Complessità computazionale

— ○ —

Oltre i limiti di PSPACE

Piero A. Bonatti

Università di Napoli Federico II

Laurea Magistrale in Informatica

Tema della lezione

- La gerarchia esponenziale e i problemi "elementari"
- Proprietà di **EXP** e **NEXP**
- Alcuni problemi di AI nella gerarchia esponenziale
- Commenti su **RE** e **coRE**

Gerarchia esponenziale e linguaggi "elementari"

- Ricordare che $\mathbf{EXP} = \mathbf{TIME}(2^{n^k})$ Similmente:

$$\begin{aligned} 2\text{-}\mathbf{EXP} &= \mathbf{TIME}(2^{2^{n^k}}) \\ 3\text{-}\mathbf{EXP} &= \mathbf{TIME}(2^{2^{2^{n^k}}}) \\ &\vdots \end{aligned}$$

- Un problema è detto elementare (sic!) se appartiene a qualche

$$\mathbf{TIME}(2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}})$$

  Equivalentemente: appartiene a qualche i-**EXP**

- Per una volta, la gerarchia *non collassa*:

$$\mathbf{EXP} \subset 2\text{-}\mathbf{EXP} \subset 3\text{-}\mathbf{EXP} \subset \cdots \subset \mathbf{ELEMENTARY}$$

Tempo esponenziale nondeterministico

Definizione

NEXP $=$ **NTIME**(2^{n^k})

- Le relazioni tra **EXP** e **NEXP** non sono note
- Provare **EXP** $\neq$ **NEXP** è difficile almeno quanto provare **P** $\neq$ **NP**. Infatti **EXP** $\neq$ **NEXP** $\Rightarrow$ **P** $\neq$ **NP**, ovvero:

Teorema 20.1 (che è equivalente)

Se **P** $=$ **NP** allora **EXP** $=$ **NEXP**

- Per quanto ne sappiamo, il viceversa non necessariamente vale: anche se **EXP** $=$ **NEXP** è possibile che **P** $\neq$ **NP**

Prova del Teorema 20.1

Se $L \in$ **NEXP** e **P** $=$ **NP** allora $L \in$ **EXP**

- Sia L un qualunque linguaggio in **NEXP**
- Per definizione, esiste una MdT nondeterministica N che decide L in tempo 2^{n^k}
- Consideriamo un L' che "gonfia" le istanze di L esponenzialmente con un nuovo simbolo $\sqcap$ che chiamiamo *quasi-blank*

$$L' = \{x\sqcap^{2^{|x|^k}-|x|} \mid x \in L\}$$

(i quasi-blank allungano ogni x a $2^{|x|^k}$ caratteri)

(segue)

Prova del Teorema 20.1 – segue

Se $L \in$ **NEXP** e **P** $=$ **NP** allora $L \in$ **EXP**

- Dimostriamo prima che $L' \in$ **NP**, con una *modifica* N' *di* N
  1. Prima controlla se l'input y ha la forma $x\sqcap^{2^{|x|^k}-|x|}$ dove x non contiene $\sqcap$ (altrimenti lo rigetta)
  2. Poi simula N trattando $\sqcap$ come fosse $\sqcup$
- N' opera in tempo polinomiale grazie al "rigonfiamento" di x
  - fase 1: polinomiale nell'input (si tratta solo di contare $|x|$ e $|y|$ e verificare che $|y| = 2^{|x|^k}$)
  - fase 2: tempo $2^{|x|^k}$ (costo di N per ipotesi) ovvero $|y|$

(segue)

Prova del Teorema 20.1 – segue

Se $L \in \mathbf{NEXP}$ e $\mathbf{P} = \mathbf{NP}$ allora $L \in \mathbf{EXP}$

- Poichè $L' \in \mathbf{NP}$ e (per ipotesi) $\mathbf{P} = \mathbf{NP}$, abbiamo $L' \in \mathbf{P}$
  - deve esistere una MdT *deterministica* M' che decide L' in *tempo polinomiale* n^ℓ
- Infine costruiamo una MdT *deterministica* M basata su M' che decide L in tempo esponenziale
  - questo proverà che $L \in \mathbf{EXP}$ completando la dimostrazione

(segue)

Prova del Teorema 20.1 – segue

Se $L \in$ **NEXP** e **P** $=$ **NP** allora $L \in$ **EXP**

- M opera così:
  1. "allunga" l'input x con $\sqcap$ fino alla lunghezza $2^{|x|^k}$
  2. esegue le istruzioni di M' sull'input allungato

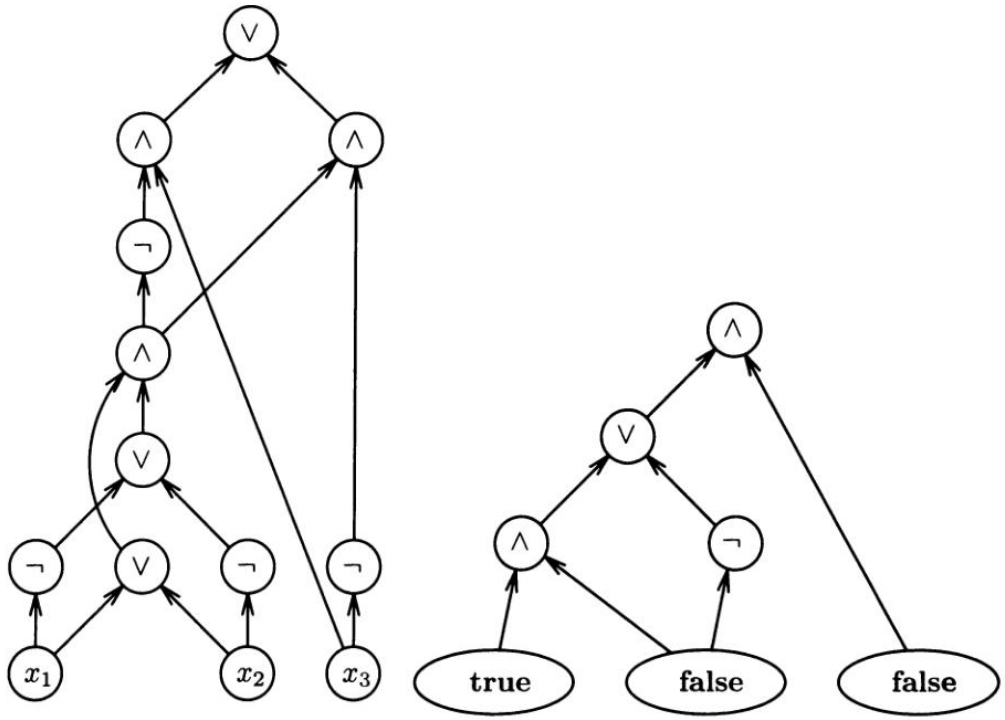
- Il tempo richiesto è proprio esponenziale
  - fase 1: $O(p(|x|) \cdot 2^{|x|^k})$ che è $O(2^{|x|^{k+j}})$ (per qualche j)
    - p è il costo polinomiale della manipolazione dei contatori che servono a calcolare e misurare $2^{|x|^k}$ (polinomiali in $|x|$)
  - fase 2: $(2^{|x|^k})^\ell = 2^{\ell|x|^k}$ che è $O(2^{|x|^{k+1}})$
- Ne concludiamo che $L \in$ **EXP**

Alcuni problemi da **EXP** in su

- Abbiamo già menzionato che SCHÖNFINKEL-BERNAYS SAT è **NEXP**-completo
- Aggiungiamo una varietà di problemi completi per **EXP** e **NEXP**
- Presentano caratteristiche simili agli L, L' del Teorema 20.1
  - versioni *esponenzialmente succinte* di problemi in **P** e **NP**

Preliminari: i circuiti

- Un circuito C non è che un grafo diretto aciclico con nodi etichettati opportunamente $(\wedge, \vee, \neg, \mathit{true}, \mathit{false}, x_1, x_2, ...)$
- l'etichetta determina quanti archi entrano/escono
- nodi di input: 0 archi entranti
- nodi di output: 0 archi uscenti
- $C(b_1, ..., b_n)$ denota l'output di C quando $x_1 = b_1, ..., x_n = b_n$



Circuiti a struttura regolare

- I circuiti VLSI hanno un gran numero di gate, ma presentano simmetrie e ripetizioni di componenti
- Queste possono essere sfruttate per descrivere concisamente il circuito

> "Repeat the (given) component C horizontally and vertically in an $N \times M$ grid $(i, j), i = 1, \ldots, N; j = 1, \ldots, M$ (where N and M are given large integers), except at the positions $i = j$, the positions $i = 2$, and the positions $i = N - 1$; at these positions place the other given component $C' \ldots$"

- Simili descrizioni algoritmiche possono essere anche *esponenzialmente* più piccole del circuito che rappresentano
  - in che linguaggio le rappresentiamo?

Circuiti a struttura regolare

- Per rappresentare concisamente il grafo di un circuito possiamo usare un altro circuito
  - che per ogni coppia di nodi dice se è un arco o no

Definizione del grafo G_C rappresentato da C

Un circuito C con $2b$ inputs *rappresenta succintamente* un grafo G_C con 2^b nodi $(1, 2, ..., 2^b)$ convenendo che:

$$(i, j) \text{ è un arco di } G_C \Leftrightarrow C(\underline{i}, \underline{j}) = true$$

(dove $\underline{i}, \underline{j}$ è la sequenza di bit che rappresenta i e j)

Problemi con codifica succinta dell'input

- Per progettare automaticamente i circuiti VLSI si devono risolvere alcuni problemi in **P** e in **NP** sul grafo del circuito
  - inclusi MAX CUT, MAX FLOW, BISECTION WIDTH ecc.
- Se il grafo del circuito è rappresentato concisamente con un altro circuito, otteniamo la *versione succinta* del problema

Problemi succinti

Dato un problema di decisione L su grafi, SUCCINCT L è

$$\{C \mid G_C \in L\}$$

- Esempio: SUCCINCT HAMILTON PATH
  - Dato un circuito C con $2n$ inputs, dire se G_C ha un ciclo hamiltoniano

SUCCINCT CIRCUIT SAT è **NEXP**-completo

- La somiglianza tra circuiti ed espressioni booleane introduce un problema naturale:
  - CIRCUIT SAT: Dato un circuito $C(x_1, ..., x_n)$, dire se esiste una stringa di bit $b_1, ..., b_n$ tale che $C(b_1, ..., b_n) = true$
- Facile vedere che CIRCUIT SAT è **NP**-completo
- SUCCINCT CIRCUIT SAT è **NEXP**-completo

Alcuni problemi con caratteristiche simili

Problema L	Completo per...	
	L	SUCCINCT L
CIRCUIT VALUE[1]	**P**	**EXP**
3-SAT	**NP**	**NEXP**
HAMILTON PATH	**NP**	**NEXP**

[1] Calcolare l'uscita del circuito quando tutti gli input sono dati (*true* o *false*)

Spazio esponenziale

- Menzioniamo solo che possiamo definire una gerarchia

  EXPSPACE, 2-EXPSPACE, 3-EXPSPACE, ...

  con caratteristiche simili
- Inclusa la separazione stretta tra le classi

I problemi di ragionamento automatico nella gerarchia esponenziale

Semantic web

- Immaginate di etichettare le risorse web con *metadati* molto ricchi...
  - espressi in un frammento decidibile della logica del primo ordine
  - ...o quasi: la sintassi utilizzata è un po' diversa
- Che un computer può comprendere ed elaborare...
- In qualche misura, i metadati "spiegano" al computer il *significato* delle risorse
  - specificando relazioni tra parole, immagini, classi di oggetti, ...
  - mediante insiemi di assiomi detti *ontologie*
- In questo modo è possibile
  - Aumentare la precisione dei search engine
  - Scoprire le relazioni implicite tra diversi documenti
  - ...e più in generale *ragionare* sulle risorse per risolvere i problemi posti dall'utente

Esempio

GoPubMed

- Search the PubMed digital library with Gene Ontology (GO)[2]
- It exploits implicit relations between terms that can be *inferred* from explicit knowledge coded in GO (∼30K axioms and ∼20K concept names)

Example of inference

$$\frac{\begin{array}{l}\mathsf{Endocardium} \sqsubseteq \exists \mathsf{isPartOf}.\mathsf{Heart} \\ \mathsf{Endocarditis} \equiv \mathsf{Disease} \sqcap \exists \mathsf{affects}.\mathsf{Endocardium} \\ \mathsf{HeartDisease} \equiv \mathsf{Disease} \sqcap \exists \mathsf{affects}.\exists \mathsf{isPartOf}.\mathsf{Heart}\end{array}}{\mathsf{Endocarditis} \sqsubseteq \mathsf{HeartDisease}}$$

passare a [SchroederW3C, 3-13]

[2] Doms A., Schroeder M.: GoPubMed: exploring PubMed with the Gene Ontology. Nucleic Acids Research 33(Web-Server-Issue): 783-786 (2005)

Gli standard OWL

- I linguaggi per esprimere i metadati ricchi e le ontologie nell'ambito del semantic web sono stati standardizzati
  - OWL, OWL2
  - sono formati basati su XML e altro per codificare i frammenti della logica utilizzati
  - detti logiche descrittive (in quanto aderiscono alla sintassi alternativa)
- Le logiche descrittive hanno nomi diversi a seconda di quali operatori supportano
  - $\mathcal{ALC}$, $\mathcal{ALE}$, $\mathcal{SHIQ}$, $\mathcal{SHIO}$, $\mathcal{SHOQ}$ ...
- La scelta della logica (del set di operatori) influisce sulla complessità delle inferenze

Complessità delle inferenze di alcune logiche descrittive

Analoghi di SAT e VALIDITY

- Per alcuni dei frammenti (*profili*) di OWL, sono in **P**
  - OWL2-EL, OWL2-QL
- Ma quasi tutti i set di operatori portano nella gerarchia esponenziale
  - $\mathcal{SHIQ}$, $\mathcal{SHIO}$, $\mathcal{SHOQ}$: **EXP**-complete
  - $\mathcal{SHOIN}$ (OWL-DL): **NEXP**-complete
  - $\mathcal{SROIQ}$ (OWL2-DL): 2-**NEXP**-complete
- Adesso mostreremo il tempo necessario a precompilare *tutte* le relazioni tra i termini definiti in alcune importanti ontologie (*classificazione*)
  - poi il ragionamento diventa una visita a un grafo...

Le ontologie considerate

Ontology Name	Number of Axioms						
	Classes	Roles	Individuals	TBox	RBox	ABox	Expressivity
Molecule Role	8849	2	128056	9243	1	128056	$\mathcal{ALE}+$
XP Uber Anatomy	11427	82	88955	14669	80	88955	$\mathcal{ALEHIF}+$
XP Plant Anatomy	19145	82	86099	35770	87	86099	$\mathcal{SHIF}$
XP Regulators	25520	4	155169	42896	3	155169	$\mathcal{SH}$
Cellular Component	27889	4	163244	47345	3	163244	$\mathcal{SH}$
NCI-1	27652	70	0	46800	140	0	$\mathcal{ALE}$
Gazetteer	150979	2	214804	167349	2	214804	$\mathcal{ALE}+$
GALEN-doctored	2748	413	0	3937	799	0	$\mathcal{ALEHIF}+$
GALEN-undoctored	2748	413	0	4179	800	0	$\mathcal{ALEHIF}+$
CHEBI	20977	9	243972	38375	2	243972	$\mathcal{ALE}+$
FMA-Lite	75141	2	46225	119558	3	46225	$\mathcal{ALEI}+$
SWEET Phenomena	1728	145	171	2419	239	491	$\mathcal{SHOIN}(\mathcal{D})$
SWEET Numerics	1506	177	113	2184	305	340	$\mathcal{SHOIN}(\mathcal{D})$
Wine	138	17	206	355	40	494	$\mathcal{SHOIN}(\mathcal{D})$
DOLCE-Plans	118	264	27	265	948	68	$\mathcal{SHOIN}(\mathcal{D})$
NCI-2	70576	189	0	100304	290	0	$\mathcal{ALCH}(\mathcal{D})$
FMA-Constitutional	41648	168	85	122695	395	86	$\mathcal{ALCOIF}(\mathcal{D})$

I tempi richiesti per la classificazione

Ontology Name	Classification Times (seconds)			
	HermiT	HermiT-Anc	Pellet	FaCT++
Molecule Role	3.3	3.4	25.7	304.5
XP Uber Anatomy	5.4	4.9	—	86.0
XP Plant Anatomy	12.8	11.2	87.2	22.9
XP Regulators	14.1	17.1	35.4	66.6
Celular Component	18.6	18.0	40.5	76.7
NCI-1	14.1	14.4	23.2	3.0
Gazetteer	131.9	132.3	—	—
GALEN-doctored	8.8	456.3	—	15.9
GALEN-undoctored	126.3	—	—	—
CHEBI	24.2	—	—	397.0
FMA-Lite	107.2	—	—	—
SWEET Phenomena	13.5	11.2	—	0.2
SWEET Numerics	76.7	72.6	3.7	0.2
Wine	343.7	524.6	19.5	162.1
DOLCE-Plans	1075.1	—	105.1	—
NCI-2	—	—	172.0	60.7
FMA-Constitutional	—	—	—	616.7

Note: entry — means that reasoner was unable to classify the ontology either due to time out or memory exhaustion.

Progressi recenti e conclusioni

- La complessità elevatissima del caso *peggiore* non impedisce l'utilizzo in pratica di OWL e OWL2
- In aggiunta sono stati perfezionati metodi per focalizzare rapidamente il ragionamento sugli assiomi rilevanti *a una query data*
- Gli speedup possono essere di diversi ordini di grandezza
- In ogni caso nel migliorare gli algoritmi di inferenza ricordare che:
  - se sono corretti e completi per uno dei frammenti completi per **EXP** non possono impiegare tempo polinomiale (**P** $\neq$ **EXP**)
  - se sono corretti e completi per $\mathcal{SROIQ}$ (OWL2), che è 2-**EXP**-completo, non possono nemmeno usare spazio polinomiale (**PSPACE** $\subset$ **EXPSPACE** $\subseteq$ 2-**EXP**)

se pensate di esserci riusciti, allora l'algoritmo è necessariamente *errato*

Esercizio 1

- Sapendo che:
  - Si dispone di un classificatore automatico per la vecchia logica descrittiva $\mathcal{ALC}$
  - Il ragionamento in $\mathcal{ALC}$ è **EXP**-completo
- Si vuole ottenere un classificatore per $\mathcal{SHOIN}$ (OWL-DL) mediante
  1. preprocessing che traduce la *KB* $\mathcal{SHOIN}$ in una $\mathcal{ALC}$
  2. invocazione del classificatore per $\mathcal{ALC}$
- Dire quanto può costare il preprocessing

Esercizio 2

- Supponiamo di avere un ragionatore automatico *RSB* per il frammento di Schönfinkel-Bernays

1. Dire se possiamo implementare le inferenze in $\mathcal{ALC}$
   1. traducendo la *KB* e le query nel frammento di Schönfinkel-Bernays
   2. invocando *RSB* sulla traduzione

2. Dire quanto può essere resa efficiente la traduzione

Esercizio 3

1. Dire se HAMILTON PATH può essere velocizzato accorciando i grafi dati con dei circuiti
2. Dire se possono esistere riduzioni tra SUCCINCT CIRCUIT VALUE e SUCCINCT 3-SAT
3. Dire se possono esistere riduzioni tra 3-SAT e SUCCINCT 3-SAT
4. Dire se possono esistere riduzioni tra 3-SAT e il ragionamento in (OWL2-DL)
5. Dire se possono esistere riduzioni tra GEOGRAPHY e il ragionamento in (OWL2-DL)

Senza garanzia di terminazione:

RE e **coRE**

Problemi non decidibili

- Tutti i problemi esaminati sinora erano *decidibili*, ovvero possono essere risolti con una MdT che *termina sempre*
- Sappiamo che questo non è sempre possibile, ad es.
  - stabilire la soddisfacibilità o la validità di una arbitraria formula del primo ordine
  - capire se un programma termina per tutti gli input, o anche solo per un input dato
  - dire se un programma solleverà una eccezione a runtime (anche solo per il caso particolare dei downcast)
  - praticamente ogni proprietà di un algoritmo (o proprietà di un programma che dipende dalla sua *esecuzione*)
- Questi problemi non hanno tutti la stessa complessità (come per i problemi intrattabili)

Cosa discuteremo

- Oltre la decidibilità, le "leggi fisiche" degli algoritmi cambiano in qualche misura
- Faremo alcune osservazioni per confrontare le caratteristiche dei problemi non decidibili con alcuni aspetti del nondeterminismo e degli oracoli

Ricordiamo che

- Dato un linguaggio $L \subset (\Sigma \setminus \{\sqcup\})^*$
- Se esiste una MdT M tale che, per ogni stringa $x \in (\Sigma \setminus \{\sqcup\})^*$

$$M(x) = \begin{cases} \text{"}yes\text{"} & \text{se } x \in L \\ \nearrow & \text{altrimenti} \end{cases}$$

allora diciamo che

1. *M accetta L*
2. *L* è un linguaggio *ricorsivamente enumerabile* o *semidecidibile*

Definizione di RE, coRE e R

RE è la classe di problemi ricorsivamente enumerabili e **coRE** è la classe dei loro complementi. **R** denota la classe di problemi decidibili.

Due problemi in **RE** e **coRE**

Teorema (1^{st}-order validity)

Decidere se una formula del 1^o ordine ϕ data è valida è in $\mathbf{RE} \setminus \mathbf{R}$

- L'appartenenza a **RE** è facile: enumerare tutte le possibili dimostrazioni e vedere se provano ϕ
- L'indecidibilità deriva da quella della terminazione delle MdT e dalla possibilità di ridurre le computazioni delle MdT alla validità di formule logiche

Corollario (1^{st}-order sat)

Decidere se una formula del 1^o ordine ϕ è soddisfacibile è in $\mathbf{coRE} \setminus \mathbf{R}$

- Notare che ϕ è soddisfacibile $\Leftrightarrow \neg\phi$ non è valida

Ruolo del nondeterminismo in **RE**

- Se ci si focalizza sulla decidibilità dei problemi, allora il nondeterminismo non ha alcun ruolo significativo

Proposizione

Se L è accettato da una MdT nondeterministica N allora è accettato anche da una MdT deterministica M

- **Prova**: Basta che M simuli tutte le computazioni di N troncandole dopo k passi, per $k = 1, 2, .., i, ...$
- Se N accetta x in k passi, M termina e accetta x mentre esplora le computazioni lunghe k
- Se N non accetta x allora M diverge

QED

Condizioni di (non) accettazione per **coRE**

- Se invertiamo gli stati "yes" e "no" di una MdT M che accetta $L \in \mathbf{RE}$ otteniamo una M' tale che per ogni input x

$$M(x) = \begin{cases} \nearrow & \text{se } x \in \bar{L} \\ \text{"}no\text{"} & \text{se } x \notin \bar{L} \end{cases}$$

- Quindi...

Certificati finiti

- Supponiamo che M accetti L e che $x \in L$
- Come per **NP**, possiamo rappresentare con una stringa la computazione (finita) di $M(x)$
- Si può verificare in tempo finito (è *decidibile*, addirittura in **P**) se una stringa data codifichi una computazione di M
- Queste stringhe provano che $x \in L$ e che $x \notin \bar{L}$
- Quindi, usando queste stringhe come certificati è immediato che
  - **RE** è la classe di problemi che ammettono certificati finiti
  - **coRE** è la classe di problemi che ammettono disqualifier finiti
  - (confrontare con i certificati per **NP** e **coNP**)

Condizioni esistenziali/universali di (non) accettazione

- **RE**: Se M accetta L, allora per ogni input x
  - se $x \in L$, esiste $t \in \mathbb{N}$ tale che M termina in t passi
  - se $x \notin L$, per ogni $t \in \mathbb{N}$, M non termina in t passi

- **coRE**: Se M' è M con gli output invertiti
  - se $x \in \bar{L}$, per ogni $t \in \mathbb{N}$, M' non termina in t passi
  - se $x \notin \bar{L}$, esiste $t \in \mathbb{N}$ tale che M' termina in t passi

- Confrontare con le condizioni di accettazione duali per **NP** e **coNP**
  - "esiste / per ogni run..."
  - per **RE** e **coRE** riguarda il tempo in cui compare una configurazione finale *nella singola computazione deterministica*

Intersezione di classi complementari

Teorema

Se $L \in \mathbf{RE} \cap \mathbf{coRE}$ allora L è decidibile

- **Prova**: Devono esistere M^+ che termina con "yes" se $x \in L$ e M^- che termina con "no" se $x \notin L$
- Simulare i primi k passi delle computazioni di M^+ ed M^- per $k = 1, 2, ...$
- Dopo un numero finito di passi una delle due termina e ci dice se x appartiene a L o no.

QED

Corollario

$\mathbf{R} = \mathbf{RE} \cap \mathbf{coRE}$

- Invece non sappiamo se $\mathbf{P} = \mathbf{NP} \cap \mathbf{coNP}$ (presumibillmente no)

MdT con oracolo

- Gerarchia di Kleene: gli oracoli del livello 1 sono in **RE**
  - o indifferentemente in **coRE**: l'oracolo risponde sempre e questo "cancella le differenze" tra **RE** e **coRE**
- Gli oracoli dei livelli successivi sono al livello precedente, come in **PH**
- Oltre il 1^o livello non esistono certificati finiti nè per L nè per $\bar{L}$
- Esempio: la Circumscription di KB del primo ordine è al 2^o livello della gerarchia di Kleene
  - così come diverse altre logiche nonmonotòne

Materiale di riferimento

- Papadimitriou:
  - Parte V, Capitolo 20
  - Parte II, Capitolo 3, paragrafi 3.2, 3.3

Esercizio 4

- Vi chiedono di scrivere un algoritmo che per ogni formula del 1^o ordine ϕ produca una descrizione finita di una interpretazione che la soddisfa, mentre se ϕ è insoddisfacibile termina in "no".
- Voi cosa rispondete?

Esercizio 4b

- Allora rilassano le specifiche:
- Vi chiedono di scrivere un algoritmo che per ogni formula del 1^o ordine ϕ produca una descrizione finita di una interpretazione che la soddisfa. Se ϕ è insoddisfacibile può divergere.
- Voi cosa rispondete?