CS21
Decidability and Tractability
Lecture 20
February 18, 2022

Outline

- 3-SAT is NP-complete
- NP-complete problems: independent set, vertex cover, clique
- NP-complete problems: Hamilton path and cycle, Traveling Salesperson Problem
- NP-complete problems: Subset Sum

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\}

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 \exists 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 | \exists y, |y| ≤ |x|^c, (x, y) ∈ R\}
  - and there is a (deterministic) TM M_R that decides R in time g(n) ≤ n^c for some c.
CIRCUIT-SAT is NP-complete

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

$$
\begin{array}{ccccccc}
W_1/a_{q_1} & W_2 & \ldots & W_n & \ldots & \ldots & \ldots \\
W_1 & W_2/a_{q_1} & \ldots & W_n & \ldots & \ldots & \ldots \\
W_1 & a & \ldots & W_n & \ldots & \ldots & \ldots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
W_1 & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
\end{array}
$$

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

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

$$
\begin{array}{cccc}
a/q_1 & b & a & a \\
& b/q_1 & a \\
& a & b & a \\
& b & \\
\end{array}
$$

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

  $$
  \begin{array}{c}
a \\
b/q_1 \downarrow \text{STEP} \\
a \end{array}
  $$

  - 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

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

CIRCUIT-SAT is NP-complete

- Recall: we are reducing language $A$:
  $$
  A = \{ x \mid \exists y, |y| \leq |x|^c, (x, y) \in R \}
  $$
  to CIRCUIT-SAT.

  - $f(x)$ produces the following circuit:

    $$
    \begin{array}{cccccccc}
x_1 & x_2 & \ldots & x_n & y_1 & y_2 & \ldots & y_m \\
    \end{array}
    $$

    - hardwire input $x$
    - leave $y$ as variables

  - Circuit $C_{M_R, w}$ has inputs $w_1, w_2, \ldots, w_n$ and $C(w) = 1$ iff $M_R$ accepts input $w$.

  Size = $O(|w|^{2c})$
CIRCUIT-SAT is NP-complete

– is $f(x)$ poly-time computable?
  • hardcode $M_R$, $k$ and $c$
  • circuit has size $O(|w|^2)$; $|w| = |(x,y)| \leq n + n^k$
  • each component easy to describe efficiently from description of $M_R$

– YES maps to YES?
  • $x \in A \Rightarrow \exists y, M_R$ accepts $(x,y) \Rightarrow f(x) \in \text{CIRCUIT-SAT}$

– NO maps to NO?
  • $x \notin A \Rightarrow \forall y, M_R$ rejects $(x,y) \Rightarrow f(x) \notin \text{CIRCUIT-SAT}$

Cook-Levin Theorem

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

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

• Recall: $3SAT = \{\phi : \phi$ is a CNF formula with 3 literals per clause for which there exists a satisfying truth assignment$\}$

Proof:

– Part 1: need to show $3SAT \in \text{NP}$
  • already done

– Part 2: need to show $3SAT$ is NP-hard
  • we will give a poly-time reduction from CIRCUIT-SAT to 3SAT

3SAT is NP-complete

Theorem: 3SAT is NP-complete

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

Proof:

– Part 1: need to show 3-SAT $\in \text{NP}$
  • 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

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)
3SAT is NP-complete

- given a circuit $C$
  - variables $x_1, x_2, \ldots, x_n$
  - AND ($\land$), OR ($\lor$), NOT ($\neg$) gates $g_1, g_2, \ldots, g_m$
- reduction $f(C)$ produces these clauses for $\phi$
  on variables $x_1, x_2, \ldots, x_n, g_1, g_2, \ldots, g_m$:

\[
\begin{align*}
  (\neg z_1 \lor g_i) \\
  (\neg z_2 \lor g_i) \\
  (\neg g_i \lor z_1 \lor z_2) \\
  (z_1 \land z_2 \iff g_i)
\end{align*}
\]

- 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 $\phi$ and gate values of $C$ when evaluating $x$ to the $g$-variables of $\phi$ gives satisfying assignment.

Search vs. Decision

- Definition: given a graph $G = (V, E)$, an independent set in $G$ is a subset $V' \subseteq V$ such that for all $u, w \in V'$, $(u, w) \notin E$
  - 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

- 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 $k$
**Theorem:** the following language is NP-complete:

\[ IS = \{(G, k) : G \text{ has an IS of size } \geq k\}. \]

- Part 1: IS ∈ NP. Proof?
- Part 2: IS is NP-hard.
  * reduce from 3-SAT

**Ind. Set is NP-complete**

The reduction f: given

\[ \phi = (x \lor y \lor \neg z) \land (\neg x \lor w \lor z) \land \ldots \land (...) \]

we produce graph \( G_\phi \):

- one triangle for each of \( m \) clauses
- edge between every pair of contradictory literals
- set \( k = m \)

**Ind. Set is NP-complete**

\[ \phi = (x \lor y \lor \neg z) \land (\neg x \lor w \lor z) \land \ldots \land (...) \]

\[ f(\phi) = (G, \# \text{ clauses}) \]

- Is \( f \) poly-time computable?
- YES maps to YES?
  - 1 true literal per clause in satisfying assign. \( A \)
  - choose corresponding vertices (1 per triangle)
  - IS, since no contradictory literals in \( A \)

- NO maps to NO?
  - IS can have at most 1 vertex per triangle
  - IS of size \( \geq \# \) clauses must have exactly 1 per
  - since IS, no contradictory vertices
  - can produce satisfying assignment by setting these literals to true