

CS151 Complexity Theory

Lecture 3
April 3, 2007

A P-complete problem

- Circuit Value (CVAL): given a variable-free Boolean circuit (gates $\wedge, \vee, \neg, 0, 1$), does it output 1?

Theorem: CVAL is P-complete.

- Proof:
 - already argued in P
 - L arbitrary language in P, TM M decides L in n^k steps

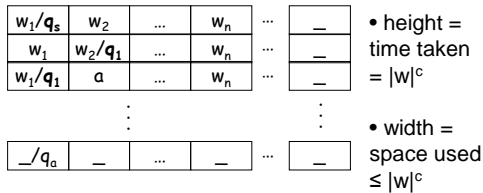
April 3, 2007

CS151 Lecture 3

2

A P-complete problem

- Tableau (configurations written in an array) for machine M on input w:



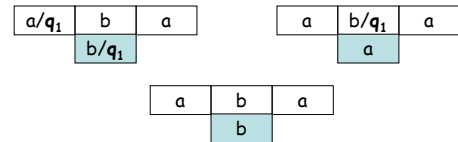
April 3, 2007

CS151 Lecture 3

3

A P-complete problem

- Important observation: contents of cell in tableau determined by 3 others above it:



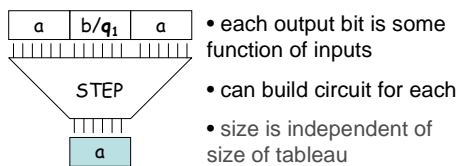
April 3, 2007

CS151 Lecture 3

4

A P-complete problem

- Can build Boolean circuit STEP
 - input (binary encoding of) 3 cells
 - output (binary encoding of) 1 cell



April 3, 2007

CS151 Lecture 3

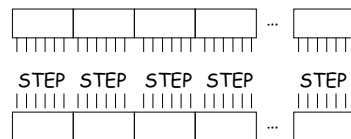
5

A P-complete problem

Tableau for M on input w

w_1/q_s	w_2	...	w_n	...	—
w_1	w_2/q_1	...	w_n	...	—
...
$_/q_a$	—	...	—	...	—

- $|w|^c$ copies of STEP compute row i from i-1



April 3, 2007

CS151 Lecture 3

6

A P-complete problem

This circuit $C_{M,w}$ has inputs w_1, w_2, \dots, w_n and $C(w) = 1$ iff M accepts w .

logspace reduction
Size = $O(|w|^{2c})$

ignore these
1 iff cell contains q_{accept}

April 3, 2007 CS151 Lecture 3 7

Answer to question

- Can we evaluate an n node Boolean circuit using $O(\log n)$ space?
- NO! (probably)
- CVAL in L if and only if $L = P$

April 3, 2007 CS151 Lecture 3 8

Padding and succinctness

Two consequences of measuring running time as function of input length:

- “padding”
 - suppose $L \in \mathbf{EXP}$, and define $\text{PAD}_L = \{x\#^N : x \in L, N = 2^{|x|^k}\}$
 - TM that decides PAD_L : ensure suffix of N #s, ignore #s, then simulate TM that decides L
 - running time now polynomial!

April 3, 2007 CS151 Lecture 3 9

Padding and succinctness

- converse (intuitively): “succinctness”
 - suppose L is **P-complete**
 - intuitively, some inputs are “hard” -- require full power of P
 - SUCCINCT_L has input encoded in exponentially shorter form than L
 - if “hard” inputs encodable this way, then candidate to be **EXP-complete**

April 3, 2007 CS151 Lecture 3 10

An EXP-complete problem

- succinct encoding for a directed graph $G = (V = \{1, 2, 3, \dots\}, E)$: 1 iff $(i, j) \in E$
- a succinct encoding for a variable-free Boolean circuit:

1 iff wire from gate i to gate j

type of gate i

type of gate j

1 iff $(i, j) \in E$

April 3, 2007 CS151 Lecture 3 11

An EXP-complete problem

- Succinct Circuit Value: given a **succinctly encoded** variable-free Boolean circuit (gates $\wedge, \vee, \neg, 0, 1$), does it output 1?

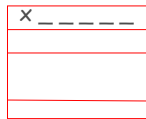
Theorem: Succinct Circuit Value is **EXP-complete**.

- Proof:
 - in **EXP** (why?)
 - L arbitrary language in **EXP**, TM M decides L in 2^{n^k} steps

April 3, 2007 CS151 Lecture 3 12

An EXP-complete problem

– tableau for input $x = x_1x_2x_3\dots x_n$:



height,
width 2^{n^k}

- Circuit C from CVAL reduction has size $O(2^{2n^k})$.
- TM M accepts input x iff circuit outputs 1

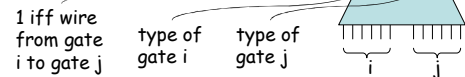
April 3, 2007

CS151 Lecture 3

13

An EXP-complete problem

– Can encode C succinctly:



1 iff wire
from gate
i to gate j

type of
gate i type of
gate j

- if i, j within single STEP circuit, easy to compute output
- if i, j between two STEP circuits, easy to compute output
- if one of i, j refers to input gates, consult x to compute output

April 3, 2007

CS151 Lecture 3

14

Summary

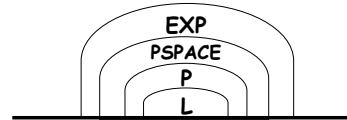
- Remaining TM details: big-oh necessary.
- First complexity classes:
L, P, PSPACE, EXP
- First separations (via simulation and diagonalization):
P ≠ EXP, L ≠ PSPACE
- First major open questions:
L = P ? P = PSPACE ?
- First complete problems:
 - CVAL is **P**-complete
 - Succinct CVAL is **EXP**-complete

April 3, 2007

CS151 Lecture 3

15

Summary



April 3, 2007

CS151 Lecture 3

16

Nondeterminism: introduction

A motivating question:

- Can computers replace mathematicians?

$L = \{ (x, 1^k) : \text{statement } x \text{ has a proof of length at most } k \}$

April 3, 2007

CS151 Lecture 3

17

Nondeterminism: introduction

- This lecture:
 - nondeterminism
 - nondeterministic time classes
 - **NP, NP-completeness, P vs. NP**
 - **coNP**
 - NTIME Hierarchy
 - Ladner's Theorem

April 3, 2007

CS151 Lecture 3

18

Nondeterminism

- Recall deterministic TM
 - Q finite set of states
 - Σ alphabet including blank: “_”
 - $q_{start}, q_{accept}, q_{reject}$ in Q
 - transition function:

$$\delta : Q \times \Sigma \rightarrow Q \times \Sigma \times \{L, R, -\}$$

April 3, 2007

CS151 Lecture 3

19

Nondeterminism

- nondeterministic** Turing Machine:
 - Q finite set of states
 - Σ alphabet including blank: “_”
 - $q_{start}, q_{accept}, q_{reject}$ in Q
 - transition **relation**

$$\Delta \subset (Q \times \Sigma) \times (Q \times \Sigma \times \{L, R, -\})$$
- given current state and symbol scanned, several choices of what to do next.

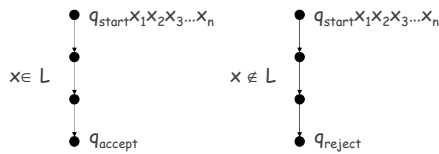
April 3, 2007

CS151 Lecture 3

20

Nondeterminism

- deterministic TM: given current configuration, unique next configuration

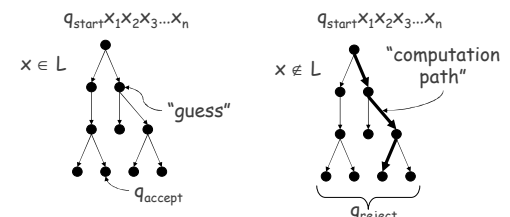


April 3, 2007

CS151 Lecture 3

21

Nondeterminism



- asymmetric accept/reject criterion

April 3, 2007

CS151 Lecture 3

22

Nondeterminism

- all paths terminate
- time used: maximum length of paths from root
- space used: maximum # of work tape squares touched on any path from root

April 3, 2007

CS151 Lecture 3

23

Nondeterminism

- NTIME(f(n))** = languages decidable by a multi-tape NTM that runs for at most $f(n)$ steps *on any computation path*, where n is the input length, and $f : \mathbf{N} \rightarrow \mathbf{N}$
- NSPACE(f(n))** = languages decidable by a multi-tape NTM that touches at most $f(n)$ squares of its work tapes *along any computation path*, where n is the input length, and $f : \mathbf{N} \rightarrow \mathbf{N}$

April 3, 2007

CS151 Lecture 3

24

Nondeterminism

- Focus on time classes first:

$$NP = \cup_k NTIME(n^k)$$

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

April 3, 2007

CS151 Lecture 3

25

Poly-time verifiers

Very useful alternative definition of NP:

“witness” or “certificate”

Theorem: language L is in NP iff it is expressible as:

efficiently verifiable

$$L = \{ x \mid \exists y, |y| \leq |x|^k, (x, y) \in R \}$$

where R is a language in P.

- poly-time TM M_R deciding R is a “verifier”

April 3, 2007

CS151 Lecture 3

26

Poly-time verifiers

- Example: 3SAT expressible as
 - 3SAT = $\{ \varphi \mid \varphi \text{ is a 3-CNF formula for which } \exists \text{ assignment } A \text{ for which } (\varphi, A) \in R \}$
 - $R = \{ (\varphi, A) \mid A \text{ is a sat. assign. for } \varphi \}$
 - satisfying assignment A is a “witness” of the satisfiability of φ (it “certifies” satisfiability of φ)
 - R is decidable in poly-time

April 3, 2007

CS151 Lecture 3

27

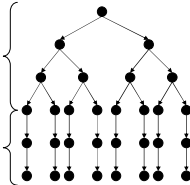
Poly-time verifiers

$$L = \{ x \mid \exists y, |y| \leq |x|^k, (x, y) \in R \}$$

Proof: (\Leftarrow) give poly-time NTM deciding L

phase 1: “guess” y with $|x|^k$ nondeterministic steps

phase 2: decide if $(x, y) \in R$



April 3, 2007

CS151 Lecture 3

28

Poly-time verifiers

- Proof:** (\Rightarrow) given $L \in NP$, describe L as:
- $$L = \{ x \mid \exists y, |y| \leq |x|^k, (x, y) \in R \}$$
- L is decided by NTM M running in time n^k
 - define the language
 - $R = \{ (x, y) \mid y \text{ is an accepting computation history of } M \text{ on input } x \}$
 - check: accepting history has length $\leq |x|^k$
 - check: R is decidable in polynomial time
 - check: M accepts x iff $\exists y, |y| \leq |x|^k, (x, y) \in R$

April 3, 2007

CS151 Lecture 3

29

Why NP?

problem requirements vs. static model of object we are seeking

- but, captures important computational feature of many problems:

exhaustive search works

- contains huge number of natural problems
- many problems have form:
 - efficient test: does y meet requirements?

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

April 3, 2007

CS151 Lecture 3

30

Why NP?

- Why not **EXP**?
 - too strong!
 - important problems not complete.

April 3, 2007

CS151 Lecture 3

31

Relationships between classes

- Easy: $P \subset NP$, $EXP \subset NEXP$
 - TM special case of NTM
- Recall: $L \in NP$ iff expressible as
 $L = \{ x \mid \exists y, |y| \leq |x|^k, (x, y) \in R \}$
- $NP \subset PSPACE$ (try all possible y)
- The central question:
 $P \stackrel{?}{=} NP$
 finding a solution vs. recognizing a solution

April 3, 2007

CS151 Lecture 3

32

NP-completeness

- Circuit SAT: given a Boolean circuit (gates \wedge, \vee, \neg), with variables y_1, y_2, \dots, y_m is there some assignment that makes it output 1?

Theorem: Circuit SAT is NP-complete.

- Proof:
 - clearly in NP

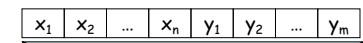
April 3, 2007

CS151 Lecture 3

33

NP-completeness

- Given $L \in NP$ of form
 $L = \{ x \mid \exists y \text{ s.t. } (x, y) \in R \}$



1 iff $(x, y) \in R$ CVAL reduction for R

- hardwire input x ; leave y as variables

April 3, 2007

CS151 Lecture 3

34

NEXP-completeness

- Succinct Circuit SAT: given a **succinctly encoded** Boolean circuit (gates \wedge, \vee, \neg), with variables y_1, y_2, \dots, y_m is there some assignment that makes it output 1?

Theorem: Succinct Circuit SAT is NEXP-complete.

- Proof:
 - same trick as for Succinct CVAL EXP-complete.

April 3, 2007

CS151 Lecture 3

35

Complement classes

- In general, if C is a complexity class
- $co-C$ is the complement class, containing all complements of languages in C
 - $L \in C$ implies $(\Sigma^* - L) \in co-C$
 - $(\Sigma^* - L) \in C$ implies $L \in co-C$
- Some classes closed under complement:
 - e.g. $co-P = P$

April 3, 2007

CS151 Lecture 3

36

coNP

- Is NP closed under complement?

Can we transform this machine: $x \in L$ $x \notin L$

into this machine?

April 3, 2007 CS151 Lecture 3 37

coNP

- “proof system” interpretation:
- Recall: $L \in \mathbf{NP}$ iff expressible as $L = \{ x \mid \exists y, |y| \leq |x|^k, (x, y) \in R \}$
 - “proof”
 - “proof verifier”
- languages in **NP** have “short proofs”
- coNP** captures (in its complete problems) problems least likely to have “short proofs”.
 - e.g., UNSAT is **coNP**-complete

April 3, 2007 CS151 Lecture 3 38

coNP

- $\mathbf{P} = \mathbf{NP}$ implies $\mathbf{NP} = \mathbf{coNP}$
- Belief:

$\mathbf{NP} \neq \mathbf{coNP}$

 - another major open problem

April 3, 2007 CS151 Lecture 3 39

NTIME Hierarchy Theorem

Theorem (Nondeterministic Time Hierarchy Theorem):

For every proper complexity function $f(n) \geq n$, and $g(n) = \omega(f(n+1))$,

$$\mathbf{NTIME}(f(n)) \subsetneq \mathbf{NTIME}(g(n)).$$

April 3, 2007 CS151 Lecture 3 40

NTIME Hierarchy Theorem

Proof attempt #1: (what's wrong?)

April 3, 2007 CS151 Lecture 3 41

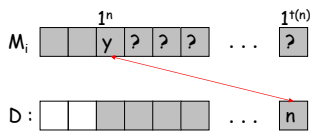
NTIME Hierarchy Theorem

- Let $t(n)$ be large enough so that can decide if NTM M running in time $f(n)$ accepts 1^n , in time $t(n)$.

April 3, 2007 CS151 Lecture 3 42

NTIME Hierarchy Theorem

- Enough time on input $1^{t(n)}$ to do the *opposite* of $M_i(1^n)$:



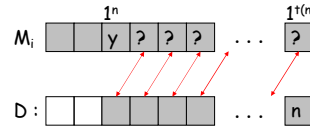
April 3, 2007

CS151 Lecture 3

43

NTIME Hierarchy Theorem

- For k in $[n \dots t(n)]$ can to do *same* as $M_i(1^{k+1})$ on input 1^k



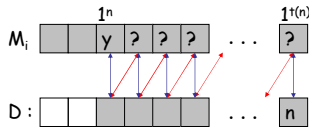
April 3, 2007

CS151 Lecture 3

44

NTIME Hierarchy Theorem

- Did we diagonalize against M_i ?
 - if $L(M_i) = L(D)$ then:



- equality along all arrows.
- contradiction.

April 3, 2007

CS151 Lecture 3

45

NTIME Hierarchy Theorem

- General scheme:
 - interval $[1 \dots t(1)]$ kills M_1
 - interval $[t(1) \dots t(t(1))]$ kills M_2
 - interval $[t^{i-1}(1) \dots t^i(1)]$ kills M_i
- Running time of D on 1^n : $f(n+1)$ + time to compute interval containing n
- conclude D in **NTIME(g(n))**

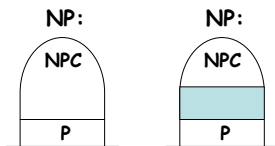
April 3, 2007

CS151 Lecture 3

46

Ladner's Theorem

- Assuming $P \neq NP$, what does the world (inside NP) look like?



April 3, 2007

CS151 Lecture 3

47

Ladner's Theorem

Theorem (Ladner): If $P \neq NP$, then there exists $L \in NP$ that is neither in P nor NP -complete.

- Proof: "lazy diagonalization"
 - deal with similar problem as in NTIME Hierarchy proof

April 3, 2007

CS151 Lecture 3

48

Ladner's Theorem

- Can enumerate (TMs deciding) all languages in **P**.
 - enumerate TMs so that each machine appears infinitely often
 - add clock to M_i so that it runs in at most n^i steps

April 3, 2007

CS151 Lecture 3

49

Ladner's Theorem

- Can enumerate (TMs deciding) all **NP**-complete languages.
 - enumerate TMs f_i computing all polynomial-time functions
 - machine N_i decides language SAT reduces to via f_i if f_i is reduction, else SAT (details omitted...)

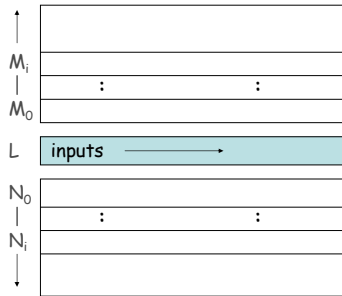
April 3, 2007

CS151 Lecture 3

50

Ladner's Theorem

- Our goal: $L \in \mathbf{NP}$ that is neither in **P** nor **NP**-complete



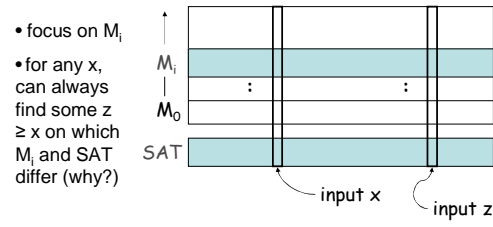
April 3, 2007

CS151 Lecture 3

51

Ladner's Theorem

- Top half, assuming $\mathbf{P} \neq \mathbf{NP}$:



April 3, 2007

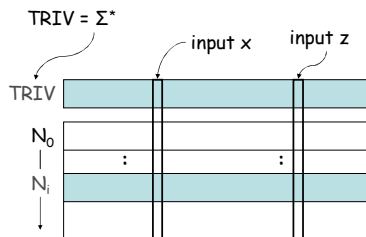
CS151 Lecture 3

52

Ladner's Theorem

- Bottom half, assuming $\mathbf{P} \neq \mathbf{NP}$:

- focus on N_i
- for any x , can always find some $z \geq x$ on which N_i and TRIV differ (why?)



April 3, 2007

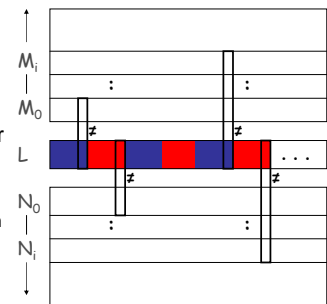
CS151 Lecture 3

53

Ladner's Theorem

- Try to "merge":

- **SAT** **TRIV**
- on input x , either
 - answer SAT(x)
 - answer TRIV(x)
- if can decide which one in **P**, $L \in \mathbf{NP}$



April 3, 2007

CS151 Lecture 3

54