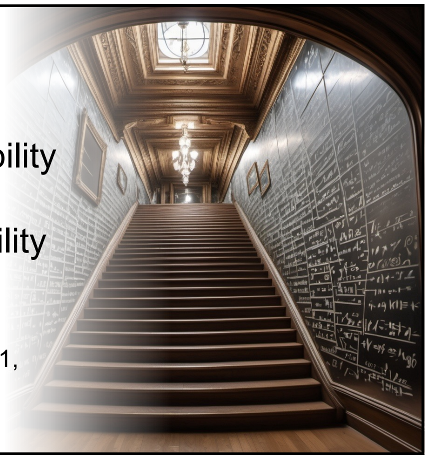


CS21 Decidability and Tractability

Lecture 19

February 21, 2025

1

Grades so far

- An idea of eventual scale:
  - 2025 so far: mean 87.4
  - 2024 mean: 85.5; median 87.0
  - 2023 mean 80.5; median 81.36
  - 2022: mean 80.9; median 83.6
  - 2021: mean 85.7; median 86.9

2024

min	max	grade
97.0	100.0	A+
93.0	97.0	A
88.0	93.0	A-
85.0	88.0	B+
81.0	85.0	B
78.0	81.0	B-
74.5	78.0	C+
70.0	74.5	C
65.0	70.0	C-
60.0	65.0	D+
55.0	60.0	D
0.0	55.0	E/F

2023

min	max	grade
97.0	100.0	A+
92.0	97.0	A
87.0	92.0	A-
84.0	87.0	B+
80.5	84.0	B
76.0	80.5	B-
72.5	76.0	C+
68.0	72.5	C
62.5	68.0	C-
59.0	62.5	D+
52.5	59.0	D
0.0	52.5	E/F

2022

min	max	grade
97.0	100.0	A+
92.0	97.0	A
87.0	92.0	A-
84.0	87.0	B+
80.5	84.0	B
76.0	80.5	B-
72.5	76.0	C+
68.0	72.5	C
62.5	68.0	C-
59.0	62.5	D+
54.0	59.0	D
0.0	54.0	E/F

2021

min	max	grade
97.5	100.0	A+
93.0	97.5	A
88.5	93.0	A-
85.0	88.5	B+
81.5	85.0	B
77.0	81.5	B-
73.0	77.0	C+
69.0	73.0	C
65.0	69.0	C-
60.5	65.0	D+
55.5	60.5	D
0.0	55.5	E/F

2

Hardness and completeness

- Reasonable that can efficiently transform one problem into another.
- Surprising:
  - can often find a special language L so that **every** language in a given complexity class reduces to L!
  - powerful tool

3

Hardness and completeness

- Recall:
  - a language L is a set of strings
  - a complexity class C is a set of languages

Definition: a language L is C-hard if for every language A ∈ C, A poly-time reduces to L; i.e., $A \leq_P L$.

meaning: L is at least as "hard" as anything in C

4

Hardness and completeness

- Recall:
  - a language L is a set of strings
  - a complexity class C is a set of languages

Definition: a language L is C-complete if L is C-hard and L ∈ C

meaning: L is a "hardest" problem in C

5

An EXP-complete problem

- Version of A_{TM} with a time bound:

ATM_B = {<M, x, m> : M is a TM that accepts x within at most m steps}

Theorem: ATM_B is EXP-complete.

Proof:
- what do we need to show?

6

An EXP-complete problem

- ATM_B = {<M, x, m> : M is a TM that accepts x within at most m steps}
- Proof that ATM_B is EXP-complete:
  - Part 1. Need to show $ATM_B \in$ EXP.
    - simulate M on x for m steps; accept if simulation accepts; reject if simulation doesn't accept.
    - running time $m^{O(1)}$.
    - n = length of input $\geq \log_2 m$
    - running time $\leq m^k = 2^{(\log m)k} \leq 2^{(kn)}$

7

An EXP-complete problem

- ATM_B = {<M, x, m> : M is a TM that accepts x within at most m steps}
- Proof that ATM_B is EXP-complete:
  - Part 2. For each language A $\in$ EXP, need to give poly-time reduction from A to ATM_B.
  - for a given language A $\in$ EXP, we know there is a TM M_A that decides A in time $g(n) \leq 2^{n^k}$ for some k.
  - what should reduction f(w) produce?

8

An EXP-complete problem

- ATM_B = {<M, x, m> : M is a TM that accepts x within at most m steps}
- Proof that ATM_B is EXP-complete:
  - f(w) = <M_A, w, m> where $m = 2^{|w|^k}$
  - is f(w) poly-time computable?
    - hardcode M_A and k…
  - YES maps to YES?
    - $w \in A \Rightarrow$ <M_A, w, m> $\in ATM_B$
  - NO maps to NO?
    - $w \notin A \Rightarrow$ <M_A, w, m> $\notin ATM_B$

9

An EXP-complete problem

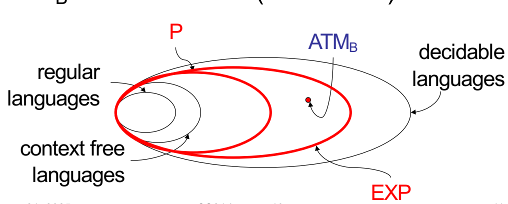
- A C-complete problem is a surrogate for the entire class C.
- For example: if you can find a poly-time algorithm for ATM_B then there is automatically a poly-time algorithm for every problem in EXP (i.e., EXP = P).

- Can you find a poly-time alg for ATM_B?

10

An EXP-complete problem

- Can you find a poly-time alg for ATM_B?
- NO! we showed that P $\subsetneq$ EXP.
- ATM_B is not tractable (intractable).



11

Back to 3SAT

- Remember 3SAT $\in$ EXP
  3SAT = {formulas in CNF with 3 literals per clause for which there exists a satisfying truth assignment}

- It seems hard. Can we show it is intractable?
  - formally, can we show 3SAT is EXP-complete?

12

Back to 3SAT

- can we show 3SAT is EXP-complete?
- Don't know how to. Believed unlikely.
- One reason: there is an important positive feature of 3SAT that doesn't seem to hold for problems in EXP (e.g. ATM_B):

3SAT is decidable in polynomial time by a nondeterministic TM

13

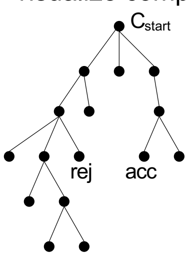
Nondeterministic TMs

- Recall: nondeterministic TM
- informally, TM with several possible next configurations at each step
- formally, A NTM is a 7-tuple
  $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ where:
  - everything is the same as a TM except the transition function:
    $\delta: Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\})$

14

Nondeterministic TMs

visualize computation of a NTM M as a tree



- nodes are configurations
- leaves are accept/reject configurations
- M accepts if and only if there exists an accept leaf
- M is a decider, so no paths go on forever
- running time is max. path length

15

The class NP

Definition: TIME(t(n)) = {L : there exists a TM M that decides L in time O(t(n))}

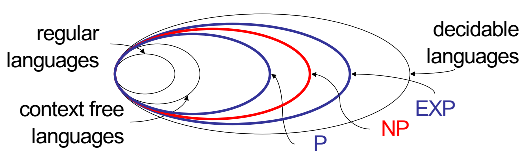
$$P = \cup_{k \geq 1} TIME(n^k)$$

Definition: NTIME(t(n)) = {L : there exists a NTM M that decides L in time O(t(n))}

$$NP = \cup_{k \geq 1} NTIME(n^k)$$

16

NP in relation to P and EXP



- P ⊆ NP (poly-time TM ***is*** a poly-time NTM)
- NP ⊆ EXP
  - configuration tree of n^k-time NTM has $\leq b^{n^k}$ nodes
  - can traverse entire tree in $O(b^{n^k})$ time

we do not know if either inclusion is proper

17

Poly-time verifiers

- NP = {L : L decide … e NTM}

"witness" or "certificate"

- Very useful alternate definition …

efficiently verifiable

Theorem: language L is in NP if … it is expressible as:

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

where R is a language in P.

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

18

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 assignment A is a “witness” of the satisfiability of φ (it “certifies” satisfiability of φ)
  – R is decidable in poly-time

19

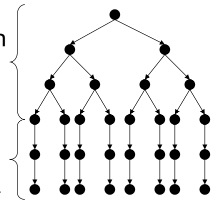
Poly-time verifiers

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

Proof: (⇐) give poly-time NTM deciding L

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

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



20

Poly-time verifiers

Proof: (⇒) given L ∈ 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: M accepts x iff $\exists\, y, |y| \le |x|^k, (x, y) \in R$

21

Cook-Levin Theorem

- Gateway to proving lots of natural, important problems NP-complete is:

Theorem (Cook, Levin): 3SAT is NP-complete.

- Recall: 3SAT = {φ : φ is a CNF formula with 3 literals per clause for which there exists a satisfying truth assignment}

22

Cook-Levin Theorem

- Proof outline
  – show CIRCUIT-SAT is NP-complete

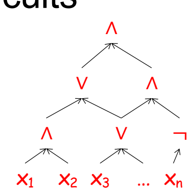
  CIRCUIT-SAT = {C : C is a Boolean circuit for which there exists a satisfying truth assignment}

  – show 3SAT is NP-complete (reduce from CIRCUIT SAT)

23

Boolean Circuits

- Boolean circuit C
  – directed acyclic graph
  – nodes: AND (∧); OR (∨); NOT (¬); variables x_i



- C computes function $f:\{0,1\}^n \to \{0,1\}$ in natural way
  – identify C with function f it computes
- size = # nodes

24

Boolean Circuits

- every function $f:\{0,1\}^n \rightarrow \{0,1\}$ computable by a circuit of size at most $O(n2^n)$
  - AND of n literals for each x such that $f(x) = 1$
  - OR of up to 2^n such terms

CIRCUIT-SAT is NP-complete

Theorem: CIRCUIT-SAT is NP-complete

CIRCUIT-SAT = {C : C is a Boolean circuit for which there exists a satisfying truth assignment}

Proof:

- Part 1: need to show CIRCUIT-SAT ∈ NP.
  - can express CIRCUIT-SAT as:

CIRCUIT-SAT = {C : C is a Boolean circuit for which ∃x such that (C, x) ∈ R}

R = {(C, x) : C is a Boolean circuit and C(x) = 1}

CIRCUIT-SAT is NP-complete

CIRCUIT-SAT = {C : C is a Boolean circuit for which there exists a satisfying truth assignment}

Proof:

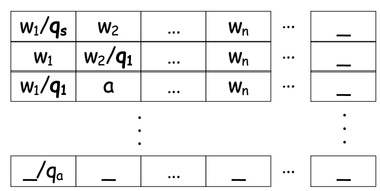
- Part 2: for each language A ∈ NP, need to give poly-time reduction from A to CIRCUIT-SAT
- for a given language A ∈ NP, we know

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

and there is a (deterministic) TM M_R that decides R in time $g(n) \le n^c$ for some c.

CIRCUIT-SAT is NP-complete

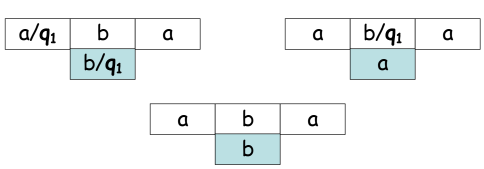
- Tableau (configurations written in an array) for machine M_R on input w = (x, y):



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

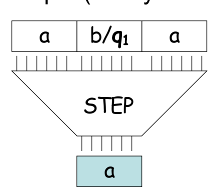
CIRCUIT-SAT is NP-complete

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



CIRCUIT-SAT is NP-complete

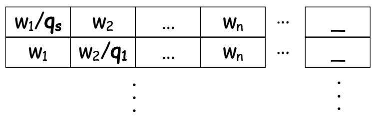
- Can build Boolean circuit STEP
  - input (binary encoding of) 3 cells
  - output (binary encoding of) 1 cell



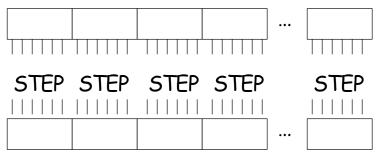
- each output bit is some function of inputs
- can build circuit for each
- size is independent of size of tableau

CIRCUIT-SAT is NP-complete

Tableau for M_R on input $w = (x, y)$

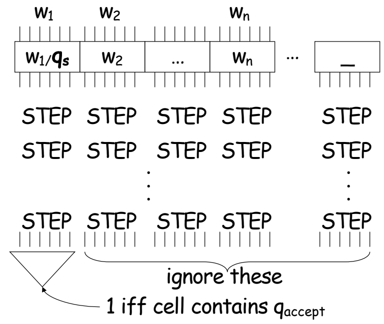


- $|w|^c$ copies of STEP compute row i from i-1



31

CIRCUIT-SAT is NP-complete



This circuit $C_{M_R, w}$ has inputs $w_1w_2\ldots w_n$

and $C(w) = 1$ iff M_R accepts input w.

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

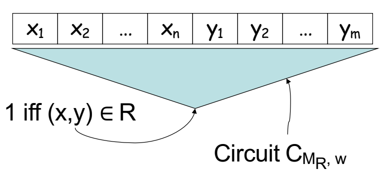
32

CIRCUIT-SAT is NP-complete

- recall: we are reducing language A:

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

  to CIRCUIT-SAT.
- f(x) produces the following circuit:



- hardwire input x
- leave y as variables

33

CIRCUIT-SAT is NP-complete

- is f(x) poly-time computable?
  - hardcode M_R, k and c
  - circuit has size $O(|w|^{2c})$; $|w| = |(x,y)| \leq n + n^k$
  - each component easy to describe efficiently from description of M_R
- YES maps to YES?
  - $x \in A \Rightarrow \exists$ y, M_R accepts (x, y) $\Rightarrow f(x) \in$ CIRCUIT-SAT
- NO maps to NO?
  - $x \notin A \Rightarrow \forall$ y, M_R rejects (x, y) $\Rightarrow f(x) \notin$ CIRCUIT-SAT

34

3SAT is NP-complete

Theorem: 3SAT is NP-complete

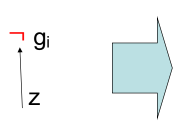
3SAT = {φ : φ is a 3-CNF formula for which there exists a satisfying truth assignment}

Proof:
- Part 1: need to show 3-SAT ∈ NP
  - already done
- Part 2: need to show 3-SAT is NP-hard
  - we will give a poly-time reduction from CIRCUIT-SAT to 3-SAT

35

3SAT is NP-complete

- given a circuit C
  - variables $x_1, x_2, \ldots, x_n$
  - AND (∧), OR (∨), NOT (¬) gates $g_1, g_2, \ldots, g_m$
- reduction f(C) produces these clauses for φ on variables $x_1, x_2, \ldots, x_n, g_1, g_2, \ldots, g_m$:



- $(g_i \lor z)$
- $(\neg z \lor \neg g_i)$

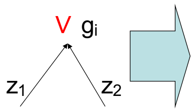
$(z \Leftrightarrow \neg g_i)$

36

3SAT is NP-complete

– given a circuit C
  - variables $x_1, x_2, \ldots, x_n$
  - AND (∧), OR (∨), NOT (¬) gates $g_1, g_2, \ldots, g_m$

– reduction f(C) produces these clauses for φ on variables $x_1, x_2, \ldots, x_n, g_1, g_2, \ldots, g_m$:



- $(\neg z_1 \vee g_i)$
- $(\neg z_2 \vee g_i)$
- $(\neg g_i \vee z_1 \vee z_2)$

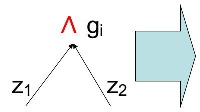
$\}$ $(z_1 \vee z_2 \Leftrightarrow g_i)$

37

3SAT is NP-complete

– given a circuit C
  - variables $x_1, x_2, \ldots, x_n$
  - AND (∧), OR (∨), NOT (¬) gates $g_1, g_2, \ldots, g_m$

– reduction f(C) produces these clauses for φ on variables $x_1, x_2, \ldots, x_n, g_1, g_2, \ldots, g_m$:



- $(\neg g_i \vee z_1)$
- $(\neg g_i \vee z_2)$
- $(\neg z_1 \vee \neg z_2 \vee g_i)$

$\}$ $(z_1 \wedge z_2 \Leftrightarrow g_i)$

38

3SAT is NP-complete

– finally, reduction f(C) produces single clause (g_m) where g_m is the output gate.

– f(C) computable in poly-time?
  - yes, simple transformation

– YES maps to YES?
  - if C(x) = 1, then assigning x-values to x-variables of φ and gate values of C when evaluating x to the g-variables of φ gives satsifying assignment.

39

3SAT is NP-complete

– NO maps to NO?
  - show that φ satisfiable implies C satisfiable
  - satisfying assignment to φ assigns values to x-variables and g-variables
  - output gate g_m must be assigned 1
  - every other gate must be assigned value it would take given values of its inputs.
  - the assignment to the x-variables must be a satisfying assignment for C.

40