1

Relationships between classes

- So far:

$$\mathbf{L} \subseteq \mathbf{P} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXP}$$

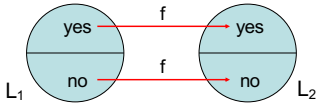
- believe all containments strict
- know $\mathbf{L} \subsetneq \mathbf{PSPACE}$, $\mathbf{P} \subsetneq \mathbf{EXP}$
- even before any mention of NP, two **major** unsolved problems:

$$\mathbf{L} \stackrel{?}{=} \mathbf{P} \qquad \mathbf{P} \stackrel{?}{=} \mathbf{PSPACE}$$

2

A **P**-complete problem

- We don't know how to prove $\mathbf{L} \neq \mathbf{P}$
- But, can identify problems in **P** least likely to be in **L** using **P**- completeness.
- need stronger notion of reduction (why?)



3

A **P**-complete problem

- **logspace reduction**: f computable by TM that uses O(log n) space
  - denoted "$L_1 \leq_L L_2$"
- If L_2 is **P**-complete, then L_2 in **L** implies **L** = **P** (homework problem)

4

A **P**-complete problem

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

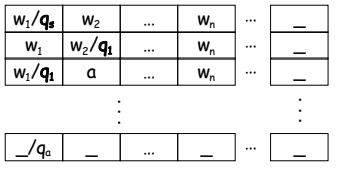
Theorem: CVAL is **P**-complete.

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

5

A **P**-complete problem

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

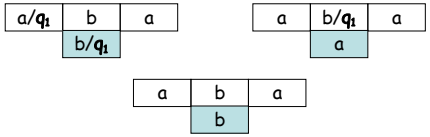


- height = time taken = $|w|^c$
- width = space used $\leq |w|^c$

6

A **P**-complete problem

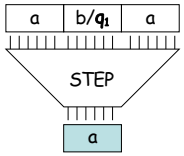
- Important observation: contents of cell in tableau determined by 3 others above it:



7

A **P**-complete problem

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

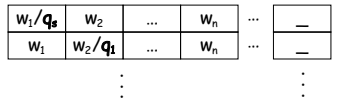


- each output bit is some function of inputs
- can build circuit for each
- size is independent of size of tableau

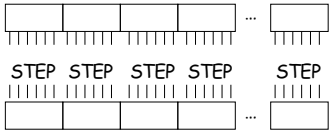
8

A **P**-complete problem

Tableau for M on input w

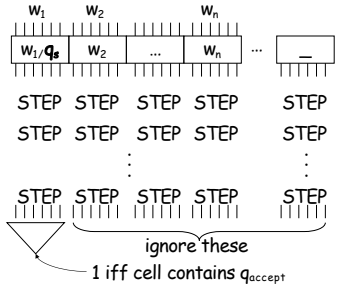


- $|w|^c$ copies of STEP compute row i from i-1



9

A **P**-complete problem



This circuit $C_{M,w}$ has inputs $w_1w_2\ldots w_n$ and outputs 1 iff M accepts input w.

logspace reduction

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

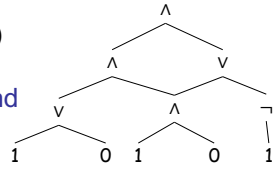
10

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**



11

Padding and succinctness

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

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

12

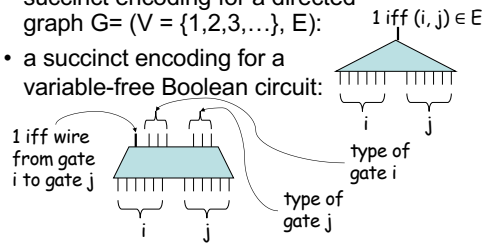
Padding and succinctness

- converse (intuition only): "succinctness"
  - suppose L is **P-complete**
  - intuitively, some inputs are "hard" -- require full power of **P**
  - SUCCINCT_L has inputs encoded in different form than L, some exponentially shorter
  - if "hard" inputs are exponentially shorter, then candidate to be **EXP**-complete

13

Succinct encodings

- succinct encoding for a directed graph G= (V = {1,2,3,…}, E):
- a succinct encoding for a variable-free Boolean circuit:



14

An **EXP**-complete problem

- Succinct Circuit Value: given a **succinctly encoded** variable-free Boolean circuit (gates ∧, ∨, ¬, 0, 1), does it output 1?

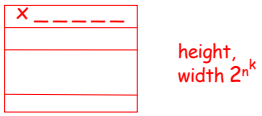
Theorem: Succinct Circuit Value is **EXP**-complete.

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

15

An **EXP**-complete problem

- **tableau** for input $x = x_1x_2x_3\ldots x_n$:

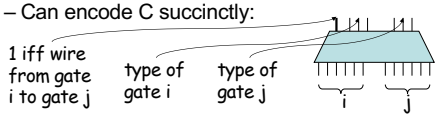


- Circuit C from CVAL reduction has size $O(2^{2n^k})$.
- TM M accepts input x iff circuit outputs 1

16

An **EXP**-complete problem

- Can encode C succinctly:



- 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

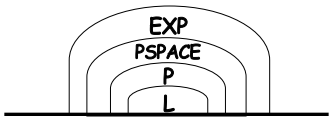
17

Summary

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

18

Summary



19

Nondeterminism: introduction

A motivating question:

- Can computers replace mathematicians?

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

20

Nondeterminism: introduction

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

21

Nondeterminism

- Recall deterministic TM
  - **Q** finite set of states
  - ∑ alphabet including blank: "_"
  - $\mathbf{q_{start}}$, $\mathbf{q_{accept}}$, $\mathbf{q_{reject}}$ in Q
  - transition function:

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

22

Nondeterminism

- **nondeterministic** Turing Machine:
  - **Q** finite set of states
  - ∑ alphabet including blank: "_"
  - $\mathbf{q_{start}}$, $\mathbf{q_{accept}}$, $\mathbf{q_{reject}}$ in Q
  - transition **relation**

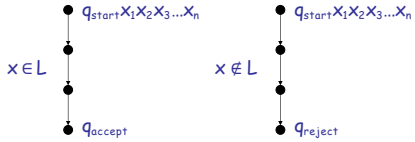
$$\Delta \subseteq (Q \times \Sigma) \times (Q \times \Sigma \times \{L, R, -\})$$

- given current state and symbol scanned, several choices of what to do next.

23

Nondeterminism

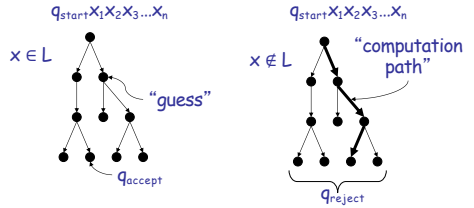
- deterministic TM: given current configuration, unique next configuration



- nondeterministic TM: given current configuration, several possible next configurations

24

Nondeterminism



- asymmetric accept/reject criterion

25

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

26

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 :**N** → **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 :**N** → **N**

27

Nondeterminism

- Focus on time classes first:

$$\mathbf{NP} = \cup_k \mathbf{NTIME}(n^k)$$

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

28

Poly-time verifiers

Very useful alternat "witness" or "certificate" f NP:

Theorem: language L is in NP if/a efficiently verifiable it is expressible as:

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

where R is a language in P.

- poly-time TM M_R deciding R is a "verifier"

29

Poly-time verifiers

- Example: 3SAT expressible as

3SAT = {φ : φ is a 3-CNF formula for which ∃ assignment A for which (φ, A) ∈ R}

R = {(φ, A) : A is a sat. assign. for φ}

  - satisfying assignmdnt A is a "witness" of the satisfiability of φ ("certifies" satisfiability of φ)
  - R is decidable in poly-time

30

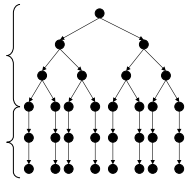
Poly-time verifiers

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

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

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

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



31

Poly-time verifiers

Proof: ($\Leftarrow$) given $L \in NP$, describe L as:

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

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

32

Why NP?

problem requirements

object we are seeking

efficient test: does y meet requirements?

- stic model of
- but, captures important computational feature of many problems:

  exhaustive search w
- contains **huge** number of nat problems
- many problems have form:

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

33

Why NP?

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

34

Relationships between classes

- Easy: **P** $\subseteq$ **NP**, **EXP** $\subseteq$ **NEXP**
  - TM special case of NTM
- Recall: $L \in$ **NP** iff expressible as

  $L = \{ x \mid \exists\, y, |y| \le |x|^k \text{ s.t. } (x,y) \in R \}$
- **NP** $\subseteq$ **PSPACE** (try all possible y)
- The central question:

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

finding a solution vs. recognizing a solution

35

NP-completeness

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

Theorem: Circuit SAT is **NP**-complete.

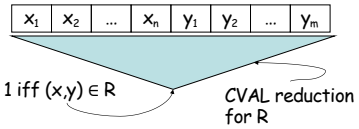
- Proof:
  - clearly in **NP**

36

NP-completeness

– Given L ∈ **NP** of form

L = { x | ∃ y s.t. (x,y) ∈ R}



– hardwire input x; leave y as variables

37

NEXP-completeness

- Succinct Circuit SAT: given a **succinctly encoded** Boolean circuit (gates ∧, ∨, ¬), with variables $y_1, y_2, \ldots, y_m$ 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.

38

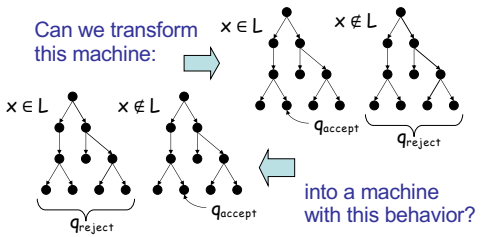
Complement classes

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

39

coNP

- Is NP closed under complement?



40

coNP

- "proof system" interpretation:
- Recall: L ∈ **NP** iff expressible as

L = { x | ∃ y, |y| ≤ $|x|^k$, (x, y) ∈ 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

41

coNP

- **P** = **NP** implies **NP** = **coNP**
- Belief:

NP ≠ coNP

  - another major open problem

42

NTIME Hierarchy Theorem

Theorem (Nondeterministic Time Hierarchy Theorem):

For every *proper complexity function* f(n) ≥ n, and g(n) = ω(f(n+1)),

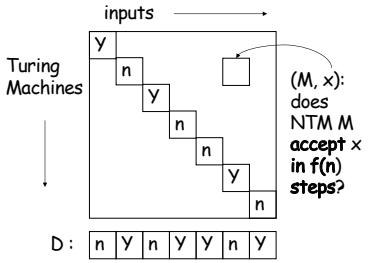
NTIME(f(n)) ⊊ NTIME(g(n)).

43

NTIME Hierarchy Theorem

Proof attempt :

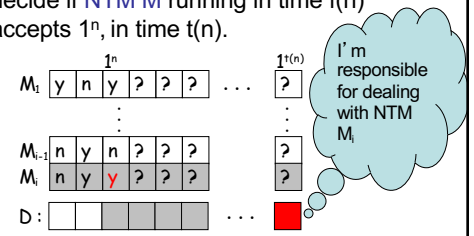
(what's wrong?)



44

NTIME Hierarchy Theorem

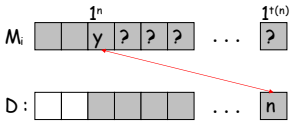
- Let t(n) be large enough so that can decide if NTM M running in time f(n) accepts 1^n, in time t(n).



45

NTIME Hierarchy Theorem

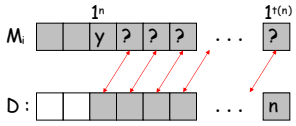
- Enough time on input $1^{t(n)}$ to do the *opposite* of $M_i(1^n)$:



46

NTIME Hierarchy Theorem

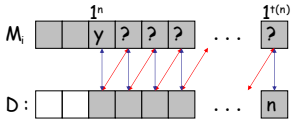
- For k in [n…t(n)] can to do *same* as $M_i(1^{k+1})$ on input 1^k



47

NTIME Hierarchy Theorem

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



  - equality along all arrows.
  - contradiction.

48

NTIME Hierarchy Theorem

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