CS151
Complexity Theory

Lecture 4

April 8, 2021

NTIME Hierarchy Theorem

Theorem (Nondeterministic Time Hierarchy Theorem):

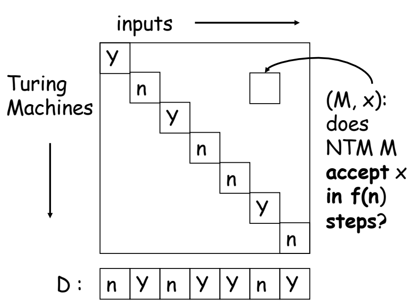
For every *proper complexity function* $f(n) \geq n$, and $g(n) = \omega(f(n+1))$,

NTIME(f(n)) $\subsetneq$ **NTIME(g(n))**.

NTIME Hierarchy Theorem

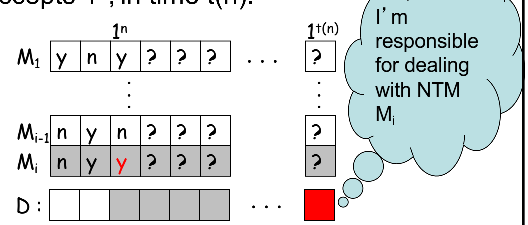
Proof attempt :

(what's wrong?)



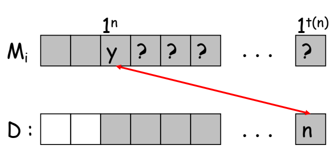
NTIME Hierarchy Theorem

- 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).



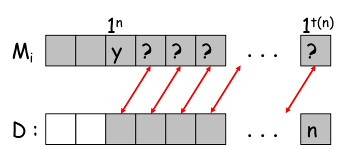
NTIME Hierarchy Theorem

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



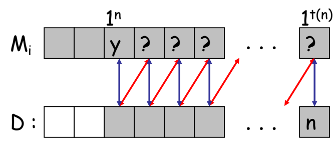
NTIME Hierarchy Theorem

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



NTIME Hierarchy Theorem

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



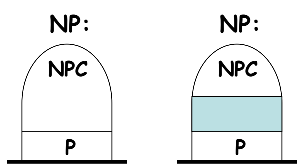
  - equality along all arrows.
  - contradiction.

NTIME Hierarchy Theorem

- General scheme:
  - interval $[1\ldots t(1)]$ kills M_1
  - interval $[t(1)\ldots t(t(1))]$ kills M_2
  - interval $[t^{i-1}(1)\ldots t^i(1)]$ 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))** **(**$g(n) = \omega(f(n+1))$**)**

Ladner's Theorem

- Assuming **P** ≠ **NP**, what does the world (inside **NP**) look like?



Ladner's Theorem

Theorem (Ladner): If **P** ≠ **NP**, then there exists L ∈ **NP** that is neither in **P** nor **NP**-complete.

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

Ladner's Theorem

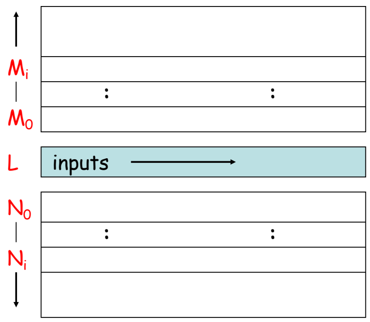
- Can enumerate (TMs deciding) all languages in **P**.
  - enumerate TMs so that each machine appears infinitely often
  - add clock to M_i so that it runs in at most n^i steps

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…)

Ladner’s Theorem

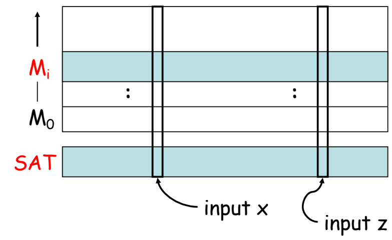
- Our goal: L ∈ **NP** that is neither in **P** nor **NP**-complete



April 8, 2021 CS151 Lecture 4

Ladner’s Theorem

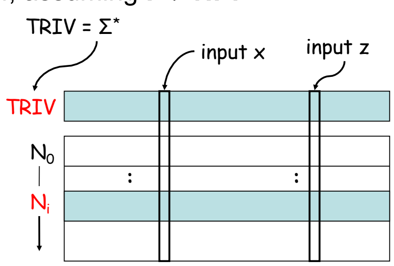
- Top half, assuming **P ≠ NP:**
  - focus on M_i
  - for any x, can always find some z ≥ x on which M_i and SAT differ (why?)



April 8, 2021 CS151 Lecture 4

Ladner’s Theorem

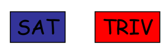
- Bottom half, assuming **P ≠ NP:**
  - focus on N_i
  - for any x, can always find some z ≥ x on which N_i and TRIV differ (why?)



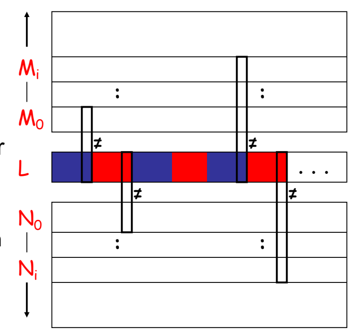
April 8, 2021 CS151 Lecture 4

Ladner’s Theorem

- Try to “merge”:



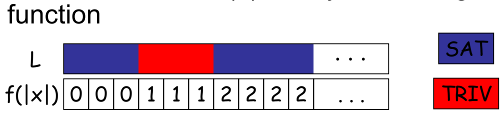
- on input x, either
  - answer SAT(x)
  - answer TRIV(x)
- if can decide which one in **P**, L ∈ **NP**



April 8, 2021 CS151 Lecture 4

Ladner’s Theorem

- General scheme: f(n) slowly increasing function



  - f(|x|) even: answer SAT(x)
  - f(|x|) odd: answer TRIV(x)
- notice choice only depends on length of input… that’s OK

April 8, 2021 CS151 Lecture 4

Ladner’s Theorem

- 1st attempt to define f(n)
- “eager f(n)”: increase at 1st opportunity
- Inductive definition: f(0) = 0; f(n) =
  - if f(n-1) = 2i, trying to kill M_i
    - if ∃ z < 1^n s.t. $M_i(z) \neq$ SAT(z), then f(n) = f(n-1) + 1; else f(n) = f(n-1)
  - if f(n-1) = 2i+1, trying to kill N_i
    - if ∃ z < 1^n s.t. $N_i(z) \neq$ TRIV(z), then f(n) = f(n-1) + 1; else f(n) = f(n-1)

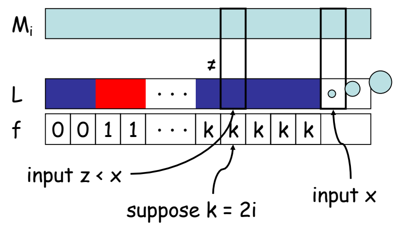
April 8, 2021 CS151 Lecture 4

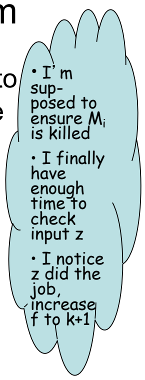
Ladner's Theorem

- Problem: eager f(n) too difficult to compute
- on input of length n,
  - look at all strings z of length < n
  - compute SAT(z) or $N_i(z)$ for each !
- Solution: "lazy" f(n)
  - on input of length n, only run for 2n steps
  - if enough time to see should increase (over f(n-1)), do it; else, stay same
  - (alternate proof: give explicit f(n) that grows slowly enough…)

Ladner's Theorem

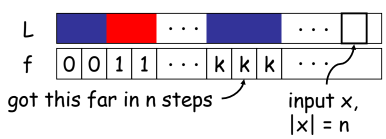
- Key: n eventually large enough to notice completed previous stage





Ladner's Theorem

- Inductive definition of f(n)
  - f(0) = 0
  - f(n): for n steps compute f(0), f(1), f(2),…



Ladner's Theorem

- if k = 2i:
  - for n steps try (lex order) to find z s.t. SAT(z) ≠ $M_i(z)$ and f(|z|) even
  - if found, f(n) = f(n-1)+1 else f(n-1)
- if k = 2i + 1:
  - for n steps try (lex order) to find z s.t. TRIV(z) ≠ $N_i(z)$ and f(|z|) odd
  - if found, f(n) = f(n-1)+1 else f(n-1)

Ladner's Theorem

- Finishing up:

$$L = \{ x \mid x \in \text{SAT if } f(|x|) \text{ even}, \; x \in \text{TRIV if } f(|x|) \text{ odd} \}$$

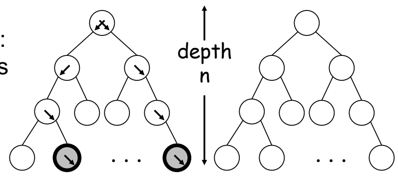
- L ∈ **NP** since f(|x|) can be computed in O(n) time

Ladner's Theorem

- suppose M_i decides L
  - f gets stuck at 2i
  - L ≡ SAT for z : |z| > n_o
  - implies SAT ∈ **P**. Contradiction.
- suppose N_i decides L
  - f gets stuck at 2i+1
  - L ≡ TRIV for z : |z| > n_o
  - implies $L(N_i)$ ∈ **P**. Contradiction.
- (last of diagonalization…)

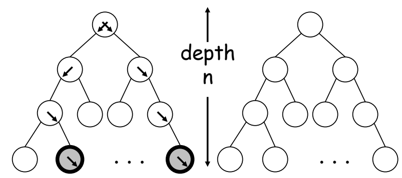
A puzzle

A puzzle:
two kinds
of trees

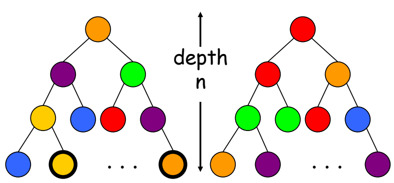


- cover up nodes with c colors
- promise: never color "arrow" same as "blank"
- determine which kind of tree in poly(n, c) steps?

A puzzle



A puzzle



Introduction

- Ideas
  - depth-first-search; stop if see
  - how many times may we see a given "arrow color"?
    - at most n+1
  - pruning rule?
    - if see a color > n+1 times, it can't be an arrow node; prune
  - # nodes visited before know answer?
    - at most c(n+2)

Sparse languages and **NP**

- We often say **NP**-compete languages are "hard"
- More accurate: **NP**-complete languages are "expressive"
  - lots of languages reduce to them

Sparse languages and **NP**

- Sparse language: one that contains at most poly(n) strings of length ≤ n
- not very expressive – can we show this cannot be **NP**-complete (assuming **P** ≠ **NP**) ?
  - yes: Mahaney '82 (homework problem)
- Unary language: subset of 1* (at most n strings of length ≤ n)

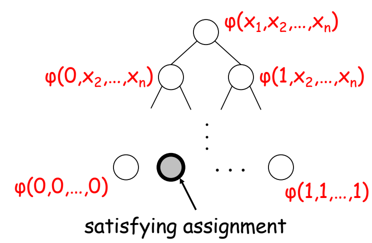
Sparse languages and **NP**

Theorem (Berman '78): if a unary language is **NP**-complete then **P** = **NP**.

- Proof:
  - let $U \subseteq 1^*$ be a unary language and assume SAT $\leq$ U via reduction R
  - $\varphi(x_1,x_2,\ldots,x_n)$ instance of SAT

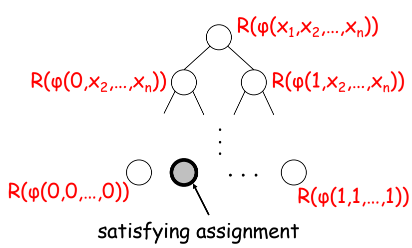
Sparse languages and **NP**

– self-reduction tree for φ:



Sparse languages and **NP**

– applying reduction R:



Sparse languages and **NP**

- on input of length $m = |\varphi(x_1,x_2,\ldots,x_n)|$, R produces string of length $\leq p(m)$
- R's different outputs are "colors"
  - 1 color for strings not in 1^*
  - at most p(m) other colors

- puzzle solution $\Rightarrow$ can solve SAT in $poly(p(m)+1, n+1) = poly(m)$ time!

Summary

- nondeterministic time classes:

  NP, coNP, NEXP
- NTIME Hierarchy Theorem:

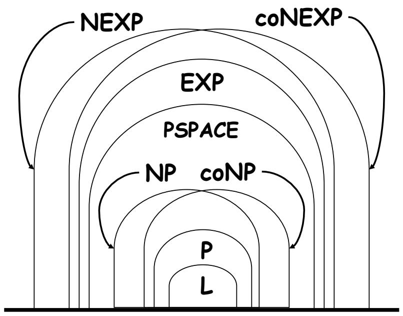
  NP ≠ NEXP
- major open questions:

  $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ $\quad$ $\mathbf{NP} \stackrel{?}{=} \mathbf{coNP}$

Summary

- **NP**-"intermediate" problems (unless **P** = **NP**)
  - technique: delayed diagonalization
- unary languages not **NP**-complete (unless **P** = **NP**)
  - true for sparse languages as well (homework)
- complete problems:
  - circuit SAT is **NP**-complete
  - UNSAT is **coNP**-complete
  - succinct circuit SAT is **NEXP**-complete

Summary



Remainder of lecture

- nondeterminism applied to space
- reachability
- two surprises:
  - Savitch's Theorem
  - Immerman/Szelepcsényi Theorem