{NP} \cap \text{coNP}$ significa che L *stesso* appartiene a $\text{NP} \cap \text{coNP}$
  - 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 $L \in \text{DP}$ tali che $L \notin \text{NP} \cup \text{coNP}$

La classe DP

Definizione di DP

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

- **ATTENZIONE**: non confondere DP con $\text{NP} \cap \text{coNP}$
  - $L \in \text{DP}$ significa che le *istanze "yes"* di L sono in $L_1 \cap L_2$
  - $L \in \text{NP} \cap \text{coNP}$ significa che L *stesso* appartiene a $\text{NP} \cap \text{coNP}$
  - 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 $L \in \text{DP}$ tali che $L \notin \text{NP} \cup \text{coNP}$

La classe DP

Definizione di DP

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

- **ATTENZIONE**: non confondere DP con $\mathrm{NP} \cap \mathrm{coNP}$
  - $L \in \mathrm{DP}$ significa che le *istanze "yes"* di L sono in $L_1 \cap L_2$
  - $L \in \mathrm{NP} \cap \mathrm{coNP}$ significa che L *stesso* appartiene a $\mathrm{NP} \cap \mathrm{coNP}$
  - 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 $L \in \mathrm{DP}$ tali che $L \notin \mathrm{NP} \cup \mathrm{coNP}$

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?

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

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 SAT-UNSAT $\in$ DP bisogna riformularlo come intersezione di due linguaggi in NP e coNP
- Semplice:
  - $L_1 = \{(\phi, \phi') \mid \phi$ è soddisfacibile$\}$
  - $L_2 = \{(\phi, \phi') \mid \phi'$ 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 SAT-UNSAT $= L_1 \cap L_2$

SAT-UNSAT è DP-completo

Appartenenza a DP

- Per mostrare che SAT-UNSAT $\in$ DP bisogna riformularlo come intersezione di due linguaggi in NP e coNP
- Semplice:
  - $L_1 = \{(\phi, \phi') \mid \phi$ è soddisfacibile$\}$
  - $L_2 = \{(\phi, \phi') \mid \phi'$ 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 SAT-UNSAT $= L_1 \cap L_2$

SAT-UNSAT è DP-completo

Appartenenza a DP

- Per mostrare che SAT-UNSAT $\in$ DP bisogna riformularlo come intersezione di due linguaggi in NP e coNP
- Semplice:
  - $L_1 = \{(\phi, \phi') \mid \phi$ è soddisfacibile$\}$
  - $L_2 = \{(\phi, \phi') \mid \phi'$ 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 SAT-UNSAT $= L_1 \cap L_2$

SAT-UNSAT è DP-completo

Appartenenza a DP

- Per mostrare che SAT-UNSAT $\in$ DP bisogna riformularlo come intersezione di due linguaggi in NP e coNP
- Semplice:
  - $L_1 = \{(\phi, \phi') \mid \phi$ è soddisfacibile$\}$
  - $L_2 = \{(\phi, \phi') \mid \phi'$ 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 SAT-UNSAT $= L_1 \cap L_2$

SAT-UNSAT è DP-completo

DP-hardness

- Sia dato un qualunque $L \in \mathrm{DP}$
  - quindi $L = L_1 \cap L_2$ per qualche $L_1 \in \mathrm{NP}$ e $L_2 \in \mathrm{coNP}$

  dobbiamo ridurre L a SAT-UNSAT

- Esistono due riduzioni $R_1 : L_1 \to \mathrm{SAT}$ e $R_2 : L_2 \to \overline{\mathrm{SAT}}$
  - (perchè SAT è NP-completo e $\overline{\mathrm{SAT}}$ coNP-completo)

  Porre $R(x) = (R_1(x), R_2(x))$ – chiaramente è in **L**

- Per le proprietà di R_1, R_2, abbiamo la correttezza:

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

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

QED

SAT-UNSAT è DP-completo

DP-hardness

- Sia dato un qualunque $L \in \mathrm{DP}$
  - quindi $L = L_1 \cap L_2$ per qualche $L_1 \in \mathrm{NP}$ e $L_2 \in \mathrm{coNP}$

  dobbiamo ridurre L a SAT-UNSAT
- Esistono due riduzioni $R_1 : L_1 \to \mathrm{SAT}$ e $R_2 : L_2 \to \overline{\mathrm{SAT}}$
  - (perchè SAT è NP-completo e $\overline{\mathrm{SAT}}$ coNP-completo)

  Porre $R(x) = (R_1(x), R_2(x))$ – chiaramente è in **L**
- Per le proprietà di R_1, R_2, abbiamo la correttezza:

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

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

QED

SAT-UNSAT è DP-completo

DP-hardness

- Sia dato un qualunque $L \in \mathrm{DP}$
  - quindi $L = L_1 \cap L_2$ per qualche $L_1 \in \mathrm{NP}$ e $L_2 \in \mathrm{coNP}$

  dobbiamo ridurre L a SAT-UNSAT
- Esistono due riduzioni $R_1 : L_1 \to \mathrm{SAT}$ e $R_2 : L_2 \to \overline{\mathrm{SAT}}$
  - (perchè SAT è NP-completo e $\overline{\mathrm{SAT}}$ coNP-completo)

  Porre $R(x) = (R_1(x), R_2(x))$ – chiaramente è in **L**
- Per le proprietà di R_1, R_2, abbiamo la correttezza:

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

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

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

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

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

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

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

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

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 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 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:

$$(q_?, w_1, u_1, \ldots, w_k, u_k) \stackrel{M}{\rightarrow} (q_{ans}, w_1, u_1, \ldots, w_k, u_k)$$

  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

MdT con oracolo

Definizione formale di computazione

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

$$(q_?, w_1, u_1, \ldots, w_k, u_k) \stackrel{M}{\rightarrow} (q_{ans}, w_1, u_1, \ldots, w_k, u_k)$$

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

  - $q_{ans} = q_{YES}$ 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 nastri[1]
- Così il costo della soluzione di L viene completamente scorporato
  - a parte la scrittura della query

[1] È 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 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 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:
- Spazio:
- Il costo della soluzione dei due sottoproblemi risulta scorporato

Esempio di MdT per SAT-UNSAT

Fare per esercizio

- Misura delle risorse:
- Tempo: $O(n)$ (copiatura dell'input su query)
- Spazio:
- Il costo della soluzione dei due sottoproblemi risulta scorporato

Esempio di MdT per SAT-UNSAT

Fare per esercizio

- Misura delle risorse:
- Tempo: $O(n)$ (copiatura dell'input su query)
- Spazio:
- Il costo della soluzione dei due sottoproblemi risulta scorporato

Esempio di MdT per SAT-UNSAT

Fare per esercizio

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

Esempio di MdT per SAT-UNSAT

Fare per esercizio

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

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: $\mathrm{P}^{\mathrm{SAT}}$, $\mathrm{NP}^{\mathrm{TSP(D)}}$, $\mathrm{EXP}^{\mathrm{SAT}}$, ...

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: $\mathsf{P}^{\mathrm{SAT}}$, $\mathsf{NP}^{\mathrm{TSP(D)}}$, $\mathsf{EXP}^{\mathrm{SAT}}$, ...

Classi con oracolo

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

Proposizione

Per qualunque L NP-completo, $\mathrm{P}^L = \mathrm{P}^{\mathrm{SAT}}$

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

Similmente per ogni $M'^{\,\mathrm{SAT}}$ 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:
  - P^{NP}, $\mathrm{NP}^{\mathrm{NP}}$, $\mathrm{EXP}^{\mathrm{NP}}$, invece di $\mathrm{P}^{\mathrm{SAT}}$, $\mathrm{NP}^{\mathrm{TSP(D)}}$, $\mathrm{EXP}^{\mathrm{CLIQUE}}$, ...

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:
  - P^{NP}, $\mathsf{NP}^{\mathsf{NP}}$, $\mathsf{EXP}^{\mathsf{NP}}$, invece di $\mathsf{P}^{\mathrm{SAT}}$, $\mathsf{NP}^{\mathrm{TSP(D)}}$, $\mathsf{EXP}^{\mathrm{CLIQUE}}$, ...

Classi con oracolo

I complementari non contano

Proposizione

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

- Prova: ogni MdT $M^{\mathcal{D}}$ si trasforma in una M'^{coD} 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 $\mathrm{P}^{\mathrm{coNP}}$, $\mathrm{NP}^{\mathrm{coNP}}$, $\mathrm{EXP}^{\mathrm{coNP}}$, ...

Classi con oracolo

I complementari non contano

Proposizione

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

- Prova: ogni MdT $M^{\mathcal{D}}$ si trasforma in una M'^{coD} 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 $\mathrm{P}^{\mathrm{coNP}}$, $\mathrm{NP}^{\mathrm{coNP}}$, $\mathrm{EXP}^{\mathrm{coNP}}$, ...

Classi con oracolo

I complementari non contano

Proposizione

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

- Prova: ogni MdT $M^{\mathcal{D}}$ si trasforma in una M'^{coD} 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 $\mathrm{P}^{\mathrm{coNP}}$, $\mathrm{NP}^{\mathrm{coNP}}$, $\mathrm{EXP}^{\mathrm{coNP}}$, ...

Prima generalizzazione di DP

- P^{NP} è proprio la generalizzazione di DP di cui parlavamo
  - Classe dei problemi risolubili 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 $DP \subseteq P^{NP}$

Prima generalizzazione di DP

- P^{NP} è proprio la generalizzazione di DP di cui parlavamo
  - Classe dei problemi risolubili 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 $\mathrm{DP} \subseteq \mathrm{P}^{\mathrm{NP}}$

Oracoli inutili (?)

- $\mathsf{P} = \mathsf{P}^{\mathsf{P}}$?
  - 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)
- $\mathsf{NP} = \mathsf{NP}^{\mathsf{P}}$?
- $\mathsf{NP} = \mathsf{NP}^{\mathsf{NP}}$?

Oracoli inutili (?)

- $\mathrm{P} = \mathrm{P}^{\mathrm{P}}$?
  - Si! 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)

- $\mathrm{NP} = \mathrm{NP}^{\mathrm{P}}$?

- $\mathrm{NP} = \mathrm{NP}^{\mathrm{NP}}$?

Oracoli inutili (?)

- $\mathrm{P} = \mathrm{P}^{\mathrm{P}}$?
  - 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)

- $\mathrm{NP} = \mathrm{NP}^{\mathrm{P}}$?
  - Sì! Per la stessa ragione

- $\mathrm{NP} = \mathrm{NP}^{\mathrm{NP}}$?

Oracoli inutili (?)

- $\mathsf{P} = \mathsf{P}^{\mathsf{P}}$?
  - 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)

- $\mathsf{NP} = \mathsf{NP}^{\mathsf{P}}$?
  - Sì! Per la stessa ragione

- $\mathsf{NP} = \mathsf{NP}^{\mathsf{NP}}$?

Oracoli inutili (?)

- $\mathrm{P} = \mathrm{P}^{\mathrm{P}}$?
  - Si! 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)
- $\mathrm{NP} = \mathrm{NP}^{\mathrm{P}}$?
  - Si! Per la stessa ragione
- $\mathrm{NP} = \mathrm{NP}^{\mathrm{NP}}$?
  - ah, saperlo...

Oracoli inutili (?)

- $\mathsf{P} = \mathsf{P}^{\mathsf{P}}$?
  - Si! 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)
- $\mathsf{NP} = \mathsf{NP}^{\mathsf{P}}$?
  - Si! Per la stessa ragione
- $\mathsf{NP} = \mathsf{NP}^{\mathsf{NP}}$?
  - ah, saperlo...

La gerarchia polinomiale (Polynomial Hierarchy, PH)

La gerarchia polinomiale (PH)

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

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

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

La gerarchia polinomiale (PH)

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

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

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

La gerarchia polinomiale (PH)

Definizione: Δ_iP, Σ_iP, Π_iP e PH

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

Notare che:

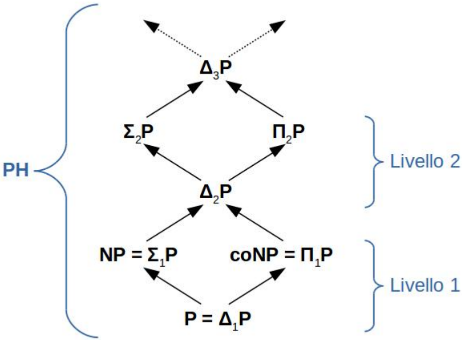
- $\Delta_1 P = P^{\Sigma_0 P} = P^P = P$
- $\Sigma_1 P = NP^{\Sigma_0 P} = NP^P = NP$
- $\Pi_1 P = coNP^{\Sigma_0 P} = coNP^P = coNP$

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

La gerarchia polinomiale (PH)

Definizione: Δ_iP, Σ_iP, Π_iP e PH

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



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

Struttura dei problemi in PH

Teorema 17.8

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

- $\{x; y \mid (x, y) \in R\}$ appartiene a $\Pi_{i-1}\mathrm{P}$ 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

Teorema 17.8

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

- $\{x; y \mid (x, y) \in R\}$ appartiene a $\Pi_{i-1}\text{P}$ 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

Teorema 17.8

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

- $\{x; y \mid (x, y) \in R\}$ appartiene a $\Pi_{i-1}\mathrm{P}$ 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 $L \in \Sigma_i$P

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

(segue)

Struttura dei problemi in PH

Struttura dei $L \in \Sigma_i$P

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

(segue)

Struttura dei problemi in PH

- **Passo induttivo (solo se)**: Ora assumiamo che $L \in \Sigma_i$P e mostriamo che R esiste
- Sia $M^?$ la MdT nondeterministica che riconosce L in tempo polinomiale usando un oracolo $K \in \Sigma_{i-1}$P
  - per ipotesi di induzione, esiste una relazione S polynomially balanced che caratterizza K, decidibile in Π_{i-2}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

- **Passo induttivo (solo se)**: Ora assumiamo che $L \in \Sigma_i$P e mostriamo che R esiste
- Sia $M^?$ la MdT nondeterministica che riconosce L in tempo polinomiale usando un oracolo $K \in \Sigma_{i-1}$P
  - per ipotesi di induzione, esiste una relazione S polynomially balanced che caratterizza K, decidibile in Π_{i-2}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

- **Passo induttivo (solo se)**: Ora assumiamo che $L \in \Sigma_i\mathrm{P}$ e mostriamo che R esiste
- Sia $M^?$ la MdT nondeterministica che riconosce L in tempo polinomiale usando un oracolo $K \in \Sigma_{i-1}\mathrm{P}$
  - per ipotesi di induzione, esiste una relazione S polynomially balanced che caratterizza K, decidibile in $\Pi_{i-2}\mathrm{P}$
- 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

- **Passo induttivo (solo se)**: Ora assumiamo che $L \in \Sigma_i$P e mostriamo che R esiste
- Sia $M^?$ la MdT nondeterministica che riconosce L in tempo polinomiale usando un oracolo $K \in \Sigma_{i-1}$P
  - per ipotesi di induzione, esiste una relazione S polynomially balanced che caratterizza K, decidibile in Π_{i-2}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 $L \in \Sigma_i$P

- (segue) Resta da dimostrare che $\{x; y \mid (x, y) \in R\}$ è in Π_{i-1}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 Π_{i-1}P
  - per le query "no", decidere se $z_i \notin K$ è in Π_{i-1}P, quindi si può programmare la MdT per verificarlo rimanendo in Π_{i-1}P

QED

Struttura dei problemi in PH

Struttura dei $L \in \Sigma_i$P

- (segue) Resta da dimostrare che $\{x; y \mid (x, y) \in R\}$ è in Π_{i-1}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 Π_{i-1}P
  - per le query "no", decidere se $z_i \notin K$ è in Π_{i-1}P, quindi si può programmare la MdT per verificarlo rimanendo in Π_{i-1}P

QED

Struttura dei problemi in PH

Struttura dei $L \in \Sigma_i$P

- (segue) Resta da dimostrare che $\{x; y \mid (x, y) \in R\}$ è in Π_{i-1}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 Π_{i-1}P
  - per le query "no", decidere se $z_i \notin K$ è in Π_{i-1}P, quindi si può programmare la MdT per verificarlo rimanendo in Π_{i-1}P

QED

Struttura dei problemi in PH

Corollario

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

- $\{x; y \mid (x, y) \in R\}$ è in $\Sigma_{i-1}\mathrm{P}$
- $L = \{x \mid \text{per ogni } y \text{ t.c. } |y| \leq |x|^k,\ (x, y) \in R\}$

- Si dimostra sfruttando $\Pi_i\mathrm{P} = \mathrm{co}\Sigma_i\mathrm{P}$
- R è il complemento della relazione polynomially balanced $\bar{R}$ che esiste per $\bar{L} \in \Sigma_i\mathrm{P}$

Struttura dei problemi in PH

Rimuovere la ricorsione

Definizione

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

Corollario

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

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

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

- Prova: applicare ricorsivamente i risultati precedenti per sostituire i Σ_iP e Π_iP con i loro certificati i-ari

Struttura di PH

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

Struttura di PH

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

Struttura di PH

Teorema 17.9

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

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

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

(segue)

Struttura di PH

Teorema 17.9

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

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

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

(segue)

Struttura di PH

Collasso di PH a livello i

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

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

- Poichè $\Sigma_i\mathrm{P} = \Pi_i\mathrm{P}$, R è anche in $\Sigma_i\mathrm{P}$. Quindi per 17.8 esiste S in $\Pi_{i-1}\mathrm{P}$ 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 $\Pi_{i-1}\mathrm{P}$
- Inoltre $|y| \le |x|^{k_1}$ e $|z| \le |x; y|^{k_2}$, quindi $|y; z| = O(|x|^{k_1 k_2})$
- Quindi S' (polynomially balanced e in $\Pi_{i-1}\mathrm{P}$) prova che $L \in \Sigma_i\mathrm{P}$

QED

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

Struttura di PH

Collasso di PH a livello i

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

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

- Poichè $\Sigma_i\mathrm{P} = \Pi_i\mathrm{P}$, R è anche in $\Sigma_i\mathrm{P}$. Quindi per 17.8 esiste S in $\Pi_{i-1}\mathrm{P}$ 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 $\Pi_{i-1}\mathrm{P}$
- Inoltre $|y| \le |x|^{k_1}$ e $|z| \le |x; y|^{k_2}$, quindi $|y; z| = O(|x|^{k_1 k_2})$
- Quindi S' (polynomially balanced e in $\Pi_{i-1}\mathrm{P}$) prova che $L \in \Sigma_i\mathrm{P}$

QED

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

Struttura di PH

Collasso di PH a livello i

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

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

- Poichè $\Sigma_i\mathrm{P} = \Pi_i\mathrm{P}$, R è anche in $\Sigma_i\mathrm{P}$. Quindi per 17.8 esiste S in $\Pi_{i-1}\mathrm{P}$ 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 $\Pi_{i-1}\mathrm{P}$
- Inoltre $|y| \le |x|^{k_1}$ e $|z| \le |x; y|^{k_2}$, quindi $|y; z| = O(|x|^{k_1 k_2})$
- Quindi S' (polynomially balanced e in $\Pi_{i-1}\mathrm{P}$) prova che $L \in \Sigma_i\mathrm{P}$

QED

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

Struttura di PH

Collasso di PH a livello i

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

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

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

QED

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

Struttura di PH

Collasso di PH a livello i

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

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

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

QED

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

Struttura di PH

Collasso di PH a livello i

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

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

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

QED

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

Struttura di PH

Collasso di PH a livello i

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

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

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

QED

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

Terminologia e corollari

- Quando per ogni $j > i$ abbiamo $\Sigma_j\mathrm{P} = \Pi_j\mathrm{P} = \Delta_j\mathrm{P} = \Sigma_i\mathrm{P}$ diciamo

  PH collassa all'i-esimo livello

Corollari

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

Problemi completi per PH e i suoi livelli

STRATEGIC COMPANIES è Σ_2P-completo

Definizione di STRATEGIC COMPANIES

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

- un insieme $G(c) \subseteq G$ di merci prodotte da c
- k insiemi $O_1(c)...O_k(c) \subseteq C$ 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

STRATEGIC COMPANIES è Σ_2P-completo

Definizione di STRATEGIC COMPANIES

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

- un insieme $G(c) \subseteq G$ di merci prodotte da c
- k insiemi $O_1(c)...O_k(c) \subseteq C$ 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 nonmonotòne 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 nonmonotòne comprendono
  - Circumscription
  - Default logic
  - Answer Set Programming (ASP)
- Il ragionamento nelle versioni proposizionali di queste logiche si colloca al 2^o 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 $dead_2$?
  - SI, è una *conseguenza logica* degli assiomi qui sopra

$$\dfrac{\dfrac{load_0 \quad load_0 \rightarrow loaded_1}{loaded_1} \qquad shoot_1 \qquad loaded_1 \wedge shoot_1 \rightarrow dead_2}{dead_2}$$

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 $dead_2$?
  - SI, è una *conseguenza logica* degli assiomi qui sopra

$$\dfrac{\dfrac{load_0 \quad load_0 \rightarrow loaded_1}{loaded_1} \qquad shoot_1 \qquad loaded_1 \wedge shoot_1 \rightarrow dead_2}{dead_2}$$

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 $dead_2$?
  - SI, è una *conseguenza logica* degli assiomi qui sopra

$$\dfrac{\dfrac{load_0 \quad load_0 \rightarrow loaded_1}{loaded_1} \qquad shoot_1 \qquad loaded_1 \wedge shoot_1 \rightarrow dead_2}{dead_2}$$

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 $dead_2$?
  - SI, è una *conseguenza logica* degli assiomi qui sopra

$$\dfrac{\dfrac{load_0 \quad load_0 \rightarrow loaded_1}{loaded_1} \qquad shoot_1 \qquad loaded_1 \wedge shoot_1 \rightarrow dead_2}{dead_2}$$

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 $dead_2$?
  - SI, è una *conseguenza logica* degli assiomi qui sopra

$$\dfrac{\dfrac{load_0 \quad load_0 \rightarrow loaded_1}{loaded_1} \qquad shoot_1 \qquad loaded_1 \wedge shoot_1 \rightarrow dead_2}{dead_2}$$

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 $dead_2$?
  - SI, è una *conseguenza logica* degli assiomi qui sopra

$$\dfrac{\dfrac{load_0 \quad load_0 \rightarrow loaded_1}{loaded_1} \qquad shoot_1 \qquad loaded_1 \wedge shoot_1 \rightarrow dead_2}{dead_2}$$

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 $dead_2$?
  - SI, è una *conseguenza logica* degli assiomi qui sopra

$$\dfrac{\dfrac{load_0 \quad load_0 \rightarrow loaded_1}{loaded_1} \qquad shoot_1 \qquad loaded_1 \wedge shoot_1 \rightarrow dead_2}{dead_2}$$

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 2

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

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 $dead_3$?
  - SI, 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 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 $dead_3$?
  - SI, 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 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 $dead_3$?
  - SI, 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 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 $dead_3$?
  - SI, 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

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

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 nonmonotòne

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 nonmonotòne

Lo Yale Shooting Problem è nonmonotòno

- 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 monotòna: l'insieme delle conseguenze <u>non</u> aumenta al crescere degli assiomi
- Limite della logica classica: è monotòna
  - 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

Lo Yale Shooting Problem è nonmonotòno

- Sia KB l'insieme degli assiomi dello YSP 2ª 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 monotòna: l'insieme delle conseguenze non aumenta al crescere degli assiomi
- Limite della logica classica: è monotòna
  - 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

Lo Yale Shooting Problem è nonmonotòno

- 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 monotòna: l'insieme delle conseguenze non aumenta al crescere degli assiomi
- Limite della logica classica: è monotòna
  - 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

Lo Yale Shooting Problem è nonmonotòno

- 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 monotòna: l'insieme delle conseguenze non aumenta al crescere degli assiomi
- Limite della logica classica: è monotòna
  - 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 anormalità
  - con *abnormality predicates*
- Nel ragionamento considera solo truth assignment che sono *massimamente normali*
  - questo rende il ragionamento nonmonotòno
- 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 $p \in M$
  - $T'(p) = \mathit{false} \Rightarrow T(p) = \mathit{false}$
  - (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

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 $p \in M$
  - $T'(p) = \mathit{false} \Rightarrow T(p) = \mathit{false}$
  - (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

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 $p \in M$
  - $T'(p) = \mathit{false} \Rightarrow T(p) = \mathit{false}$
  - (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

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 $p \in M$
  - $T'(p) = \mathit{false} \Rightarrow T(p) = \mathit{false}$
  - (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 $dead_3$ ($KB \models_M dead_3$)

$$\cfrac{\cfrac{\cfrac{load_0 \quad (1)}{loaded_1} \quad \text{assume } \neg ab_1 \quad loaded_1 \wedge \neg ab_1 \rightarrow loaded_2}{loaded_2} \quad shoot_2 \quad (3)}{dead_3}$$

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 $dead_3$ ($KB \models_M dead_3$)

$$\dfrac{\dfrac{\dfrac{load_0 \quad (1)}{loaded_1} \quad \text{assume } \neg ab_1 \quad loaded_1 \wedge \neg ab_1 \rightarrow loaded_2}{loaded_2} \quad shoot_2 \quad (3)}{dead_3}$$

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 $dead_3$ ($KB \models_M dead_3$)

$$\dfrac{\dfrac{\dfrac{load_0 \quad (1)}{loaded_1} \quad \text{assume } \neg ab_1 \quad loaded_1 \wedge \neg ab_1 \rightarrow loaded_2}{loaded_2} \quad shoot_2 \quad (3)}{dead_3}$$

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 $dead_3$

$$\dfrac{\dfrac{\dfrac{load_0 \quad (1)}{loaded_1} \quad \text{assume } \neg ab_1 \quad loaded_1 \wedge \neg ab_1 \rightarrow loaded_2}{loaded_2} \qquad \dfrac{unload_1 \quad (2)}{\neg loaded_2}}{loaded_2 \wedge \neg loaded_2}$$

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 $dead_3$

$$\dfrac{\dfrac{\dfrac{load_0 \quad (1)}{loaded_1} \quad \text{assume } \neg ab_1 \quad loaded_1 \wedge \neg ab_1 \rightarrow loaded_2}{loaded_2} \qquad \dfrac{unload_1 \quad (2)}{\neg loaded_2}}{loaded_2 \wedge \neg loaded_2}$$

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 $t = 1$)

$$\dfrac{\dfrac{\dfrac{load_0 \quad (1)}{loaded_1} \quad \text{assume } \neg ab_1 \quad loaded_1 \wedge \neg ab_1 \rightarrow loaded_2}{loaded_2} \quad \dfrac{unload_1 \quad (2)}{\neg loaded_2}}{loaded_2 \wedge \neg loaded_2}$$

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 $t = 1$)

$$\dfrac{\dfrac{\dfrac{load_0 \quad (1)}{loaded_1} \quad \text{assume } \neg ab_1 \quad loaded_1 \wedge \neg ab_1 \rightarrow loaded_2}{loaded_2} \qquad \dfrac{unload_1 \quad (2)}{\neg loaded_2}}{loaded_2 \wedge \neg loaded_2}$$

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 $t = 1$)

$$\dfrac{\dfrac{\dfrac{load_0 \quad (1)}{loaded_1} \quad \text{assume } \neg ab_1 \quad loaded_1 \wedge \neg ab_1 \rightarrow loaded_2}{loaded_2} \qquad \dfrac{unload_1 \quad (2)}{\neg loaded_2}}{loaded_2 \wedge \neg loaded_2}$$

Complessità intrinseca della Circumscription

Teorema (CIRC SAT e CIRC VALIDITY)

Decidere se un modello di KB soddisfa ϕ è Σ_2P-completo.
Decidere se $KB \models_M \phi$ è Π_2P-completo.

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

Complessità intrinseca della Circumscription

Teorema (CIRC SAT e CIRC VALIDITY)

Decidere se un modello di KB soddisfa ϕ è Σ_2P-completo.
Decidere se $KB \models_M \phi$ è Π_2P-completo.

- Notare che CIRC SAT $\approx$ SAT $+$ "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)

Una fonte di problemi completi per tutta la PH

- Le QBF sono il frammento proposizionale della *logica del* 2^o *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)

Una fonte di problemi completi per tutta la PH

- Le QBF sono il frammento proposizionale della *logica del* 2^o *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)

Una fonte di problemi completi per tutta la PH

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

Quantified Boolean Formulae (QBF)

Una fonte di problemi completi per tutta la PH

- Le QBF sono il frammento proposizionale della *logica del* 2^o *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)

Una fonte di problemi completi per tutta la PH

- Le QBF sono il frammento proposizionale della *logica del 2^o 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)

Una fonte di problemi completi per tutta la PH

- Le QBF sono il frammento proposizionale della *logica del 2º 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^o livello di PH), le QBF possono codificare anche la Circumscription (2^o livello)
- Data una formula $\phi(\vec{p}, \vec{ab})$ con predicati $p_1, \ldots, p_n, ab_1, \ldots, ab_n$ 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

- Oltre a SAT e VALIDITY (1^o livello di PH), le QBF possono codificare anche la Circumscription (2^o livello)
- Data una formula $\phi(\vec{p}, \vec{ab})$ con predicati $p_1, \ldots, p_n, ab_1, \ldots, ab_n$ 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

- Oltre a SAT e VALIDITY (1^o livello di PH), le QBF possono codificare anche la Circumscription (2^o livello)

- Data una formula $\phi(\vec{p}, \vec{ab})$ con predicati $p_1, \ldots, p_n, ab_1, \ldots, ab_n$ 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

- Oltre a SAT e VALIDITY (1^o livello di PH), le QBF possono codificare anche la Circumscription (2^o 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

- Oltre a SAT e VALIDITY (1^o livello di PH), le QBF possono codificare anche la Circumscription (2^o 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

- Oltre a SAT e VALIDITY (1^o livello di PH), le QBF possono codificare anche la Circumscription (2^o livello)
- Data una formula $\phi(\vec{p}, \vec{ab})$ con predicati $p_1, \ldots, p_n, ab_1, \ldots, ab_n$ 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 $p_1, \ldots, p_n, ab_1, \ldots, ab_n$ possiamo codificare $\phi(\vec{p}, \vec{ab}) \models_M \psi(\vec{p}, \vec{ab})$
    - $\forall \vec{p}, \vec{a}b$, "se $\phi(\vec{p}, \vec{ab})$ è vera e $\psi(\vec{p}, \vec{ab})$ falsa allora il truth assignment rappresentato dai valori di $(\vec{p}, \vec{a}b)$ non è un modello (non è $\preceq_M$-minimo)"
    - $\forall \vec{p}, \vec{a}b.\ [\phi(\vec{p}, \vec{ab}) \land \neg\psi(\vec{p}, \vec{ab})] \rightarrow$ "esiste un truth assignment $(\vec{p}', \vec{ab}')$ più normale"
    - $\forall \vec{p}, \vec{a}b.\ [\phi(\vec{p}, \vec{ab}) \land \neg\psi(\vec{p}, \vec{ab})] \rightarrow$ $\exists \vec{p}', \vec{ab}'.\ \phi(\vec{p}', \vec{ab}') \land \bigwedge_{i=1}^{n}(ab_i' \rightarrow ab_i) \land \neg \bigwedge_{i=1}^{n}(ab_i \rightarrow ab_i')$
    - $\forall \vec{p}, \vec{a}b\ \exists \vec{p}', \vec{ab}'.\ [\phi(\vec{p}, \vec{ab}) \land \neg\psi(\vec{p}, \vec{ab})] \rightarrow$ $\phi(\vec{p}', \vec{ab}') \land \bigwedge_{i=1}^{n}(ab_i' \rightarrow ab_i) \land \neg \bigwedge_{i=1}^{n}(ab_i \rightarrow ab_i')$

Quantified Boolean Formulae (QBF)

QBF vs. Circumscription

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

Quantified Boolean Formulae (QBF)

QBF vs. Circumscription

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

Quantified Boolean Formulae (QBF)

QBF vs. Circumscription

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

Quantified Boolean Formulae (QBF)

QBF vs. Circumscription

- Data una formula $\phi(\vec{p}, \vec{ab})$ con predicati $p_1, \ldots, p_n, ab_1, \ldots, ab_n$ 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 $\preceq_M$-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 Σ_2P (2^o livello di PH)
  - Le QBF della forma $\forall\vec{x}\,\exists\vec{y}.\,\phi$ sono hard per Π_2P (2^o livello di PH)
- Confrontare con quelle hard per il 1^o livello:
  - $\exists\vec{p}.\phi$ (Σ_1P) e $\forall\vec{p}.\phi$ (Π_1P)
- Il numero di alternanze tra i quantificatori corrisponde ai livelli. Questo NON è un caso

QSAT_i is Σ_iP-complete

Definizione di QSAT_i

Data una QBF con i alternanze di quantificatori

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

rappresentata più concisamente come

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

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

QSAT_i is Σ_iP-complete

Teorema 17.10

QSAT_i è Σ_iP-completo.

- Quindi decidere la verità delle formule

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

è completo per Π_iP, perchè è il complemento di QSAT_i

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

QSAT_i is Σ_iP-complete

Teorema 17.10

QSAT_i è Σ_iP-completo.

- Quindi decidere la verità delle formule

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

è completo per Π_iP, perchè è il complemento di QSAT_i

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

QSAT_i is Σ_iP-complete

Teorema 17.10

QSAT_i è Σ_iP-completo.

- Quindi decidere la verità delle formule

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

è completo per Π_iP, perchè è il complemento di QSAT_i

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

QSAT_i is Σ_iP-complete

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

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

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

- Scegliamo come y_j un assegnamento di verità alle variabili $\vec{p}_j$
  - $y_1, \ldots, y_i$ è un assegnamento di verità per tutte le variabili di ϕ
  - definiamo $(x, y_1, \ldots, y_i) \in R$ sse $y_1, \ldots, y_i$ soddisfa ϕ (si può verificare in tempo polinomiale)

- Chiaramente x è vera sse $\exists y_1 \forall y_2 \exists y_3 \ldots Q y_i$ tali che $(x, y_1, \ldots, y_i) \in R$
  - quindi per il Corollario QSAT_i è in Σ_iP

QSAT_i is Σ_iP-complete

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

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

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

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

QSAT_i is Σ_iP-complete

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

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

per qualche R polynomially balanced e decidibile in tempo polinomiale

- Consideriamo la MdT M_R che decide R

(segue)

QSAT_i is Σ_iP-complete

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

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

  per qualche R polynomially balanced e decidibile in tempo polinomiale
- Consideriamo la MdT M_R che decide R

(segue)

QSAT_i is Σ_iP-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 $P^i_{s,1}$ descrivono il nastro di input
  - per $s = 1, .., |x|$ 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 $(s = |x| + 2, ..., |x| + 1 + |y_1|)$
  - 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 ($P^i_{s,t}$ con $t > 1$, Q^i_t, $S_{s,t}$)

(segue)

QSAT_i is Σ_iP-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 $P^i_{s,1}$ descrivono il nastro di input
  - per $s = 1, .., |x|$ 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 $(s = |x| + 2, \ldots, |x| + 1 + |y_1|)$
  - 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 ($P^i_{s,t}$ con $t > 1$, Q^i_t, $S_{s,t}$)

(segue)

QSAT_i is Σ_iP-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 $P^i_{s,1}$ descrivono il nastro di input
  - per $s = 1, .., |x|$ 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 $(s = |x| + 2, ..., |x| + 1 + |y_1|)$
  - 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 ($P^i_{s,t}$ con $t > 1$, Q^i_t, $S_{s,t}$)

(segue)

QSAT_i is Σ_iP-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 $P^i_{s,1}$ descrivono il nastro di input
  - per $s = 1, .., |x|$ 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 $(s = |x| + 2, ..., |x| + 1 + |y_1|)$
  - 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 ($P^i_{s,t}$ con $t > 1$, Q^i_t, $S_{s,t}$)

(segue)

QSAT_i is Σ_iP-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 $P^i_{s,1}$ descrivono il nastro di input
  - per $s = 1, .., |x|$ 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 $(s = |x| + 2, ..., |x| + 1 + |y_1|)$
  - 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 ($P^i_{s,t}$ con $t > 1$, Q^i_t, $S_{s,t}$)

(segue)

QSAT_i is Σ_iP-complete

Prova di hardness (segue)

- Per decidere se $x \in L$ dobbiamo verificare se M_R accetta $(x, y_1, \ldots, y_i)$ (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 $y_1, \ldots, y_i$ così determinato deve esistere un assegnamento alle variabili in V che soddisfa ϕ
- Questo equivale a verificare se è vera la QBF

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

che è una istanza di QSAT_i

(segue)

QSAT_i is Σ_iP-complete

Prova di hardness (segue)

- Per decidere se $x \in L$ dobbiamo verificare se M_R accetta $(x, y_1, ..., y_i)$ (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 $y_1, ..., y_i$ così determinato deve esistere un assegnamento alle variabili in V che soddisfa ϕ
- Questo equivale a verificare se è vera la QBF

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

che è una istanza di QSAT_i

(segue)

QSAT_i is Σ_iP-complete

Prova di hardness (segue)

- Per decidere se $x \in L$ dobbiamo verificare se M_R accetta $(x, y_1, ..., y_i)$ (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 $y_1, ..., y_i$ così determinato deve esistere un assegnamento alle variabili in V che soddisfa ϕ
- Questo equivale a verificare se è vera la QBF

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

che è una istanza di QSAT_i

(segue)

QSAT_i is Σ_iP-complete

Prova di hardness (segue)

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

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

QED

QSAT_i is Σ_iP-complete

Prova di hardness (segue)

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

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

QED

QSAT_i is Σ_iP-complete

Prova di hardness (segue)

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

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

QED

Esercizi

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

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

Considerare entrambe le direzioni

Risposte possibili: “si”, “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 $\overline{\text{QSAT}}_{i+1}$
- da QSAT_i a $\overline{\text{QSAT}}_i$
- da $\overline{\text{QSAT}}_i$ a QSAT_{i+1}
- da QSAT_i a TSP(D)
- da QSAT_i a VALIDITY

CIRC SAT è Σ_2P-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 Σ_2P-completo

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

$$x_i \vee x_i' \qquad y_j \vee y_j' \qquad \phi \rightarrow y_j \wedge y_j'$$

dove le x_i' e le y_i' 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 Σ_2P-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 Σ_2P-completo

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

$$x_i \vee x_i' \qquad y_j \vee y_j' \qquad \phi \rightarrow y_j \wedge y_j'$$

dove le x_i' e le y_i' sono nuove proposizioni. La query è ϕ
- Mettiamo in *M* (proposizioni da "minimizzare") <u>tutte</u> le proposizioni
    - i modelli massimamente normali di KB saranno quelli in cui quante più proposizioni possibili sono false (segue)

CIRC SAT è Σ_2P-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 Σ_2P-completo

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

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

  dove le x'_i e le y'_i 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 Σ_2P-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 $T'(x_i') = false$ 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 Σ_2P-completo

Hardness: riduzione da QSAT$_2$ – segue

- Ora mostriamo che Φ è vera sse ϕ 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 $y_1, \ldots, y_m$ rende vera ϕ (*)
  - Estendiamo T_{xy} a un T per tutte le proposizioni di *KB* ponendo
    - $T(x_i') = \neg T(x_i)$, $T(y_i') = T(y_i) = true$

  Verificare che $T \models KB$

(segue)

CIRC SAT è Σ_2P-completo

Hardness: riduzione da QSAT$_2$ – segue

- Ora mostriamo che Φ è vera sse ϕ 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 $y_1, \ldots, y_m$ rende vera ϕ (*)
  - Estendiamo T_{xy} a un T per tutte le proposizioni di KB ponendo
    - $T(x_i') = \neg T(x_i)$, $T(y_i') = T(y_i) = true$

    Verificare che $T \models KB$

(segue)

CIRC SAT è Σ_2P-completo

Hardness: riduzione da QSAT$_2$ – 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 Σ_2P-completo

Hardness: riduzione da QSAT$_2$ – 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 Σ_2P-completo

Hardness: riduzione da QSAT_2 – 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 Σ_2P-completo

Hardness: riduzione da QSAT_2 – 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 Σ_2P-completo

Hardness: riduzione da QSAT$_2$ – 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 Σ_2P-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 $T'(x_i') = \neg T_{xy}(x_i)$, $T'(y_i') = \neg T_{xy}(y_i)$
- Verificare che $T' \prec_M T$ (assurdo, perchè T è $\preceq_M$-minimo!)

(fine della prova di hardness)

CIRC SAT è Σ_2P-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 $T'(x_i') = \neg T_{xy}(x_i)$, $T'(y_i') = \neg T_{xy}(y_i)$
- Verificare che $T' \prec_M T$ (assurdo, perchè T è $\preceq_M$-minimo!)

(fine della prova di hardness)

CIRC SAT è Σ_2P-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 $T'(x_i') = \neg T_{xy}(x_i)$, $T'(y_i') = \neg T_{xy}(y_i)$
- Verificare che $T' \prec_M T$ (assurdo, perchè T è $\preceq_M$-minimo!)

(fine della prova di hardness)

CIRC SAT è Σ_2P-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 $T'(x_i') = \neg T_{xy}(x_i)$, $T'(y_i') = \neg T_{xy}(y_i)$
- Verificare che $T' \prec_M T$ (assurdo, perchè T è $\preceq_M$-minimo!)

(fine della prova di hardness)

CIRC SAT è Σ_2P-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 $T'(x_i') = \neg T_{xy}(x_i)$, $T'(y_i') = \neg T_{xy}(y_i)$
- Verificare che $T' \prec_M T$ (assurdo, perchè T è $\preceq_M$-minimo!)

(fine della prova di hardness)

CIRC SAT è Σ_2P-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

CIRC SAT è Σ_2P-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

CIRC SAT è Σ_2P-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 Σ_2P-completo

Prova di hardness – riduzione da QSAT$_2$

- Dobbiamo ridurre una qualunque QBF $\Phi = \exists x_1 \ldots \exists x_n \forall y_1 \ldots \forall y_m . \phi$ 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 Σ_2P-hard
- Riduzione: $C = \{x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n, y_1, \ldots, y_m, \neg y_1, \ldots, \neg y_m, w\}$,
    - $G = \{gx_1, \ldots, gx_n, gy_1, \ldots, gy_m\}$
    - $G(x_i) = G(\neg x_i) = \{gx_i\}, \quad G(y_i) = G(\neg y_i) = \{gy_i\}$,
    - $O(x_i) = \{\{x_i\}\}, \quad O(\neg x_i) = \{\{\neg x_i\}\}$,
      $O(y_i) = O(\neg y_i) = \{\{w\}\}$,
      $O(w) = \{\{L_{i1}, L_{i2}, L_{i3}\} \mid i = 1 \ldots k\}$

(segue)

STRATEGIC COMPANIES è Σ_2P-completo

Prova di hardness – riduzione da QSAT$_2$

- Dobbiamo ridurre una qualunque QBF $\Phi = \exists x_1 \ldots \exists x_n \forall y_1 \ldots \forall y_m . \phi$ 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 Σ_2P-hard
- Riduzione: $C = \{x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n, y_1, \ldots, y_m, \neg y_1, \ldots, \neg y_m, w\}$,
  - $G = \{gx_1, \ldots, gx_n, gy_1, \ldots, gy_m\}$
  - $G(x_i) = G(\neg x_i) = \{gx_i\}, \quad G(y_i) = G(\neg y_i) = \{gy_i\}$,
  - $O(x_i) = \{\{x_i\}\}, \quad O(\neg x_i) = \{\{\neg x_i\}\}$,
    $O(y_i) = O(\neg y_i) = \{\{w\}\}$,
    $O(w) = \{\{L_{i1}, L_{i2}, L_{i3}\} \mid i = 1 \ldots k\}$

(segue)

STRATEGIC COMPANIES è Σ_2P-completo

Prova di hardness – riduzione da QSAT$_2$

- Dobbiamo ridurre una qualunque QBF $\Phi = \exists x_1 \ldots \exists x_n \forall y_1 \ldots \forall y_m . \phi$ 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 Σ_2P-hard
- Riduzione: $C = \{x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n, y_1, \ldots, y_m, \neg y_1, \ldots, \neg y_m, w\}$,
  - $G = \{gx_1, \ldots, gx_n, gy_1, \ldots, gy_m\}$
  - $G(x_i) = G(\neg x_i) = \{gx_i\}, \quad G(y_i) = G(\neg y_i) = \{gy_i\}$,
  - $O(x_i) = \{\{x_i\}\}, \quad O(\neg x_i) = \{\{\neg x_i\}\}$,
    $O(y_i) = O(\neg y_i) = \{\{w\}\}$,
    $O(w) = \{\{L_{i1}, L_{i2}, L_{i3}\} \mid i = 1 \ldots k\}$

(segue)

STRATEGIC COMPANIES è Σ_2P-completo

Prova di hardness – riduzione da QSAT$_2$

- Dobbiamo ridurre una qualunque QBF $\Phi = \exists x_1 \dots \exists x_n \forall y_1 \dots \forall y_m . \phi$ 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 Σ_2P-hard
- Riduzione: $C = \{x_1, \dots, x_n, \neg x_1, \dots, \neg x_n, y_1, \dots, y_m, \neg y_1, \dots, \neg y_m, w\}$,
  - $G = \{gx_1, \dots, gx_n, gy_1, \dots, gy_m\}$
  - $G(x_i) = G(\neg x_i) = \{gx_i\}, \quad G(y_i) = G(\neg y_i) = \{gy_i\}$,
  - $O(x_i) = \{\{x_i\}\}, \quad O(\neg x_i) = \{\{\neg x_i\}\}$,
    $O(y_i) = O(\neg y_i) = \{\{w\}\}$,
    $O(w) = \{\{L_{i1}, L_{i2}, L_{i3}\} \mid i = 1 \dots k\}$

(segue)

STRATEGIC COMPANIES è Σ_2P-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 $x_1, \ldots, x_n$ che rende vera Φ, sia $C' \subseteq C$ il seguente insieme:
  - contiene x_i sse $T(x_i) = \mathit{true}$ e $\neg x_i$ sse $T(x_i) = \mathit{false}$
  - contiene $y_1, \ldots, y_m, \neg y_1, \ldots, \neg y_m, w$
- 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 Σ_2P-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 $x_1, \ldots, x_n$ che rende vera Φ, sia $C' \subseteq C$ il seguente insieme:
  - contiene x_i sse $T(x_i) = \mathit{true}$ e $\neg x_i$ sse $T(x_i) = \mathit{false}$
  - contiene $y_1, \ldots, y_m, \neg y_1, \ldots, \neg y_m, w$
- 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 Σ_2P-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 $x_1, \ldots, x_n$ che rende vera Φ, sia $C' \subseteq C$ il seguente insieme:
  - contiene x_i sse $T(x_i) = \mathit{true}$ e $\neg x_i$ sse $T(x_i) = \mathit{false}$
  - contiene $y_1, \ldots, y_m, \neg y_1, \ldots, \neg y_m, w$
- 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 Σ_2P-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 $x_1, \ldots, x_n$ che rende vera Φ, sia $C' \subseteq C$ il seguente insieme:
  - contiene x_i sse $T(x_i) = \mathit{true}$ e $\neg x_i$ sse $T(x_i) = \mathit{false}$
  - contiene $y_1, \ldots, y_m, \neg y_1, \ldots, \neg y_m, w$
- 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 Σ_2P-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 gx_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, sennò perdiamo un bene
- questi dipendono tutti da w quindi – per rispettare la proprietà di chiusura – deve essere $w \notin C''$

(segue)

STRATEGIC COMPANIES è Σ_2P-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 gx_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, sennò perdiamo un bene
- questi dipendono tutti da w quindi – per rispettare la proprietà di chiusura – deve essere $w \notin C''$

(segue)

STRATEGIC COMPANIES è Σ_2P-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 gx_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, sennò perdiamo un bene
- questi dipendono tutti da w quindi – per rispettare la proprietà di chiusura – deve essere $w \notin C''$

(segue)

STRATEGIC COMPANIES è Σ_2P-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 gx_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, sennò perdiamo un bene
- questi dipendono tutti da w quindi – per rispettare la proprietà di chiusura – deve essere $w \notin C''$

(segue)

STRATEGIC COMPANIES è Σ_2P-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 $T(y_i) = \mathit{true}$ allora $y_i \in C''$
  - se $T(y_i) = \mathit{false}$ 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 Σ_2P-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 $T(y_i) = \mathit{true}$ allora $y_i \in C''$
  - se $T(y_i) = \mathit{false}$ 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 Σ_2P-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 $T(y_i) = \mathit{true}$ allora $y_i \in C''$
  - se $T(y_i) = \mathit{false}$ 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 Σ_2P-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 $T(y_i) = \mathit{true}$ allora $y_i \in C''$
  - se $T(y_i) = \mathit{false}$ 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 Σ_2P-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 $T(x_i) = true$ sse $x_i \in C'$ e supponiamo che una sua estensione a $y_1, \ldots, y_m$ 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 Σ_2P-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 $T(x_i) = \mathit{true}$ sse $x_i \in C'$ e supponiamo che una sua estensione a $y_1, \ldots, y_m$ 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 Σ_2P-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 $T(x_i) = \mathit{true}$ sse $x_i \in C'$ e supponiamo che una sua estensione a $y_1, \ldots, y_m$ 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 Σ_2P-completo

Prova di hardness – riduzione da QSAT$_2$ – segue

- (Correttezza – segue) Definiamo
  - $C'' = \{v_i \mid T(v_i) = \mathit{true}\} \cup \{\neg v_i \mid T(v_i) = \mathit{false}\}$ $(w \notin C'')$
- Verificare che T è "contenuto" in C'' nel senso di prima:
  - se $T(v_i) = \mathit{true}$ allora $y_i \in C''$
  - se $T(v_i) = \mathit{false}$ 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 Σ_2P-completo

Prova di hardness – riduzione da QSAT$_2$ – segue

- (Correttezza – segue) Definiamo
  - $C'' = \{v_i \mid T(v_i) = true\} \cup \{\neg v_i \mid T(v_i) = false\}$ $(w \notin C'')$
- Verificare che T è "contenuto" in C'' nel senso di prima:
  - se $T(v_i) = true$ allora $y_i \in C''$
  - se $T(v_i) = false$ 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 Σ_2P-completo

Prova di hardness – riduzione da QSAT$_2$ – segue

- (Correttezza – segue) Definiamo
  - $C'' = \{v_i \mid T(v_i) = true\} \cup \{\neg v_i \mid T(v_i) = false\}$ $(w \notin C'')$
- Verificare che T è "contenuto" in C'' nel senso di prima:
  - se $T(v_i) = true$ allora $y_i \in C''$
  - se $T(v_i) = false$ 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 Σ_2P-completo

Prova di hardness – riduzione da QSAT$_2$ – segue

- (Correttezza – segue) Definiamo
  - $C'' = \{v_i \mid T(v_i) = \mathit{true}\} \cup \{\neg v_i \mid T(v_i) = \mathit{false}\}$ $(w \notin C'')$
- Verificare che T è "contenuto" in C'' nel senso di prima:
  - se $T(v_i) = \mathit{true}$ allora $y_i \in C''$
  - se $T(v_i) = \mathit{false}$ 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 Σ_2P-completo

Prova di hardness – riduzione da QSAT$_2$ – segue

- (Correttezza – segue) Definiamo
  - $C'' = \{v_i \mid T(v_i) = true\} \cup \{\neg v_i \mid T(v_i) = false\}$ $(w \notin C'')$
- Verificare che T è "contenuto" in C'' nel senso di prima:
  - se $T(v_i) = true$ allora $y_i \in C''$
  - se $T(v_i) = false$ 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 Σ_2P-completo

Prova di hardness – riduzione da QSAT$_2$ – segue

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

QED

STRATEGIC COMPANIES è Σ_2P-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 $\mathrm{NP}^{\mathrm{NP}} = \Sigma_2\mathrm{P}$

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?

- 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?

- 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?

- 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 $L \in \mathrm{PH}$ e $\mathrm{PH} = \bigcup_i \Sigma_i\mathrm{P}$, deve esistere $k \geq 0$ tale che

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

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

QED

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 $L \in \mathrm{PH}$ e $\mathrm{PH} = \bigcup_i \Sigma_i\mathrm{P}$, deve esistere $k \geq 0$ tale che

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

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

QED

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 $L \in \mathrm{PH}$ e $\mathrm{PH} = \bigcup_i \Sigma_i\mathrm{P}$, deve esistere $k \geq 0$ tale che

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

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

QED

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 $L \in \mathrm{PH}$ e $\mathrm{PH} = \bigcup_i \Sigma_i\mathrm{P}$, deve esistere $k \geq 0$ tale che
$$L \in \Sigma_k\mathrm{P}$$
- Ma ogni $L' \in \Sigma_{k+1}\mathrm{P}$ si riduce a L (che è PH-completo)
- Quindi (siccome tutti i livelli sono chiusi rispetto alle riduzioni) $L' \in \Sigma_k\mathrm{P}$
- Ne segue che $\Sigma_k\mathrm{P} = \Sigma_{k+1}\mathrm{P}$ 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 $L \in \Sigma_i$P

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

  per una opportuna R calcolabile in tempo polinomiale
- R è polynomially balanced, quindi $|y_1, \ldots, y_i|$ è polinomiale in x
- Facile vedere che la enumerazione dei $y_1, \ldots, y_i$ 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

Proposizione 17.1

PH $\subseteq$ PSPACE

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

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

per una opportuna R calcolabile in tempo polinomiale

- R è polynomially balanced, quindi $|y_1, ..., y_i|$ è polinomiale in x
- Facile vedere che la enumerazione dei $y_1, ..., y_i$ 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

Proposizione 17.1

PH $\subseteq$ PSPACE

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

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

per una opportuna R calcolabile in tempo polinomiale

- R è polynomially balanced, quindi $|y_1, ..., y_i|$ è polinomiale in x
- Facile vedere che la enumerazione dei $y_1, ..., y_i$ 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

Proposizione 17.1

PH $\subseteq$ PSPACE

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

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

per una opportuna R calcolabile in tempo polinomiale

- R è polynomially balanced, quindi $|y_1, ..., y_i|$ è polinomiale in x
- Facile vedere che la enumerazione dei $y_1, ..., y_i$ 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 $Q_1 p_1 ... Q_n p_n . \phi(p_1, ..., p_n)$
output il valore di verità della QBF
  if $n = 0$ restituire il valore di ϕ // non ci sono proposizioni
  $b_0 = Truth(Q_2 p_2 ... Q_n p_n, \phi(0, p_2, ..., p_n))$
  $b_1 = Truth(Q_2 p_2 ... Q_n p_n, \phi(1, p_2, ..., p_n))$ // 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

Altra versione ricorsiva (in pseudocodice) che usa spazio polinomiale

algorithm *Truth*
input una QBF $Q_1 p_1 \ldots Q_n p_n . \phi(p_1, \ldots, p_n)$
output il valore di verità della QBF
  if $n = 0$ restituire il valore di ϕ // non ci sono proposizioni
  $b_0 = \mathit{Truth}(Q_2 p_2 \ldots Q_n p_n, \phi(0, p_2, \ldots, p_n))$
  $b_1 = \mathit{Truth}(Q_2 p_2 \ldots Q_n p_n, \phi(1, p_2, \ldots, p_n))$ // 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 PH = PSPACE allora PH collassa a qualche suo livello finito

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

QED

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

PH e PSPACE

Corollario di 17.11

Se PH $=$ PSPACE allora PH collassa a qualche suo livello finito

- **Prova**: Premessa: PSPACE ha problemi completi (slide successiva)
- Quindi se PH $=$ PSPACE anche PH avrebbe problemi completi
- Di conseguenza, per il Teorema 17.11, PH collasserebbe 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

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

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

- 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

- 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 $\psi_i(A, B)$ $(i = 1, .., m)$ 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 $x \stackrel{M}{\to} y$
    - 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

- Definiamo una sequenza di QBF $\psi_i(A, B)$ ($i = 1, .., m$) 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 $x \stackrel{M}{\to} y$
  - 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

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

- 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

- 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

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

- 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

- 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

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

- 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)

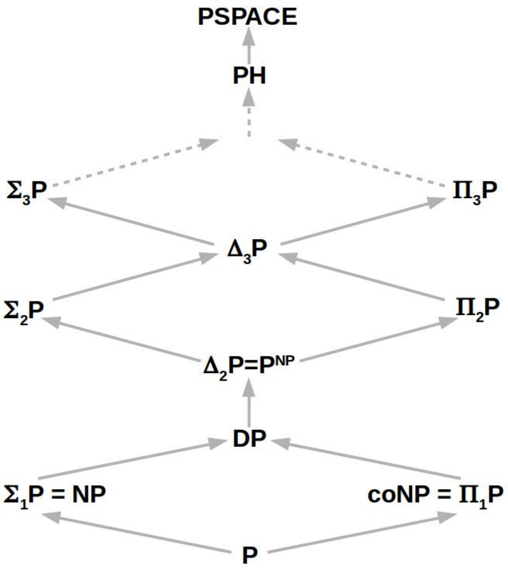
QSAT (senza limiti di alternanza) è PSPACE-completo

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


PSPACE
PH
$\Sigma_3 P$
$\Pi_3 P$
$\Delta_3 P$
$\Sigma_2 P$
$\Pi_2 P$
$\Delta_2 P = P^{NP}$
DP
$\Sigma_1 P = NP$
$coNP = \Pi_1 P$
P

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 $\overline{\mathrm{QSAT}}_i$
- CLIQUE e QSAT_1
- CLIQUE e $\mathrm{CIRC\ SAT}$
- CLIQUE e $\overline{\mathrm{QSAT}}_1$
- CLIQUE e $\overline{\mathrm{STRATEGIC\ COMPANIES}}$
- QSAT e QSAT_3
- QSAT e $\overline{\mathrm{HAMILTON\ PATH}}$

Considerare entrambe le direzioni

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

Database queries

Misure di complessità

Ci sono diversi modi di misurare la complessità delle query:

Combined complexity: Sia la query che il database sono parte dell'input

Data complexity: L'input è il solo il database (la query è fissata)

- come se si considerasse la query trascurabile rispetto alla dimensione del database

Expression complexity: L'input è la sola query (il database è fissato)

- che succede interrogando uno specifico database con query sempre più complesse

Risultati fondamentali

Il setting:

- Database relazionali
- Query language: relational algebra

Teorema: Complessità delle query booleane ai database relazionali

1. Combined complexity: PSPACE-complete
2. Expression complexity: PSPACE-complete
3. Data complexity: **L**-completa (L $\subseteq$ P !)

Answer membership: idem

[Non presenti sul libro]

Risultati fondamentali

Il setting:

- Database relazionali
- Query language: relational algebra

Teorema: Complessità delle query booleane ai database relazionali

1. Combined complexity: PSPACE-complete
2. Expression complexity: PSPACE-complete
3. Data complexity: **L**-completa ($\mathbf{L} \subseteq \mathbf{P}$!)

Answer membership: idem

[Non presenti sul libro]

Risultati fondamentali

Il setting:

- Database relazionali
- Query language: relational algebra

Teorema: Complessità delle query booleane ai database relazionali

1. Combined complexity: PSPACE-complete
2. Expression complexity: PSPACE-complete
3. Data complexity: **L**-completa ($L \subseteq P$!)

Answer membership: idem

[Non presenti sul libro]

Risultati fondamentali

Il setting:

- Database relazionali
- Query language: relational algebra

Teorema: Complessità delle query booleane ai database relazionali

1. Combined complexity: PSPACE-complete
2. Expression complexity: PSPACE-complete
3. Data complexity: **L**-completa (L $\subseteq$ P !)

Answer membership: idem

[Non presenti sul libro]

Risultati fondamentali

Il setting:

- Database relazionali
- Query language: relational algebra

Teorema: Complessità delle query booleane ai database relazionali

1. Combined complexity: PSPACE-complete
2. Expression complexity: PSPACE-complete
3. Data complexity: **L**-completa (L $\subseteq$ P !)

Answer membership: idem

[Non presenti sul libro]

Un caso particolare: Le query congiuntive

Il setting: query *congiuntive*, cioè

- senza unioni
- senza negazione

Complessità delle conjunctive queries

1. Combined complexity: NP-complete
2. Expression complexity: NP-complete
3. Data complexity: $\mathbf{AC}_0$-complete ($AC_0 \subseteq L \subseteq P$) (!!)

[Non presenti sul libro]

Alternanza e giochi

Giochi

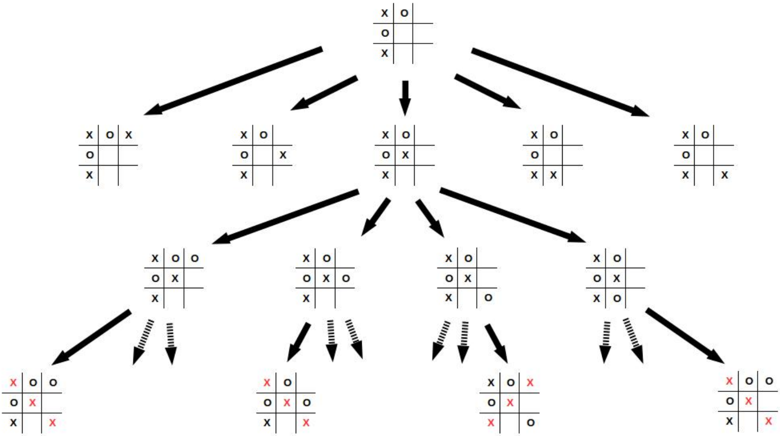
- In termini matematici un gioco è essenzialmente un grafo
  - rappresentato esplicitamente o implicitamente
  - spesso in forma di albero
- I nodi sono analoghi alle *configurazioni* di una MdT
  - rappresentano uno stato
  - ad es. pezzi su di una scacchiera
- Gli archi sono analoghi alle *transizioni*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

Giochi

- 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

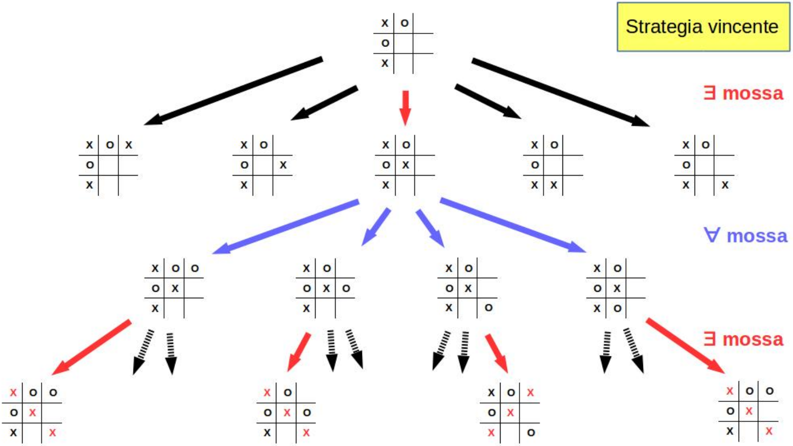
Giochi

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



Giochi

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

- 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

- 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

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

- 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

- 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

- 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^o giocatore a partire dal nodo 1?

Teorema 19.3

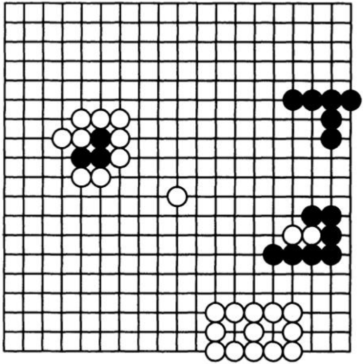
GEOGRAPHY è PSPACE completo

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

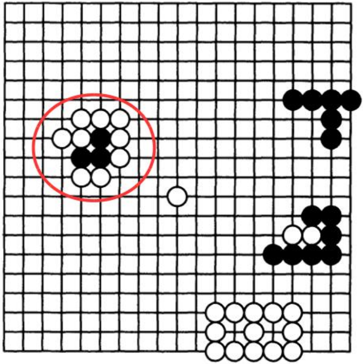
Esempio



- Esempio di configurazione

Go

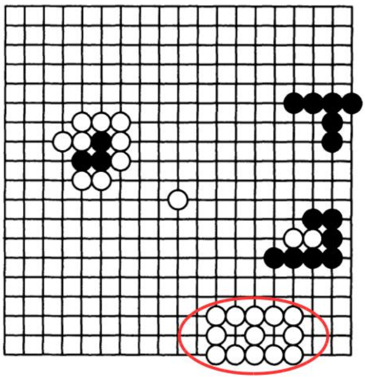
Esempio



- 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 si, dire se esiste un unico intero k che limita l'alternanza di quantificatori in Φ per tutte le istanze di GEOGRAPHY

Risposte possibili: "si", "no", "solo se $\langle$conseguenze$\rangle$", "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