Advanced Topics in Theoretical Computer Science

Part 5: Complexity (Part II)

30.01.2014

Viorica Sofronie-Stokkermans

Universität Koblenz-Landau

e-mail: sofronie@uni-koblenz.de

Contents

Motivation

Goals:

- Define formally time and space complexity   last time
- Define a family of "complexity classes": P, NP, PSPACE, ...
- Study the links between complexity classes
- Learn how to show that a problem is in a certain complexity class

    Reductions to problems known to be in the complexity class

- Closure of complexity classes

We will give examples of problems from various areas and study their complexity.

DTIME/NTIME and DSPACE/NSPACE

DTIME/NTIME **Basic model:** k-DTM or k-NTM M (one tape for the input)

If M makes for every input word of length n at most $T(n)$ steps, then M is $T(n)$-time bounded.

Definition ($NTIME(T(n))$, $DTIME(T(n))$)

- $DTIME(T(n))$ class of all languages accepted by $T(n)$-time bounded DTMs.
- $NTIME(T(n))$ class of all languages accepted by $T(n)$-time bounded NTMs.

DSPACE/NSPACE **Basic model:** k-DTM or k-NTM M with special tape for the input (is read-only) + k storage tapes (offline DTM) $\mapsto$ needed if $S(n)$ sublinear

If M needs, for every input word of length n, at most $S(n)$ cells on the storage tapes then M is $S(n)$-space bounded.

Definition ($NSPACE(S(n))$, $DSPACE(S(n))$)

- $DSPACE(S(n))$ class of all languages accepted by $S(n)$-space bounded DTMs.
- $NSPACE(S(n))$ class of all languages accepted by $S(n)$-space bounded NTMs.

Questions

Time: Is any language in $DTIME(f(n))$ decided by some DTM?

Space: Is any language in $DSPACE(f(n))$ decided by some DTM?

Time/Space: What about $NTIME(f(n)), NSPACE(f(n))$

Time vs. Space: What are the links between $DTIME(f(n)), DSPACE(f(n)), NTIME(f(n)), NSPACE(f(n))$

Answers

Answers (Informally)

Time: Every language from $DTIME(f(n))$ is decidable:

for an input of length n we wait as long as the value $f(n)$.

If until then no answer "YES" then the answer is "NO".

Space: Every language from $DSPACE(f(n))$ is decidable:

There are only finitely many configurations. We write all configurations.

If the TM does not halt then there is a loop. This can be detected.

Answers

Answers (Informally)

NTM vs. DTM: Clearly, $DTIME(f(n)) \subseteq NTIME(f(n))$ and

$$DSPACE(f(n)) \subseteq NSPACE(f(n))$$

If we try to simulate an NTM with a DTM we may need exponentially more time. Therefore:

$$NTIME(f(n)) \subseteq DTIME(2^{h(n)}) \text{ where } h \in O(f).$$

For the space complexity we can show that:

$$NSPACE(f(n)) \subseteq DSPACE(f^2(n))$$

Time vs. Space: Clearly, $DTIME(f(n)) \subseteq DSPACE(f(n))$ and

$$NTIME(f(n)) \subseteq NSPACE(f(n))$$

$DSPACE(f(n))$, $NSPACE(f(n))$ are much larger.

Question

What about constant factors?

Constant factors are ignored. Only the rate of growth of a function in complexity classes is important.

Theorem.
For every $c \in \mathbb{R}^+$ and every storage function $S(n)$ the following hold:

- $DSPACE(S(n)) = DSPACE(cS(n))$
- $NSPACE(S(n)) = NSPACE(cS(n))$

Proof (Idea). One direction is trivial. The other direction can be proved by representing a fixed amount $r > \frac{2}{c}$ of neighboring cells on the tape as a new symbol.

The states of the new machine simulate the movements of the read/write head as transitions. For r-cells of the old machine we use only two: in the most unfavourable case when we go from one block to another.

Time acceleration

Theorem For every $c \in \mathbb{R}^+$ and every time function $T(n)$ with $\lim_{n\to\infty} \frac{T(n)}{n} = \infty$ the following hold:

- $DTIME(T(n)) = DTIME(cT(n))$
- $NTIME(T(n)) = NTIME(cT(n))$

Proof (Idea). One direction is trivial. The other direction can be proved by representing a fixed amount $r > \frac{4}{c}$ of neighboring cells on the tape as a new symbol.

The states of the new machine simulate also now which symbol and which position the read/write head of the initial machine has. When the machine is simulated the new machine needs to make 4 steps instead of r: 2 in order to write on the new fields and 2 in order to move the head on the new field and then back on the old (in the worst case).

Big O notation

Theorem: Let T be a time function with $\lim_{n\to\infty} \frac{T(n)}{n} = \infty$ and S a storage function.

(a) If $f(n) \in O(T(n))$ then $DTIME(f(n)) \subseteq DTIME(T(n))$.

(b) If $g(n) \in O(S(n))$ then $DSPACE(g(n)) \subseteq DSPACE(S(n))$.

P, NP, PSPACE

Definition

$$\begin{aligned} P &= \bigcup_{i\geq 1} DTIME(n^i) \\ NP &= \bigcup_{i\geq 1} NTIME(n^i) \\ PSPACE &= \bigcup_{i\geq 1} DSPACE(n^i) \end{aligned}$$

P, NP, PSPACE

Definition

$$
\begin{aligned}
P &= \bigcup_{i\geq 1} DTIME(n^i) \\
NP &= \bigcup_{i\geq 1} NTIME(n^i) \\
PSPACE &= \bigcup_{i\geq 1} DSPACE(n^i)
\end{aligned}
$$

Lemma $NP \subseteq \bigcup_{i\geq 1} DTIME(2^{O(n^d)})$

Proof: Follows from the fact that if L is accepted by a $f(n)$-time bounded NTM then L is accepted by an $2^{O(f(n))}$-time bounded DTM, hence for every $d \geq 1$ we have:

$$NTIME(n^d) \subseteq DTIME(2^{O(n^d)})$$

P, NP, PSPACE

$$
\begin{array}{lcl}
P & = & \bigcup_{i\geq 1} DTIME(n^i) \\
NP & = & \bigcup_{i\geq 1} NTIME(n^i) \\
PSPACE & = & \bigcup_{i\geq 1} DSPACE(n^i) \\
NP & \subseteq & \bigcup_{i\geq 1} DTIME(2^{O(n^d)})
\end{array}
$$

Intuition

- Problems in P can be solved efficiently; those in NP can be solved in exponential time
- PSPACE is a very large class, much larger that P and NP.

Complexity classes for functions

Definition

A function $f : \mathbb{N} \to \mathbb{N}$ is in P if there exists a DTM M and a polynomial $p(n)$ such that for every n the value $f(n)$ can be computed by M in at most $p(\text{length}(n))$ steps.

Here $\text{length}(n) = \log(n)$: we need $\log(n)$ symbols to represent (binary) the number n.

The other complexity classes for functions are defined in an analogous way.

Relationships between complexity classes

Question:

Which are the links between the complexity classes P, NP and PSPACE?

Relationships between complexity classes

Question:

Which are the links between the complexity classes P, NP and PSPACE?

$\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE}$

Complexity classes

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

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! SAT

Complexity classes

Can we find in NP problems which are the most difficult ones in NP?

Complexity classes

Can we find in NP problems which are the most difficult ones in NP?

Answer

There are various ways of defining "the most difficult problem".

They depend on the notion of reducibility which we use.

For a given notion of reducibility the answer is YES.

Such problems are called complete in the complexity class with respect to the notion of reducibility used.

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

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

Reduction

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.

Reduction

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.

Proof: Assume $L_1 \in P$. Then there exists $k \geq 1$ such that L_1 is accepted by n^k-time bounded DTM M_1.

Since $L_2 \preceq_{\text{pol}} L_1$ there exists a polynomial time bounded DTM M_f, which for every input w computes an output $f(w)$ such that $w \in L_2$ if and only if $f(w) \in L_1$.

Let $M_2 = M_f M_1$. Clearly, M_2 accepts L_2. We have to show that M_2 is polynomial time bounded. $w \mapsto M_f$ computes $f(w)$ (pol.size) $\mapsto M_1$ decides if $f(w) \in L_1$ (polynomially many steps)

NP

Theorem (Characterisation of NP)

A language L is in NP if and only if there exists a language L' in P and a $k \geq 0$ such that for all $w \in \Sigma^*$:

$$w \in L \text{ iff } \quad \text{there exists } c : \langle w, c \rangle \in L' \text{ and } |c| < |w|^k$$

c is also called witness or certificate for w in L.

A DTM which accepts the language L' is called verifier.

Important

A decision procedure is in NP iff every "Yes" instance has a short witness

(i.e. its length is polynomial in the length of the input)

which can be verified in polynomial time.

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

Complete and hard problems

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)

Complete and hard problems

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

Complete and hard problems

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

Is this sufficient?

Complete and hard problems

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

Is this sufficient?

Yes.

If L' is NP-complete then every language in NP is reducible to L', therefore also to L.

Complete and hard problems

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

Is this sufficient?

Yes.

If $L' \in NP$ then every language in NP is reducible to L' and therefore also to L.

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

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

Stephen Cook

Stephen Arthur Cook (born 1939)

- Major contributions to complexity theory.
  Considered one of the forefathers of computational complexity theory.
- 1971 'The Complexity of Theorem Proving Procedures'
  Formalized the notions of polynomial-time reduction and NP-completeness, and proved the existence of an NP-complete problem by showing that the Boolean satisfiability problem (SAT) is NP-complete.
- Currently University Professor at the University of Toronto
- 1982: Turing award for his contributions to complexity theory.

Cook's theorem

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

Cook's theorem

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

Proof (Idea)

To show: (1) $SAT \in NP$

(2) for all $L \in NP$, $L \preceq_{\text{pol}} SAT$

Cook's theorem

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

Proof (Idea)

To show: (1) $SAT \in NP$

(2) for all $L \in NP$, $L \preceq_{\text{pol}} SAT$

(1) Construct a k-tape NTM M which can accept SAT in polynomial time:

$w \in \Sigma_{PL}^*$ $\mapsto$ M does not halt if $w \notin SAT$

M finds in polynomial time a satisfying assignment

(a) scan w and see if it a well-formed formula; collect atoms $\mapsto O(|w|^2)$
(b) if not well-formed: inf.loop; if well-formed M guesses a satisfying assignment $\mapsto O(|w|)$
(c) check whether w true under the assignment $\mapsto O(p(|w|))$
(d) if false: inf.loop; otherwise halt.

"guess (satisfying) assignment $\mathcal{A}$; check in polynomial time that formula true under $\mathcal{A}$"

Cook's theorem

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

Proof (Idea) (2) We show that for all $L \in NP$, $L \preceq_{\text{pol}} SAT$

- We show that we can simulate the way a NTM works using propositional logic.
- Let $L \in NP$. There exists a p-time bounded NTM which accepts L. (Assume w.l.o.g. that M has only one tape and does not hang.)

  For M and w we define a propositional logic language and a formula $T_{M,w}$ such that

  $$M \text{ accepts } w \quad \text{iff} \quad T_{M,w} \text{ is satisfiable.}$$

- We show that the map f with $f(w) = T_{M,w}$ has polynomial complexity.

Closure of complexity classes

P, PSPACE are closed under complement

All complexity classes which are defined in terms of deterministic Turing machines are closed under complement.

Proof: If a language L is in such a class then also its complement is

(run the machine for L and revert the output)

Closure of complexity classes

Is NP closed under complement?

Closure of complexity classes

Is NP closed under complement?

Nobody knows!

Definition

co-NP is the class of all laguages for which the complement is in NP

$$\text{co-NP} = \{L \mid \overline{L} \in NP\}$$

Relationships between complexity classes

It is not yet known whether the following relationships hold:

$P \stackrel{?}{=} NP$

$NP \stackrel{?}{=} \text{co-}NP$

$P \stackrel{?}{=} PSPACE$

$NP \stackrel{?}{=} PSPACE$

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)
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 (SAT, 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

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

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:

$F \in \text{SAT}$ 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:

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

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).

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

F is satisfiable iff the following formula is satisfiable:

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

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:

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

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 $\underbrace{L \vee \cdots \vee L}_{k-2 \text{ times}}$).

$f(w) = \perp$ otherwise.

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

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.

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

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 $\prec_{\text{pol}}$ 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

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

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

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

... other examples next time