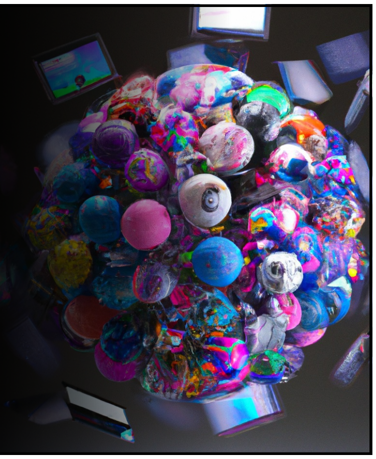
# CS21 Decidability and Tractability

Lecture 19
February 16, 2024





## Grades so far

- An idea of eventual scale:
  - 2024 so far: 80.0; 84.82
  - 2023 mean 80.5; median 81.36
  - 2022: mean 80.9; median 83.6
  - 2021: mean 85.7; median 86.9
  - 2020: mean 81.3; median 81.8

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 |

2020

| | | |
|---|---|---|
| 97.5 | 100 | A+ |
| 89.5 | 97.5 | A |
| 86.5 | 89.5 | A- |
| 83.5 | 86.5 | B+ |
| 78.5 | 83.5 | B |
| 73.5 | 78.5 | B- |
| 69.5 | 73.5 | C+ |
| 65.5 | 69.5 | C |
| 61 | 65.5 | C- |
| 56.5 | 61 | D+ |
| 51.5 | 56.5 | D |
| 0 | 51.5 | E/F |





## The class NP

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

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

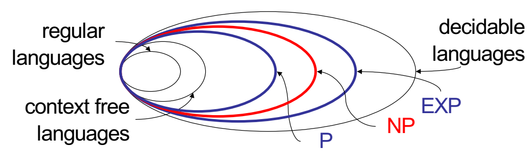
**Definition**: NTIME(t(n)) = {L : there exists a NTM M that decides L in time O(t(n))}

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





## 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 ≤ $b^{n^k}$ nodes
  - can traverse entire tree in $O(b^{n^k})$ time

**we do not know if either inclusion is proper**





## Poly-time verifiers

- NP = {L : L decided by poly-time NTM}
- Very useful alternate definition of NP:

**Theorem**: language L is in NP if and only 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.

"witness" or "certificate"

efficiently verifiable

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





## 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

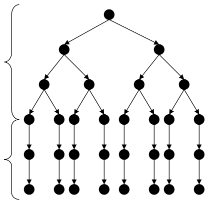
## Poly-time verifiers

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

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

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

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





7

## Poly-time verifiers

**Proof**: (⇒) 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) : y is an accepting computation history of M on input x}
- check: accepting history has length $\leq |x|^k$
- check: M accepts x iff $\exists\, y, |y| \leq |x|^k, (x, y) \in R$



8

## 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}



9

## Cook-Levin Theorem

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

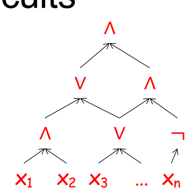
    CIRCUIT-SAT = {C : C is a Boolean circuit for which there exists a satisfying truth assignment}
  - show 3SAT is NP-complete (reduce from CIRCUIT SAT)



10

## Boolean Circuits

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




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



11

## 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



12

# 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 $\in$ NP.
  - can express CIRCUIT-SAT as:

CIRCUIT-SAT = {C : C is a Boolean circuit for which $\exists$x such that (C, x) $\in$ R}

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



13

# 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 $\in$ NP, need to give poly-time reduction from A to CIRCUIT-SAT
- for a given language A $\in$ NP, we know

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

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



14

# CIRCUIT-SAT is NP-complete

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

| | | | | | |
|---|---|---|---|---|---|
| $w_1/q_s$ | $w_2$ | … | $w_n$ | … | _ |
| $w_1$ | $w_2/q_1$ | … | $w_n$ | … | _ |
| $w_1/q_1$ | a | … | $w_n$ | … | _ |
| | ⋮ | | | | ⋮ |
| $\_/q_a$ | _ | … | _ | … | _ |

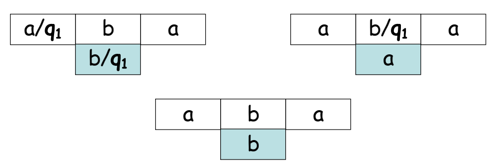
- height = time taken = $|w|^c$
- width = space used $\leq |w|^c$



15

# CIRCUIT-SAT is NP-complete

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

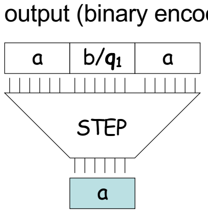





16

# 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



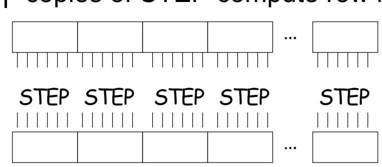
17

# CIRCUIT-SAT is NP-complete

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

| | | | | | |
|---|---|---|---|---|---|
| $w_1/q_s$ | $w_2$ | … | $w_n$ | … | _ |
| $w_1$ | $w_2/q_1$ | … | $w_n$ | … | _ |
| | ⋮ | | | | ⋮ |

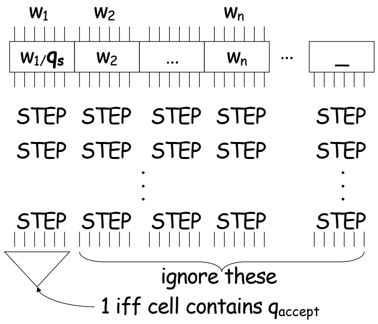
- $|w|^c$ copies of STEP compute row i from i-1






18

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





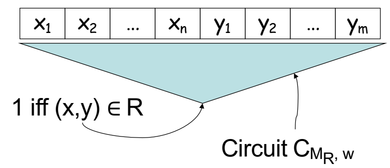
## CIRCUIT-SAT is NP-complete

– recall: we are reducing language A:

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

to CIRCUIT-SAT.

– f(x) produces the following circuit:




– hardwire input x

– leave y as variables





## CIRCUIT-SAT is NP-complete

– is f(x) poly-time computable?
  - hardcode $M_R$, k and c
  - circuit has size $O(|w|^{2c})$; $|w| = |(x,y)| \le n + n^k$
  - 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





## 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

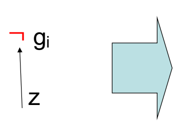




## 3SAT is NP-complete

– 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 $x_1, x_2, \ldots, x_n, g_1, g_2, \ldots, g_m$:




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

$(z \Leftrightarrow \neg g_i)$

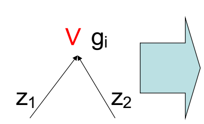




## 3SAT is NP-complete

– 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 $x_1, x_2, \ldots, x_n, g_1, g_2, \ldots, g_m$:

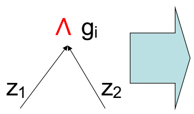



- $(\neg z_1 \lor g_i)$
- $(\neg z_2 \lor g_i)$
- $(\neg g_i \lor z_1 \lor z_2)$

$(z_1 \lor z_2 \Leftrightarrow g_i)$

## 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 $x_1, x_2, \ldots, x_n, g_1, g_2, \ldots, g_m$:




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

$(z_1 \land z_2 \Leftrightarrow g_i)$



25

## 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.



26

## 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.



27

## Search vs. Decision

- Definition: given a graph G = (V, E), an independent set in G is a subset V'⊆ V such that for all u,w ∈ V' (u,w) ∉ E
- A problem:

  given G, find the largest independent set
- This is called a search problem
  - searching for *optimal* object of some type
  - comes up frequently



28

## Search vs. Decision

- We want to talk about languages (or decision problems)
- Most search problems have a natural, related decision problem by adding a bound "k"; for example:
  - search problem: given G, find the largest independent set
  - decision problem: given (G, k), is there an independent set of size *at least* k



29