COMPSCI 501: Formal Language Theory

Lecture 23: NP-complete Problems

Marius Minea
marius@cs.umass.edu

University of Massachusetts Amherst

22 March 2019

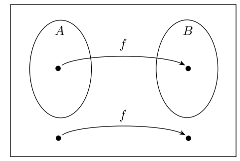
Review: Polynomial-time Reductions

Def. Language A is **polynomial-time (mapping) reducible** to language B ($A \leq_{\mathrm{P}} B$) if a polynomial-time computable function $f : \Sigma^* \to \Sigma^*$ exists, where for all w,

$$w \in A \Leftrightarrow f(w) \in B$$

Use membership testing (solution) for B to decide A *efficiently*

- Reduction (function) goes one-way
  (construct B-problem from A)
- Equivalence proof goes **both** ways
  YES maps to YES
  also need NO mapped to NO



Theorem. If $A \leq_{\mathrm{P}} B$ and $B \in \mathrm{P}$, then $A \in \mathrm{P}$

SAT and 3SAT

SAT: Given a propositional formula, is it satisfiable?

Useful to write formula in *conjunctive normal form*:
conjunction of *clauses* $(x_i \vee x_j \vee \ldots \vee \overline{x_k}) \wedge \ldots \wedge (\ldots)$
each clause: disjunction of *literals* (variable or negation)

Can we polynomially reduce *SAT* $\leq_{\mathrm{P}}$ *CNF-SAT* ?
CNF conversion exponential. Polynomial reduction possible.

k-SAT: at most (some defs: exactly) k literals per clause
in particular: *3SAT* (exactly 3 literals per clause)

How about *2SAT* ?

$(x_1 \vee x_3) \wedge (\overline{x_1} \vee x_2) \wedge \ldots \wedge (x_4 \vee \overline{x_2})$

2SAT is polynomial-time

3SAT $\leq_{\mathrm{P}}$ *CLIQUE*

Construct an instance of *CLIQUE* from a formula
for k clauses, construct $\langle G, k \rangle$ (*k-CLIQUE*)

Construction: k groups of 3 nodes, one group for each clause.

- *no connection* between nodes from one clause
  can have *k-CLIQUE* only by choosing one node per clause
  equivalent to satisfying each clause
- *no connection* between any x_i and $\overline{x_i}$
  cannot choose x_i and $\overline{x_i}$ at the same time
  equivalent to maintaining *consistency*
- connect *all other* nodes

NP-completeness

Def. A *language* B is NP-complete iff
1. B is in NP
2. for any A in NP, we have $A \leq_{\mathrm{P}} B$

(2) means B is *NP-hard* (at least as hard as any problem in NP);
B itself need not be in NP

In NP (1) + NP-hard (2) means NP-complete.

- If B is NP-complete and $B \leq_{\mathrm{P}} C$, then C is NP-complete
  reduce *known* NP-complete problem B to target C
  reduce target problem C *from* NP-complete problem B
- If B is NP-complete and $B \in \mathrm{P}$, then P = NP

All NP-complete problems are polynomially reducible to one another
(the hardest problems in NP)

Cook-Levin Theorem: *SAT* is NP-complete

SAT: the prototypical NP-complete problem

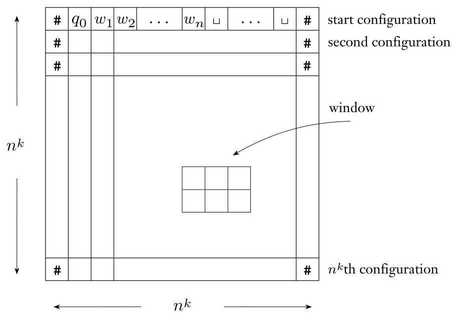
Stephen Cook, 1971: proof
Richard Karp, 1972: 21 NP-complete problems
Leonid Levin (USSR), 1970s: 6 "universal search problems"

SAT clearly in NP: guess satisfying assignment, verify in poly-time

Proof idea:
Reduce any polynomial time NTM decider to a SAT problem.

Concretely: let $A \in \mathrm{NP}$ and N a NTM that decides A in time n^k.

A Tableau of Configurations



Since time n^k, tape length at most n^k

Encoding Computation History as *SAT*

Each cell is a variable: $x_{i,j,s} = 1$ iff $cell[i, j] = s$
$s \in Q \cup \Gamma \cup \{\#\}$ (state or symbol)

Expressing the constraints

- One true variable per cell (unique cell contents) $O(n^{2k})$
- Encode starting configuration $O(n^k)$
- Encode valid moves (transition relation)
  using 3x2 *windows* $O(n^{2k})$ constant size per cell
- Some state in some line is accepting $O(n^{2k})$

Claim: If top row is start configuration, and every window is legal, then every row of the tableau is a configuration that legally follows the preceding one.

This shows we have indeed achieved a reduction.

3SAT is NP-complete

Proof options

1) Do nondeterministic TM conversion directly to *3SAT*
   formula "almost" in CNF, except "windows" (constant size)

2) Converting arbitrary formula to *3SAT* (Tseitin transform)
   new variable for each subformula

$x \leftrightarrow \neg A \qquad (\neg x \vee \neg A) \wedge (x \vee A)$

$x \leftrightarrow A \vee B \qquad (x \rightarrow A \vee B) \wedge (A \vee B \rightarrow x)$
$= (\neg x \vee A \vee B) \wedge (\neg A \vee x) \wedge (\neg B \vee x)$

$x \leftrightarrow A \wedge B \qquad (x \rightarrow A \wedge B) \wedge (A \wedge B \rightarrow x)$
$= (\neg x \vee A) \wedge (\neg x \vee B) \wedge (\neg A \vee \neg B \vee x)$

In both cases, convert clause with n literals to three:

$x_1 \vee \ldots \vee x_n$ *equisatifiable* with
$(x_1 \vee x_2 \vee z_1) \wedge (\overline{z_1} \vee x_3 \vee z_2) \wedge \ldots \wedge (\overline{z_{n-3}} \vee x_{n-1} \vee x_n)$