Advanced Topics in Theoretical Computer Science

Part 5: Complexity (Part 3)

31.01.2018

Viorica Sofronie-Stokkermans

Universität Koblenz-Landau

e-mail: sofronie@uni-koblenz.de

Info

Sample exams and solutions can be found on the website of the exercise.

Question/Answer Session

preferences: Friday, 23.02 or Monday, 26.02 (in the morning)

doodle in the next days.

Contents

- Recall: Turing machines and Turing computability
- Register machines (LOOP, WHILE, GOTO)
- Recursive functions
- The Church-Turing Thesis
- Computability and (Un-)decidability
- Complexity

Complexity classes

How do we show that a certain problem is in a certain complexity class?

Reduction to a known problem

We need one problem we can start with! (for NP: SAT)

Reduction

Definition (Polynomial time reducibility)

Let L_1, L_2 be languages.
L_2 is polynomial time reducible to L_1 (notation: $L_2 \preceq_{\text{pol}} L_1$)
if there exists a polynomial time bounded DTM, which for every input w computes an output $f(w)$ such that

$$w \in L_2 \text{ if and only if } f(w) \in L_1$$

Lemma (Polynomial time reduction)

- Let L_2 be polynomial time reducible to L_1 ($L_2 \preceq_{\text{pol}} L_1$). Then:
  If $L_1 \in NP$ then $L_2 \in NP$.
  If $L_1 \in P$ then $L_2 \in P$.
- The composition of two polynomial time reductions is again a polynomial time reduction.

Complete and hard problems

Definition (NP-complete, NP-hard)

- A language L is NP-hard (NP-difficult) if every language L' in NP is reducible in polynomial time to L.
- A language L is NP-complete if:
  – $L \in NP$
  – L is NP-hard

Definition (PSPACE-complete, PSPACE-hard)

- A language L is PSPACE-hard (PSPACE-difficult) if every language L' in PSPACE is reducible in polynomial time to L.
- A language L is PSPACE-complete if:
  – $L \in PSPACE$
  – L is PSPACE-hard

Complete and hard problems

Remarks:

- If we can prove that at least one NP-hard problem is in P then P = NP
- If P $\neq$ NP then no NP complete problem can be solved in polynomial time

Open problem: Is P = NP? (Millenium Problem)

How to show that a language L is NP-complete?

1. Prove that $L \in NP$
2. Find a language L' known to be NP-complete and reduce it to L

Often used: the SAT problem (Proved to be NP-complete by S. Cook)

$$L' = L_{\text{sat}} = \{w \mid w \text{ is a satisfiable formula of propositional logic}\}$$

Cook's theorem

Theorem $SAT = \{w \mid w \text{ is a satisfiable formula of propositional logic}\}$ is NP-complete.

Idea of proof: Last lecture

Examples of NP-complete problems

Examples of NP-complete problems:

1. Is a logical formula satisfiable? (SAT, 3-CNF-SAT)
2. Does a graph contain a clique of size k? (Clique of size k)
3. Is a (un)directed graph hamiltonian? (Hamiltonian circle)
4. Can a graph be colored with three colors? (3-colorability)
5. Has a set of integers a subset with sum x? (subset sum)
6. Rucksack problem (knapsack)
7. Multiprocessor scheduling

Examples of NP-complete problems

Examples of NP-complete problems:

1. Is a logical formula satisfiable? (SAT, 3-CNF-SAT)
2. Does a graph contain a clique of size k? (Clique of size k)
3. Is a (un)directed graph hamiltonian? (Hamiltonian circle)
4. Can a graph be colored with three colors? (3-colorability)
5. Has a set of integers a subset with sum x? (subset sum)
6. Rucksack problem (knapsack)
7. Multiprocessor scheduling

Examples of NP-complete problems

Definition (CNF, DNF, k-CNF, k-DNF)

DNF: A formula is in DNF if it has the form

$$(L_1^1 \wedge \cdots \wedge L_{n_1}^1) \vee \cdots \vee (L_1^m \wedge \cdots \wedge L_{n_m}^m)$$

CNF: A formula is in CNF if it has the form

$$(L_1^1 \vee \cdots \vee L_{n_1}^1) \wedge \cdots \wedge (L_1^m \vee \cdots \vee L_{n_m}^m)$$

k-DNF: A formula is in k-DNF if it is in DNF and all its conjunctions have k literals

k-CNF: A formula is in k-CNF if it is in CNF and all its disjunctions have k literals

Examples of NP-complete problems

SAT $= \{w \mid w$ is a satisfiable formula of propositional logic$\}$

CNF-SAT $= \{w \mid w$ is a satisfiable formula of propositional logic in CNF$\}$

k-CNF-SAT $= \{w \mid w$ is a satisfiable formula of propositional logic in k-CNF$\}$

Examples of NP-complete problems

Theorem

The following problems are in NP and are NP-complete:

(1) SAT

(2) CNF-SAT

(3) k-CNF-SAT for $k \geq 3$

Examples of NP-complete problems

Theorem

The following problems are in NP and are NP-complete:

(1) SAT

(2) CNF-SAT

(3) k-CNF-SAT for $k \geq 3$

Proof: (1) SAT is NP-complete by Cook's theorem.

CNF-SAT and k-CNF-SAT are clearly in NP.

(3) We show that 3-CNF-SAT is NP-hard. For this, we construct a polynomial reduction of SAT to 3-CNF-SAT.

Examples of NP-complete problems

Proof: (ctd.) Polynomial reduction of SAT to 3-CNF.

Let F be a propositional formula of length n

Step 1 Move negation inwards (compute the negation normal form) $\mapsto O(n)$

Step 2 Fully bracket the formula $\mapsto O(n)$

$$P \wedge Q \wedge R \mapsto (P \wedge Q) \wedge R$$

Step 3 Starting from inside out replace subformula $Q \circ R$ with a new propositional variable $P_{Q \circ R}$ and add the formula

$$P_{Q \circ R} \rightarrow (Q \circ R) \text{ and } (Q \circ R) \rightarrow P_{Q \circ R} \quad (\circ \in \{\vee, \wedge\}) \qquad \mapsto O(p(n))$$

Step 4 Write all formulae above as clauses $\mapsto$ Rename(F) $\mapsto O(n)$

Let $f : \Sigma^* \rightarrow \Sigma^*$ be defined by:

$f(F) = P_F \wedge \text{Rename}(F)$ if F is a well-formed formula

and $f(w) = \perp$ otherwise. Then:

$F \in$ SAT iff F is a satisfiable formula in prop. logic iff $P_F \wedge \mathit{Rename}(F)$ is satisfiable

iff $f(F) \in$ 3-CNF-SAT

Example

Let F be the following formula:

$$[(Q \wedge \neg P \wedge \neg(\neg(\neg Q \vee \neg R))) \vee (Q \wedge \neg P \wedge \neg(Q \wedge \neg P))] \wedge (P \vee R).$$

Step 1: After moving negations inwards we obtain the formula:

$$F_1 = [(Q \wedge \neg P \wedge (\neg Q \vee \neg R)) \vee (Q \wedge \neg P \wedge (\neg Q \vee P))] \wedge (P \vee R)$$

Step 2: After fully bracketing the formula we obtain:

$$F_2 = [((Q \wedge \neg P) \wedge (\neg Q \vee \neg R)) \vee ((Q \wedge \neg P) \wedge (\neg Q \vee P)] \wedge (P \vee R)$$

Step 3: Replace subformulae with new propositional variables (starting inside).

$$\underbrace{\underbrace{[\underbrace{(\underbrace{(Q \wedge \neg P)}_{P_1} \wedge \underbrace{(\neg Q \vee \neg R)}_{P_2})}_{P_6} \vee \underbrace{(\underbrace{(Q \wedge \neg P)}_{P_1} \wedge \underbrace{(\neg Q \vee P)}_{P_4})}_{P_7}]}_{P_8} \wedge \underbrace{(P \vee R)}_{P_5}}_{P_F}.$$

Example

Step 3: Replace subformulae with new propositional variables (starting inside).

$$\underbrace{\underbrace{[\underbrace{(\underbrace{(Q \land \neg P)}_{P_1} \land \underbrace{(\neg Q \lor \neg R)}_{P_2})}_{P_6} \lor \underbrace{(\underbrace{(Q \land \neg P)}_{P_1} \land \underbrace{(\neg Q \lor P)}_{P_4})}_{P_7}]}_{P_8} \land \underbrace{(P \lor R)}_{P_5}}_{P_F} .$$

F is satisfiable iff the following formula is satisfiable:

$$\begin{array}{llllll}
P_F & \land & (P_F \leftrightarrow (P_8 \land P_5) & \land & (P_1 \leftrightarrow (Q \land \neg P)) \\
 & \land & (P_8 \leftrightarrow (P_6 \lor P_7)) & \land & (P_2 \leftrightarrow (\neg Q \lor \neg R)) \\
 & \land & (P_6 \leftrightarrow (P_1 \land P_2)) & \land & (P_4 \leftrightarrow (\neg Q \lor P)) \\
 & \land & (P_7 \leftrightarrow (P_1 \land P_4)) & \land & (P_5 \leftrightarrow (P \lor R))
\end{array}$$

can further exploit polarity

Example

Step 3: Replace subformulae with new propositional variables (starting inside).

$$\underbrace{\underbrace{[\underbrace{(\underbrace{(Q \wedge \neg P)}_{P_1} \wedge \underbrace{(\neg Q \vee \neg R)}_{P_2})}_{P_6} \vee \underbrace{(\underbrace{(Q \wedge \neg P)}_{P_1} \wedge \underbrace{(\neg Q \vee P)}_{P_4})}_{P_7}]}_{P_8} \wedge \underbrace{(P \vee R)}_{P_5}}_{P_F}.$$

F is satisfiable iff the following formula is satisfiable:

$$\begin{array}{llllll} P_F & \wedge & (P_F \rightarrow (P_8 \wedge P_5)) & \wedge & (P_1 \rightarrow (Q \wedge \neg P)) \\ & \wedge & (P_8 \rightarrow (P_6 \vee P_7)) & \wedge & (P_2 \rightarrow (\neg Q \vee \neg R)) \\ & \wedge & (P_6 \rightarrow (P_1 \wedge P_2)) & \wedge & (P_4 \rightarrow (\neg Q \vee P)) \\ & \wedge & (P_7 \rightarrow (P_1 \wedge P_4)) & \wedge & (P_5 \rightarrow (P \vee R)) \end{array}$$

Example

F is satisfiable iff the following formula is satisfiable:

$$
\begin{array}{lllll}
P_F & \wedge & (P_F \rightarrow (P_8 \wedge P_5) & \wedge & (P_1 \rightarrow (Q \wedge \neg P)) \\
& \wedge & (P_8 \rightarrow (P_6 \vee P_7)) & \wedge & (P_2 \rightarrow (\neg Q \vee \neg R)) \\
& \wedge & (P_6 \rightarrow (P_1 \wedge P_2)) & \wedge & (P_4 \rightarrow (\neg Q \vee P)) \\
& \wedge & (P_7 \rightarrow (P_1 \wedge P_4)) & \wedge & (P_5 \rightarrow (P \vee R))
\end{array}
$$

Step 4: Compute the CNF (at most 3 literals per clause)

$$
\begin{array}{lllll}
P_F & \wedge & (\neg P_F \vee P_8) \wedge (\neg P_F \vee P_5) & \wedge & (\neg P_1 \vee Q) \wedge (\neg P_1 \vee \neg P) \\
& \wedge & (\neg P_8 \vee P_6 \vee P_7) & \wedge & (\neg P_2 \vee \neg Q \vee \neg R) \\
& \wedge & (\neg P_6 \vee P_1) \wedge (\neg P_6 \vee P_2) & \wedge & (\neg P_4 \vee \neg Q \vee P) \\
& \wedge & (\neg P_7 \vee P_1) \wedge (\neg P_7 \vee P_4) & \wedge & (\neg P_5 \vee P \vee R)
\end{array}
$$

Examples of NP-complete problems

Proof: (ctd.) It immediately follows that CNF and k-CNF are NP-complete

Polynomial reduction from 3-CNF-SAT to CNF-SAT:

$f(F) = F$ for every formula in 3-CNF-SAT and $\perp$ otherwise.

$F \in$ 3-CNF-SAT iff $f(F) = F \in$ CNF-SAT.

Polynomial reduction from 3-CNF-SAT to k-CNF-SAT, $k > 3$

For every formula in 3-CNF-SAT:
$f(F) = F'$ (where F' is obtained from F by replacing a literal L with $\underbrace{L \vee \dots \vee L}_{k-2 \text{ times}}$).

$f(w) = \perp$ otherwise.

$F \in$ 3-CNF-SAT iff $f(F) = F' \in k$-CNF-SAT (because $F' \equiv F$)

Examples of problems in P

Theorem

The following problems are in P:

(1) DNF

(2) k-DNF for all k

(3) 2-CNF

(1) Let $F = (L_1^1 \wedge \cdots \wedge L_{n_1}^1) \vee \cdots \vee (L_1^m \wedge \cdots \wedge L_{n_m}^m)$ be a formula in DNF.

F is satisfiable iff for some i: $(L_1^i \wedge \cdots \wedge L_{n_1}^i)$ is satisfiable. A conjunction of literals is satisfiable iff it does not contain complementary literals.

(2) follows from (1)

(3) Finite set of 2-CNF formulae over a finite set of propositional variables. Resolution $\mapsto$ at most quadratically many inferences needed.

Examples of NP-complete problems

Examples of NP-complete problems:

1. Is a logical formula satisfiable? (SAT)
2. Does a graph contain a clique of size k?
3. Rucksack problem
4. Is a (un)directed graph hamiltonian?
5. Can a graph be colored with three colors?
6. Multiprocessor scheduling

Examples of NP-complete problems

Definition

A clique in a graph G is a complete subgraph of G.

Clique $= \{(G, k) \mid G$ is an undirected graph which has a clique of size $k\}$

Examples of NP-complete problems

Theorem Clique is NP-complete.

Proof: (1) We show that Clique is in NP:

We can construct for instance an NTM which accepts Clique.

- M builds a set V' of nodes (subset of the nodes of G) by choosing k nodes of G (we say that M "guesses" V').
- M checks for all nodes in V' if there are nodes to all other nodes. (this can be done in polynomial time)

"guess a subgraph with k vertices; check in polynomial time that it is a clique"

Examples of NP-complete problems

> **Theorem** Clique is NP-complete.

Proof: (2) We show that Clique is *NP*-hard by showing that 3-CNF-SAT $\preceq_{\text{pol}}$ Clique.

Let $\mathcal{G}$ be the set of all undirected graphs. We want to construct a map f (DTM computable in polynomial time) which associates with every formula F a pair $(G_F, k_F) \in \mathcal{G} \times \mathbb{N}$ such that

$$F \in \text{3-CNF-SAT} \quad \text{iff} \quad G_F \text{ has a clique of size } k_F.$$

$F \in$ 3-CNF $\Rightarrow F = (L_1^1 \vee L_2^1 \vee L_3^1) \wedge \cdots \wedge (L_1^m \vee L_2^m \vee L_3^m)$

F satisfiable iff there exists an assignment $\mathcal{A}$ such that in every clause in F at least one literal is true and it is impossible that P and $\neg P$ are true at the same time.

Examples of NP-complete problems

Theorem Clique is NP-complete.

Proof: (ctd.) Let $k_F := m$ (the number of clauses). We construct G_F as follows:

- **Vertices:** all literals in F.
- **Edges:** We have an edge between two literals if they (i) can become true in the same assignment and (ii) belong to different clauses.

Then:
(1) $f(F)$ is computable in polynomial time.
(2) The following are equivalent:

(a) G_F has a clique of size k_F.

(b) There exists a set of nodes $\{L^1_{i_1}, \dots, L^m_{i_m}\}$ in G_F which does not contain complementary literals.

(c) There exists an assignment which makes F true.

(d) F is satisfiable.

Examples of NP-complete problems

Examples of NP-complete problems:

1. Is a logical formula satisfiable? (SAT, 3-CNF-SAT)
2. Does a graph contain a clique of size k?
3. Rucksack problem
4. Is a (un)directed graph hamiltonian?
5. Can a graph be colored with three colors?
6. Multiprocessor scheduling

Examples of NP-complete problems

Examples of NP-complete problems:

1. Is a logical formula satisfiable? (SAT)
2. Does a graph contain a clique of size k?
3. Rucksack problem
4. Can a graph be colored with three colors?
5. Is a (un)directed graph hamiltonian?
6. Multiprocessor scheduling

Examples of NP-complete problems

Definition (Rucksack problem)

A rucksack problem consists of:

- n objects with weights $a_1, \ldots, a_n$
- a maximum weight b

The rucksack problem is solvable if there exists a subset of the given objects with total weight b.

$$\text{Rucksack} = \{(b, a_1, \ldots, a_n) \in \mathbb{N}^{n+1} \mid \exists I \subseteq \{1, \ldots, n\} \ s.t. \ \textstyle\sum_{i \in I} a_i = b\}$$

Examples of NP-complete problems

Theorem Rucksack is NP-complete.

Proof: (1) Rucksack is in NP: We guess I and check whether $\sum_{i \in I} a_i = b$

(2) Rucksack is NP-hard: We show that 3-CNF-SAT $\prec_{\text{pol}}$ Rucksack.

Construct $f : \text{3-CNF} \rightarrow \mathbb{N}^*$ as follows.

Consider a 3-CNF formula $F = (L_1^1 \vee L_2^1 \vee L_3^1) \wedge \cdots \wedge (L_1^m \vee L_2^m \vee L_3^m)$

$f(F) = (b, a_1, \ldots, a_n)$ where:

(i) a_i encodes which atom occurs in which clause as follows:
p_i positive occurrences; n_i negative occurrences (numbers with $n + m$ positions)

– first m digits of p_i: p_{i_j} how often i-th atom occurs positively in j-th clause
– first m digits of n_i: n_{i_j} how often i-th atom occurs negatively in j-th clause

– last n digits of p_i, n_i: p_{i_j}, n_{i_j} which atom is referred by p_i
p_i, n_i contain 1 at position $m + i$ and 0 otherwise.

Example

Let the set Prop of propositional variables consist of $\{x_1, x_2, x_3, x_4, x_5\}$.

$F: \quad (x_1 \vee \neg x_2 \vee x_4) \wedge (x_2 \vee x_2 \vee \neg x_5) \wedge (\neg x_3 \vee \neg x_1 \vee x_4)$

$p_1 = 100\ 10000$	$n_1 = 001\ 10000$
$p_2 = 020\ 01000$	$n_2 = 100\ 01000$
$p_3 = 000\ 00100$	$n_3 = 001\ 00100$
$p_4 = 101\ 00010$	$n_4 = 000\ 00010$
$p_5 = 000\ 00001$	$n_5 = 010\ 00001$

Satisfying assignment: $\mathcal{A}(x_1)=\mathcal{A}(x_2)=\mathcal{A}(x_5)=1$ and $\mathcal{A}(x_3)=\mathcal{A}(x_4)=0$.

$$p_1 + p_2 + p_5 + n_3 + n_4 = \underbrace{121}_{\substack{\text{all digits } \leq 3 \\ \text{because 3 lit./clause}}} \quad \underbrace{11111}_{\substack{\text{all 1} \\ \text{all atoms considered}}}$$

Examples of NP-complete problems

Proof: (ctd.) If we have a satisfying assignment $\mathcal{A}$, we take for every propositional variable x_i mapped to 0 the number n_i and for every propositional variable x_i mapped to 1 the number p_i.

The sum of these numbers is $b_1 \dots b_m \underbrace{1 \dots 1}_{n \text{ times}}$ with $b_i \leq 3$,

so $b_1 \dots b_m \underbrace{1 \dots 1}_{n} < \underbrace{4 \dots 4}_{m} \underbrace{1 \dots 1}_{n}$

Let $b := \underbrace{4 \dots 4}_{m} \underbrace{1 \dots 1}_{n}$. We choose $\{a_1, \dots, a_k\} = \{p_1, \dots, p_n\} \cup \{n_1, \dots, n_n\} \cup C$.

The role of the numbers in $C = \{c_1, \dots, c_m, d_1, \dots, d_m\}$ is to make the sum of the a_is equal to b: $c_{i_j} = 1$ iff $i = j$; $d_{i_j} = 2$ iff $i = j$ (they are zero otherwise).

$f(F) \in$ Rucksack iff a subset I of $\{a_1, \dots, a_k\}$ adds up to b

iff a subset I of $\{p_1, \dots, p_n\} \cup \{n_1, \dots, n_n\}$ adds up to $b_1 \dots b_m 1 \dots 1$

iff for a subset I of $\{p_1, \dots, p_n\} \cup \{n_1, \dots, n_n\}$ there exists an assignment $\mathcal{A}$ with $\mathcal{A}(P_i) = 1$(*resp.* 0) iff p_i(*resp.* n_i) occurs in I iff F satisfiable

Summary

Examples of NP-complete problems:

1. Is a logical formula satisfiable? (SAT)
2. Does a graph contain a clique of size k?
3. Rucksack problem
4. Can a graph be colored with three colors?
5. Is a (un)directed graph hamiltonian?
6. Multiprocessor scheduling

more examples next time