Formal Languages and Automata, 2024 Fall semester

Lec 21. NP-completeness

Eunjung Kim

Polynomial-time many-one reduction

Let $A \subseteq \{0,1\}^*$ and $B \subseteq \{0,1\}^*$ be two languages, i.e. two (decision) problems

P-time many-one reduction

We say that A is polynomial-time many-one reducible to B, written as $A \leq_p B$, if there is a polynomial-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ such that for every input $w \in \{0,1\}^*$,

$$w \in A \text{ if and only if } f(w) \in B.$$

Also called polynomial-time mapping reduction, polynomial-time Karp reduction and polynomial-time transformation.

NP-hard, NP-complete

Definitions

A language L is said to be NP-hard if $A \leq_p L$ for every language $A \in$ NP.

A language L is said to be NP-complete if L is NP-hard and $L \in$ NP.

$A \leq_p B$ means B is as hard as A

Transitivity

If $A \leq_p B$ and $B \leq_p C$, then $A \leq_p C$.

NP-hardness propagates forwards

If $A \leq_p B$ and A is NP-hard, then B is NP-hard.

P-time for one NP-complete problem

Let L be NP-complete. Then $L \in \mathrm{P}$ if and only if $\mathrm{P} = \mathrm{NP}$.

First NP-complete problem

Cook-Levin Theorem

Satisfiability is NP-complete.

Boolean Satisfiability

- Boolean Formula: an expression formed from boolean variables using logical connectives $\wedge$ ('AND'), $\vee$'s ('OR) and the negation $\neg$.
  $(x_1 \vee \bar{x_3}) \wedge (\neg(x_2 \wedge x_4) \vee ((\bar{x_1} \vee x_2) \wedge \bar{x_5})) \wedge (\neg(\bar{x_1} \wedge \bar{x_3}) \vee (\bar{x_4} \wedge x_5))$.
- Rather formally: a single boolean variable is an atomic boolean formula. A negation of a boolean formula, OR of two boolean formulas, AND of two boolean formulas are boolean formulas.
- For a boolean formula φ, one can evaluate φ on an assignment $\gamma : V(\varphi) \to \{0, 1\}$, i.e. determine if φ is satisfied (=1) or not (=0) by γ.
- A formula φ is satisfiable if there is an assignment γ satisfying φ.

Boolean Satisfiability, Satisfiability in short

Input: a boolean formula φ

Question: is there a satisfying assignment to the variables of φ?

Satisfiability is NP-complete

Cook-Levin Theorem

Satisfiability is NP-complete.

- Let $L \subseteq \{0,1\}^*$ be an arbitrary language in NP.
- Equivalently, there exist a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a P-time TM M s.t. for every $x \in \{0,1\}^n$

$$x \in L \text{ if and only if}$$
$$\text{there exists } w \in \{0,1\}^{p(n)} \text{ for which } M(x,w) = 1.$$

- Goal: show $L \leq_p$ Satisfiability.
- How: for every $x \in \{0,1\}^n$, construct a boolean formula φ_x s.t.

  there exists a witness w of length $p(n)$ if and only if
  there exists a satisfying assignment to φ_x

  in time polynomial in n.

Proof of Cook-Levin Theorem

Cook-Levin Theorem

Satisfiability is NP-complete.

- Let $L \subseteq \{0,1\}^*$ be an arbitrary language in NP.
- Equivalently, there exist a single-tape nondeterministic TM M and a polynomial $p : \mathbb{N} \to \mathbb{N}$ and s.t. for every $x \in \{0,1\}^n$

  $x \in L$ if and only if there is an accepting computation history of M on x, and M halts on x in $p(n)$ steps.
- Goal: show $L \leq_p$ Satisfiability.
- How: for every $x \in \{0,1\}^n$, construct a boolean formula φ_x s.t.

  M, upon input string x, admits an accepting computation history of length $p(n)$
  if and only if there exists a satisfying assignment to φ_x.

Proof of Cook-Levin Theorem

- For simplicity, assume $p(n) \leq n^k$ for some k.
- NTM N would not move the header beyond the first n^k cells in the tape.
- An accepting computation history can be written in a $n^k \times n^k$ table.

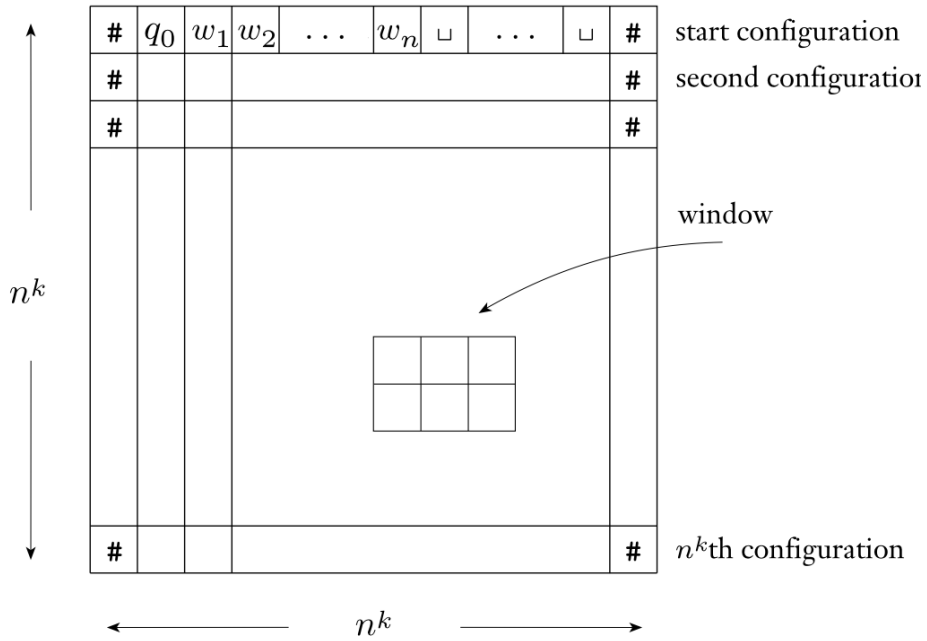

Figure 7.38, Sipser 2012.

Proof of Cook-Levin Theorem

Idea similar to the one used for a reduction from A_{TM} to PCP.

$$\varphi_x = \varphi_{cell} \wedge \varphi_{start} \wedge \varphi_{move} \wedge \varphi_{accept}.$$

- Each cell can contain a symbol from $Q \cup \Gamma$ (from N).
- We allocate a variable $z_{i,j,s}$ to each cell (i, j) (row i and column j) and $s \in Q \cup \Gamma$.
- We want to simulate an accepting computation history of M on a length-n input string x by φ_x.
- More specifically, we express the constraints using a boolean formula φ_x so that
  - an accepting computation $n^k \times n^k$ table starting with $q_0 x B B \cdots B$ must satisfy the constraint,
  - any satisfying assignment to φ_x , when written on the cells of the table, forms an accepting computation history.

Proof of Cook-Levin Theorem

$$\varphi_x = \varphi_{cell} \wedge \varphi_{start} \wedge \varphi_{move} \wedge \varphi_{accept}.$$

- φ_{cell} expresses the constraint
  "each cell contains a single entry from either Q or Γ"
- φ_{start} expresses the constraint
  "the first row is $\#q_0 xBB \cdots B\#$."
- φ_{accept} expresses the constraint
  "the last row contains $q_{accept} \in Q$."
- φ_{move} expresses the constraint
  "every (i,j)-th 2×3-window is a part of legit move in δ."

Proof of Cook-Levin Theorem

$$\varphi_x = \varphi_{cell} \wedge \varphi_{start} \wedge \varphi_{move} \wedge \varphi_{accept}.$$

- φ_{cell} expresses the constraint
  "each cell contains a single entry from either Q or Γ"

Proof of Cook-Levin Theorem

$$\varphi_x = \varphi_{cell} \wedge \varphi_{start} \wedge \varphi_{move} \wedge \varphi_{accept}.$$

- φ_{start} expresses the constraint
  "the first row is $\#q_0xBB\cdots B\#$*."*

Proof of Cook-Levin Theorem

$$\varphi_x = \varphi_{cell} \wedge \varphi_{start} \wedge \varphi_{move} \wedge \varphi_{accept}.$$

- φ_{accept} expresses the constraint

  "the last row contains $q_{accept} \in Q$."

Proof of Cook-Levin Theorem

$$\varphi_x = \varphi_{cell} \wedge \varphi_{start} \wedge \varphi_{move} \wedge \varphi_{accept}.$$

- φ_{move} expresses the constraint
  "every (i, j)-th 2×3-window is a part of legal move in δ."
- Legal and illegal moves when $\delta(q_1, a) = \{(q_1, b, R)\}$ and $\delta(q_1, b) = \{(q_2, c, L), (q_2, a, R)\}$.

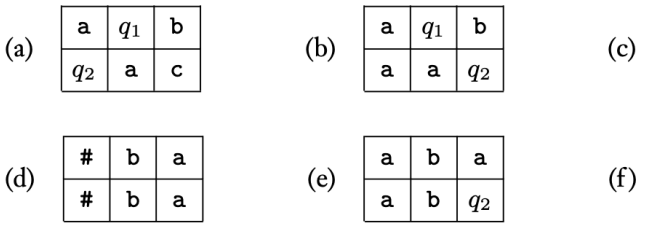

Figure 7.39, Sipser 2012.

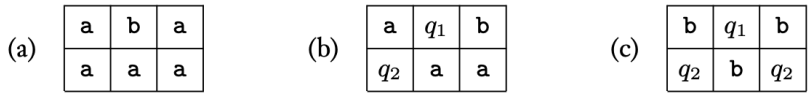

Figure 7.40, Sipser 2012.

Proof of Cook-Levin Theorem

$$\varphi_x = \varphi_{cell} \wedge \varphi_{start} \wedge \varphi_{move} \wedge \varphi_{accept}.$$

- φ_{move} expresses the constraint

  "Each (i,j)-th 2×3-window is a part of legal move in δ"

  or equivalently,

$$\varphi_{move} = \bigwedge_{1 \le i \le n^k, 1 \le j \le n^k} (i,j)\text{-th window is legal}$$

- Legal windows

#	a	b	q_1	b	c	a	#
#	a	b	c	q_2	c	a	#

Proof of Cook-Levin Theorem

We described a construction of a boolean formula φ_x from an input string $x \in \Sigma^*$ to L.

It remains to prove that this is a polynomial-time transformation from $L \in$ NP to SATISFIABILITY.

- The construction can be done in polynomial time in $|x|$ (tedious to check).
- ($\Rightarrow$) If $x \in L$, then consider the $n^k \times n^k$ table displaying an accepting computation history.
- Choose the assignment to $(z_{i,j,s})_{1 \leq i,j \leq n^k, s \in Q \cup \Gamma}$ which encodes this table.
- Easy to see that each of the subformulas φ_{cell}, φ_{start}, φ_{accept} and φ_{move} is satisfied.

Proof of Cook-Levin Theorem

($\Leftarrow$) Suppose that φ_x is satisfied by an assignment $(\tilde{z}_{i,j,s})_{1 \le i,j \le n^k, s \in Q \cup \Gamma}$.

1. it encodes a $n^k \times n^k$ table, where each cell contains a single symbol from $Q \cup \Gamma$; imposed by φ_{cell}
2. The first row must be $\#q_0 xBB \cdots B\#$; imposed by φ_{start}.
3. The last row must contain q_{accept}; imposed by φ_{accept}.

Proof of Cook-Levin Theorem

($\Leftarrow$) Suppose that φ_x is satisfied by an assignment $(\tilde{z}_{i,j,s})_{1 \le i,j \le n^k, s \in Q \cup \Gamma}$.

1. it encodes a $n^k \times n^k$ table, where each cell contains a single symbol from $Q \cup \Gamma$; imposed by φ_{cell}
2. The first row must be $\#q_0 xBB \cdots B\#$; imposed by φ_{start}.
3. The last row must contain q_{accept}; imposed by φ_{accept}.

To see that $C_i \vdash_M C_{i+1}$ for rows C_i and C_{i+1}

- Obs 1: if C_i contains a single symbol from Q, then C_{i+1} contains a single symbol from Q, in a position captured by some 2×3-window.
- Obs 2: the 2×3-window with q in the top center must describe a legal move.

CNF-SAT

- CNF (Conjunctive Normal Form) formula: $\wedge$ ('AND') of $\vee$'s ('OR) of variables and their negations.
  $(x_1 \vee \bar{x_3}) \wedge (x_2 \vee x_4) \wedge (\bar{x_1} \vee x_2 \vee \bar{x_5}) \wedge (\bar{x_1} \vee \bar{x_3} \vee \bar{x_4} \vee x_5)$.
- Literals are variables and its negations.
- A clause is a OR's over literals.
- CNF formula is a conjunction ('AND') of clauses.
- A formula φ is kCNF if each clause contains at most k literals.

CNF-SAT

INPUT: a CNF formula φ

QUESTION: is there a satisfying assignment to the variables of φ?