Theory of Computer Science

D3. Proving NP-Completeness

Gabriele Röger

University of Basel

May 8, 2023

Overview

Reminder: P and NP

P: class of languages that are decidable in polynomial time by a deterministic Turing machine

NP: class of languages that are decidable in polynomial time by a non-deterministic Turing machine

Reminder: Polynomial Reductions

Definition (Polynomial Reduction)

Let $A \subseteq \Sigma^*$ and $B \subseteq \Gamma^*$ be decision problems.
We say that A can be polynomially reduced to B,
written $A \leq_p B$, if there is a function $f : \Sigma^* \to \Gamma^*$ such that:

- f can be computed in polynomial time by a DTM
- f reduces A to B
  - i. e., for all $w \in \Sigma^*$: $w \in A$ iff $f(w) \in B$

f is called a polynomial reduction from A to B

Transitivity of $\leq_p$: If $A \leq_p B$ and $B \leq_p C$, then $A \leq_p C$.

Reminder: NP-Hardness and NP-Completeness

Definition (NP-Hard, NP-Complete)

Let B be a decision problem.

B is called NP-hard if $A \leq_{\mathrm{p}} B$ for all problems $A \in \mathrm{NP}$.

B is called NP-complete if $B \in \mathrm{NP}$ and B is NP-hard.

Proving NP-Completeness by Reduction

- Suppose we know one NP-complete problem
  (we will use satisfiability of propositional logic formulas).
- With its help, we can then prove quite easily
  that further problems are NP-complete.

Proving NP-Completeness by Reduction

- Suppose we know one NP-complete problem
  (we will use satisfiability of propositional logic formulas).
- With its help, we can then prove quite easily
  that further problems are NP-complete.

Theorem (Proving NP-Completeness by Reduction)

Let A and B be problems such that:

- *A is NP-hard, and*
- *$A \leq_{\mathrm{p}} B$.*

Then B is also NP-hard.
If furthermore $B \in \mathrm{NP}$, then B is NP-complete.

Proving NP-Completeness by Reduction: Proof

Proof.

First part: We must show $X \leq_p B$ for all $X \in NP$.

Proving NP-Completeness by Reduction: Proof

Proof.

First part: We must show $X \leq_p B$ for all $X \in NP$.

From $X \leq_p A$ (because A is NP-hard) and $A \leq_p B$ (by prerequisite), this follows due to the transitivity of $\leq_p$.

Proving NP-Completeness by Reduction: Proof

Proof.

First part: We must show $X \leq_{\mathrm{p}} B$ for all $X \in \mathrm{NP}$.

From $X \leq_{\mathrm{p}} A$ (because A is NP-hard) and $A \leq_{\mathrm{p}} B$ (by prerequisite), this follows due to the transitivity of $\leq_{\mathrm{p}}$.

Second part: follows directly by definition of NP-completeness. □

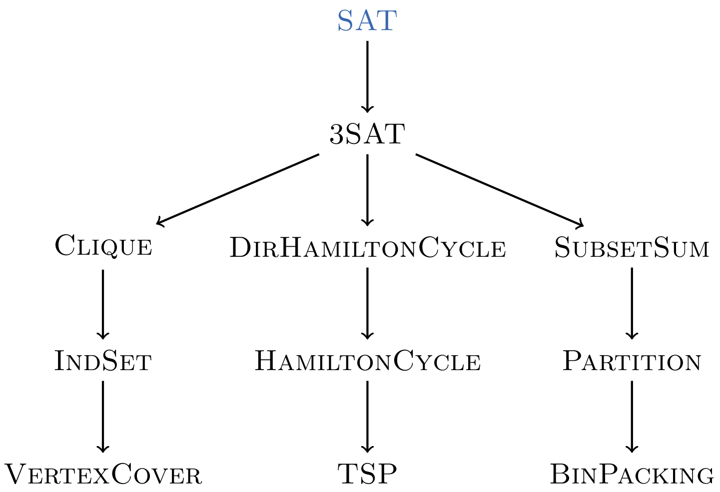
NP-Complete Problems

- There are thousands of known NP-complete problems.
- An extensive catalog of NP-complete problems from many areas of computer science is contained in:

  Michael R. Garey and David S. Johnson:
  Computers and Intractability —
  A Guide to the Theory of NP-Completeness
  W. H. Freeman, 1979.

- In the remaining chapters, we get to know some of these problems.

Overview of the Reductions

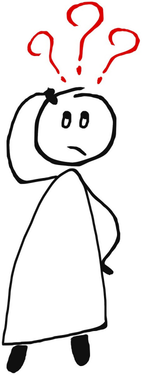


What Do We Have to Do?

- We want to show the NP-completeness of these 11 problems.
- We first show that SAT is NP-complete.
- Then it is sufficient to show
  - that polynomial reductions exist for all edges in the figure (and thus all problems are NP-hard)
  - and that the problems are all in NP.

(It would be sufficient to show membership in NP only for the leaves in the figure. But membership is so easy to show that this would not save any work.)

Questions



Questions?

Propositional Logic

- We need to establish NP-completeness of one problem "from scratch".
- We will use satisfiability of propositional logic formulas.
- So what is this?

Let's briefly cover the basics.

Propositional Logic: Syntax

- Let A be a set of atomic propositions
  $\rightarrow$ variables that can be true or false

Propositional Logic: Syntax

- Let A be a set of atomic propositions
  $\rightarrow$ variables that can be true or false
- Every $a \in A$ is a propositional formula over A.

Propositional Logic: Syntax

- Let A be a set of atomic propositions
  $\rightarrow$ variables that can be true or false
- Every $a \in A$ is a propositional formula over A.
- If φ is a propositional formula over A,
  then so is its negation $\neg\varphi$.

Propositional Logic: Syntax

- Let A be a set of atomic propositions
  $\rightarrow$ variables that can be true or false
- Every $a \in A$ is a propositional formula over A.
- If φ is a propositional formula over A, then so is its negation $\neg\varphi$.
- If $\varphi_1, \dots, \varphi_n$ are propositional formulas over A, then so is the conjunction $(\varphi_1 \wedge \dots \wedge \varphi_n)$.

Propositional Logic: Syntax

- Let A be a set of atomic propositions
  $\rightarrow$ variables that can be true or false
- Every $a \in A$ is a propositional formula over A.
- If φ is a propositional formula over A,
  then so is its negation $\neg\varphi$.
- If $\varphi_1, \ldots, \varphi_n$ are propositional formulas over A,
  then so is the conjunction $(\varphi_1 \wedge \cdots \wedge \varphi_n)$.
- If $\varphi_1, \ldots, \varphi_n$ are propositional formulas over A,
  then so is the disjunction $(\varphi_1 \vee \cdots \vee \varphi_n)$.

Propositional Logic: Syntax

- Let A be a set of atomic propositions
  $\rightarrow$ variables that can be true or false
- Every $a \in A$ is a propositional formula over A.
- If φ is a propositional formula over A,
  then so is its negation $\neg\varphi$.
- If $\varphi_1, \dots, \varphi_n$ are propositional formulas over A,
  then so is the conjunction $(\varphi_1 \wedge \cdots \wedge \varphi_n)$.
- If $\varphi_1, \dots, \varphi_n$ are propositional formulas over A,
  then so is the disjunction $(\varphi_1 \vee \cdots \vee \varphi_n)$.

Example

$\neg(X \wedge (Y \vee \neg(Z \wedge Y)))$ is a propositional formula over $\{X, Y, Z\}$.

Propositional Logic: Semantics

- A truth assignment for a set of atomic propositions A is a function $\mathcal{I} : A \to \{T, F\}$.
- A formula can be true or false under a given truth assignment. Write $\mathcal{I} \models \varphi$ to express that φ is true under $\mathcal{I}$.
  - Atomic variable a is true under $\mathcal{I}$ iff $\mathcal{I}(a) = T$.
  - Negation $\neg\varphi$ is true under $\mathcal{I}$ iff φ is not: $\mathcal{I} \models \neg\varphi$ iff $\mathcal{I} \not\models \varphi$
  - Conjunction $(\varphi_1 \wedge \cdots \wedge \varphi_n)$ is true under $\mathcal{I}$ iff each φ_i is: $\mathcal{I} \models (\varphi_1 \wedge \cdots \wedge \varphi_n)$ iff $\mathcal{I} \models \varphi_i$ for all $i \in \{1, \ldots, n\}$
  - Disjunction $(\varphi_1 \vee \cdots \vee \varphi_n)$ is true under $\mathcal{I}$ iff some φ_i is: $\mathcal{I} \models (\varphi_1 \vee \cdots \vee \varphi_n)$ iff exists $i \in \{1, \ldots, n\}$ such that $\mathcal{I} \models \varphi_i$

Propositional Logic: Example

Consider truth assignment $\mathcal{I} = \{X \mapsto F, Y \mapsto T, Z \mapsto F\}$.

Is $\neg(X \wedge (Y \vee \neg(Z \wedge Y)))$ true under $\mathcal{I}$?

Propositional Logic: Exercise (slido)

Consider truth assignment

$$\mathcal{I} = \{X \mapsto F, Y \mapsto T, Z \mapsto F\}.$$

Is $(X \lor (\neg Z \land Y))$ true under $\mathcal{I}$?

More Propositional Logic

- $(\varphi \to \psi)$ is a short-hand notation for formula $(\neg\varphi \vee \psi)$.

More Propositional Logic

- $(\varphi \rightarrow \psi)$ is a short-hand notation for formula $(\neg\varphi \vee \psi)$.
- $(\varphi \rightarrow \psi)$ is true under variable assignment $\mathcal{I}$ if

More Propositional Logic

- $(\varphi \rightarrow \psi)$ is a short-hand notation for formula $(\neg\varphi \vee \psi)$.
- $(\varphi \rightarrow \psi)$ is true under variable assignment $\mathcal{I}$ if
  - φ is not true under $\mathcal{I}$, or

More Propositional Logic

- $(\varphi \to \psi)$ is a short-hand notation for formula $(\neg\varphi \lor \psi)$.
- $(\varphi \to \psi)$ is true under variable assignment $\mathcal{I}$ if
  - φ is not true under $\mathcal{I}$, or
  - ψ is true under $\mathcal{I}$.

More Propositional Logic

- $(\varphi \to \psi)$ is a short-hand notation for formula $(\neg\varphi \lor \psi)$.
- $(\varphi \to \psi)$ is true under variable assignment $\mathcal{I}$ if
  - φ is not true under $\mathcal{I}$, or
  - ψ is true under $\mathcal{I}$.
- If $(\varphi \to \psi)$ and φ are true under $\mathcal{I}$
  then also ψ must be true under $\mathcal{I}$.

More Propositional Logic

- $(\varphi \rightarrow \psi)$ is a short-hand notation for formula $(\neg\varphi \vee \psi)$.
- $(\varphi \rightarrow \psi)$ is true under variable assignment $\mathcal{I}$ if
  - φ is not true under $\mathcal{I}$, or
  - ψ is true under $\mathcal{I}$.
- If $(\varphi \rightarrow \psi)$ and φ are true under $\mathcal{I}$ then also ψ must be true under $\mathcal{I}$.

- $(\varphi \leftrightarrow \psi)$ is a short-hand notation for formula $((\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi))$

More Propositional Logic

- $(\varphi \rightarrow \psi)$ is a short-hand notation for formula $(\neg\varphi \vee \psi)$.
- $(\varphi \rightarrow \psi)$ is true under variable assignment $\mathcal{I}$ if
  - φ is not true under $\mathcal{I}$, or
  - ψ is true under $\mathcal{I}$.
- If $(\varphi \rightarrow \psi)$ and φ are true under $\mathcal{I}$ then also ψ must be true under $\mathcal{I}$.

- $(\varphi \leftrightarrow \psi)$ is a short-hand notation for formula $((\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi))$
- $(\varphi \leftrightarrow \psi)$ is true under variable assignment $\mathcal{I}$ if

More Propositional Logic

- $(\varphi \rightarrow \psi)$ is a short-hand notation for formula $(\neg\varphi \vee \psi)$.
- $(\varphi \rightarrow \psi)$ is true under variable assignment $\mathcal{I}$ if
  - φ is not true under $\mathcal{I}$, or
  - ψ is true under $\mathcal{I}$.
- If $(\varphi \rightarrow \psi)$ and φ are true under $\mathcal{I}$ then also ψ must be true under $\mathcal{I}$.

- $(\varphi \leftrightarrow \psi)$ is a short-hand notation for formula $((\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi))$
- $(\varphi \leftrightarrow \psi)$ is true under variable assignment $\mathcal{I}$ if
  - both, φ and ψ are true under $\mathcal{I}$, or

More Propositional Logic

- $(\varphi \rightarrow \psi)$ is a short-hand notation for formula $(\neg\varphi \vee \psi)$.
- $(\varphi \rightarrow \psi)$ is true under variable assignment $\mathcal{I}$ if
  - φ is not true under $\mathcal{I}$, or
  - ψ is true under $\mathcal{I}$.
- If $(\varphi \rightarrow \psi)$ and φ are true under $\mathcal{I}$ then also ψ must be true under $\mathcal{I}$.

- $(\varphi \leftrightarrow \psi)$ is a short-hand notation for formula $((\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi))$
- $(\varphi \leftrightarrow \psi)$ is true under variable assignment $\mathcal{I}$ if
  - both, φ and ψ are true under $\mathcal{I}$, or
  - neither φ nor ψ is true under $\mathcal{I}$.

Short Notations for Conjunctions and Disjunctions

Short notation for addition:

$$\sum_{x \in \{x_1,\dots,x_n\}} x = x_1 + x_2 + \dots + x_n$$

Analogously (possible because of commutativity of $\wedge$ and $\vee$):

$$\Big(\bigwedge_{\varphi \in X} \varphi\Big) = (\varphi_1 \wedge \varphi_2 \wedge \dots \wedge \varphi_n)$$
$$\Big(\bigvee_{\varphi \in X} \varphi\Big) = (\varphi_1 \vee \varphi_2 \vee \dots \vee \varphi_n)$$
$$\text{for } X = \{\varphi_1, \dots, \varphi_n\}$$

SAT Problem

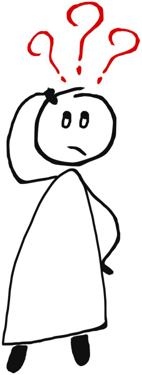
Definition (SAT)

The problem SAT (satisfiability) is defined as follows:

Given: a propositional logic formula φ

Question: Is φ satisfiable,
i.e. is there a variable assignment $\mathcal{I}$ such that $\mathcal{I} \models \varphi$?

Questions



Questions?

Cook-Levin Theorem

SAT is NP-complete

Definition (SAT)

The problem SAT (satisfiability) is defined as follows:

Given: a propositional logic formula φ

Question: Is φ satisfiable?

SAT is NP-complete

Definition (SAT)

The problem SAT (satisfiability) is defined as follows:

Given: a propositional logic formula φ

Question: Is φ satisfiable?

Theorem (Cook, 1971; Levin, 1973)

SAT *is NP-complete.*

SAT is NP-complete

Definition (SAT)

The problem SAT (satisfiability) is defined as follows:

Given: a propositional logic formula φ

Question: Is φ satisfiable?

Theorem (Cook, 1971; Levin, 1973)

SAT *is NP-complete.*

Proof.

SAT $\in$ NP: guess and check.
SAT is NP-hard: somewhat more complicated (to be continued)

...

NP-hardness of SAT (1)

Proof (continued).

We must show: $A \leq_{\mathsf{p}} \mathrm{SAT}$ for all $A \in \mathsf{NP}$.

NP-hardness of SAT (1)

Proof (continued).

We must show: $A \leq_p \mathrm{SAT}$ for all $A \in \mathrm{NP}$.

Let A be an arbitrary problem in NP.

We have to find a polynomial reduction of A to SAT,
i. e., a function f computable in polynomial time
such that for every input word w over the alphabet of A:

$w \in A$ iff $f(w)$ is a satisfiable propositional formula. ...

NP-hardness of SAT (2)

Proof (continued).

Because $A \in \mathrm{NP}$, there is an NTM M and a polynomial p such that M decides the problem A in time p.

Idea: construct a formula that encodes the possible configurations which M can reach in time $p(|w|)$ on input w and that is satisfiable if and only if an accepting configuration can be reached in this time. ...

NP-hardness of SAT (3)

Proof (continued).

Let $M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}} \rangle$ be an NTM for A, and let p be a polynomial bounding the computation time of M. Without loss of generality, $p(n) \geq n$ for all n.

Let $w = w_1 \dots w_n \in \Sigma^*$ be the input for M.

NP-hardness of SAT (3)

Proof (continued).

Let $M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}} \rangle$ be an NTM for A, and let p be a polynomial bounding the computation time of M. Without loss of generality, $p(n) \geq n$ for all n.

Let $w = w_1 \dots w_n \in \Sigma^*$ be the input for M.

We number the tape positions with natural numbers such that the TM head initially is on position 1.

NP-hardness of SAT (3)

Proof (continued).

Let $M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}} \rangle$ be an NTM for A, and let p be a polynomial bounding the computation time of M. Without loss of generality, $p(n) \geq n$ for all n.

Let $w = w_1 \dots w_n \in \Sigma^*$ be the input for M.

We number the tape positions with natural numbers such that the TM head initially is on position 1.

Observation: within $p(n)$ computation steps the TM head can only reach positions in the set $Pos = \{1, \dots, p(n) + 1\}$.

NP-hardness of SAT (3)

Proof (continued).

Let $M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}} \rangle$ be an NTM for A, and let p be a polynomial bounding the computation time of M. Without loss of generality, $p(n) \geq n$ for all n.

Let $w = w_1 \dots w_n \in \Sigma^*$ be the input for M.

We number the tape positions with natural numbers such that the TM head initially is on position 1.

Observation: within $p(n)$ computation steps the TM head can only reach positions in the set $Pos = \{1, \dots, p(n) + 1\}$.

Instead of infinitely many tape positions, we now only need to consider these (polynomially many!) positions. ...

NP-hardness of SAT (4)

Proof (continued).

We can encode configurations of M by specifying:

- what the current state of M is
- on which position in Pos the TM head is located
- which symbols from Γ the tape contains at positions Pos

$\rightsquigarrow$ can be encoded by propositional variables

NP-hardness of SAT (4)

Proof (continued).

We can encode configurations of M by specifying:

- what the current state of M is
- on which position in Pos the TM head is located
- which symbols from Γ the tape contains at positions Pos

$\rightsquigarrow$ can be encoded by propositional variables

To encode a full computation (rather than just one configuration), we need copies of these variables for each computation step.

NP-hardness of SAT (4)

Proof (continued).

We can encode configurations of M by specifying:

- what the current state of M is
- on which position in Pos the TM head is located
- which symbols from Γ the tape contains at positions Pos

$\rightsquigarrow$ can be encoded by propositional variables

To encode a full computation (rather than just one configuration), we need copies of these variables for each computation step.

We only need to consider the computation steps $Steps = \{0, 1, \dots, p(n)\}$ because M should accept within $p(n)$ steps. ...

NP-hardness of SAT (5)

Proof (continued).

Use the following propositional variables in formula $f(w)$:

- $state_{t,q}$ $(t \in Steps,\ q \in Q)$
  $\rightsquigarrow$ encodes the state of the NTM in the t-th configuration
- $head_{t,i}$ $(t \in Steps,\ i \in Pos)$
  $\rightsquigarrow$ encodes the head position in the t-th configuration
- $tape_{t,i,a}$ $(t \in Steps,\ i \in Pos,\ a \in \Gamma)$
  $\rightsquigarrow$ encodes the tape content in the t-th configuration

Construct $f(w)$ such that every satisfying interpretation

- describes a sequence of NTM configurations
- that begins with the start configuration,
- reaches an accepting configuration
- and follows the NTM rules in δ

...

NP-hardness of SAT (6)

Proof (continued).

Auxiliary formula:

$$\mathit{oneof}\, X := \left(\bigvee_{x \in X} x\right) \land \neg \left(\bigvee_{x \in X} \bigvee_{y \in X \setminus \{x\}} (x \land y)\right)$$

Auxiliary notation:

The symbol $\perp$ stands for an arbitrary unsatisfiable formula (e.g., $(A \land \neg A)$, where A is an arbitrary proposition). ...

NP-hardness of SAT (7)

Proof (continued).

1. describe the configurations of the TM:

$$Valid := \bigwedge_{t \in Steps} \Big(oneof\,\{state_{t,q} \mid q \in Q\} \wedge oneof\,\{head_{t,i} \mid i \in Pos\} \wedge \bigwedge_{i \in Pos} oneof\,\{tape_{t,i,a} \mid a \in \Gamma\} \Big)$$

...

NP-hardness of SAT (8)

Proof (continued).

2. begin in the start configuration

$$Init := state_{0,q_0} \wedge \text{head}_{0,1} \wedge \bigwedge_{i=1}^{n} tape_{0,i,w_i} \wedge \bigwedge_{i \in Pos \setminus \{1,\dots,n\}} tape_{0,i,\square}$$

...

NP-hardness of SAT (9)

Proof (continued).

3. reach an accepting configuration

$$Accept := \bigvee_{t \in Steps} state_{t,q_{\text{accept}}}$$

...

NP-hardness of SAT (10)

Proof (continued).

4. follow the rules in δ:

$$Trans := \bigwedge_{t \in Steps} \left(state_{t,q_{\mathrm{accept}}} \vee state_{t,q_{\mathrm{reject}}} \vee \bigvee_{R \in \delta} Rule_{t,R} \right)$$

where. . .

. . .

NP-hardness of SAT (11)

Proof (continued).

4. follow the rules in δ (continued):

$$
\begin{aligned}
& \mathit{Rule}_{t,\langle\langle q,a\rangle,\langle q',a',D\rangle\rangle} := \\
& \qquad \mathit{state}_{t,q} \land \mathit{state}_{t+1,q'} \land \\
& \qquad \bigwedge_{i \in \mathit{Pos}} \left(\mathit{head}_{t,i} \rightarrow \left(\mathit{tape}_{t,i,a} \land \mathit{head}_{t+1,i+D} \land \mathit{tape}_{t+1,i,a'}\right)\right) \land \\
& \qquad \bigwedge_{i \in \mathit{Pos}} \bigwedge_{a'' \in \Gamma} \left(\left(\neg \mathit{head}_{t,i} \land \mathit{tape}_{t,i,a''}\right) \rightarrow \mathit{tape}_{t+1,i,a''}\right)
\end{aligned}
$$

- For $i + D$, interpret $i + \mathrm{R} \rightsquigarrow i + 1$, $i + \mathrm{L} \rightsquigarrow \max\{1, i - 1\}$.
- special case: *tape* and *head* variables with a tape index $i + D$ outside of *Pos* are replaced by $\bot$; likewise all variables with a time index outside of *Steps*.

...

NP-hardness of SAT (12)

Proof (continued).

Putting the pieces together:

Set $f(w) := \mathit{Valid} \land \mathit{Init} \land \mathit{Accept} \land \mathit{Trans}$.

NP-hardness of SAT (12)

Proof (continued).

Putting the pieces together:

Set $f(w) := \mathit{Valid} \land \mathit{Init} \land \mathit{Accept} \land \mathit{Trans}$.

- $f(w)$ can be constructed in time polynomial in $|w|$.
- $w \in A$ iff M accepts w in $p(|w|)$ steps
  iff $f(w)$ is satisfiable
  iff $f(w) \in \mathrm{SAT}$

$\rightsquigarrow$ $A \leq_{\mathrm{p}} \mathrm{SAT}$

NP-hardness of SAT (12)

Proof (continued).

Putting the pieces together:

Set $f(w) := \mathit{Valid} \wedge \mathit{Init} \wedge \mathit{Accept} \wedge \mathit{Trans}$.

- $f(w)$ can be constructed in time polynomial in $|w|$.
- $w \in A$ iff M accepts w in $p(|w|)$ steps
  iff $f(w)$ is satisfiable
  iff $f(w) \in \mathrm{SAT}$

$\rightsquigarrow$ $A \leq_{\mathsf{p}} \mathrm{SAT}$

Since $A \in \mathrm{NP}$ was arbitrary, this is true for every $A \in \mathrm{NP}$.

NP-hardness of SAT (12)

Proof (continued).

Putting the pieces together:

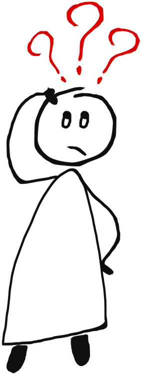
Set $f(w) := \mathit{Valid} \land \mathit{Init} \land \mathit{Accept} \land \mathit{Trans}$.

- $f(w)$ can be constructed in time polynomial in $|w|$.
- $w \in A$ iff M accepts w in $p(|w|)$ steps
  iff $f(w)$ is satisfiable
  iff $f(w) \in \mathrm{SAT}$

$\rightsquigarrow$ $A \leq_{\mathrm{p}} \mathrm{SAT}$

Since $A \in \mathrm{NP}$ was arbitrary, this is true for every $A \in \mathrm{NP}$.
Hence SAT is NP-hard and thus also NP-complete. □

Questions



Questions?

3SAT

More Propositional Logic: Conjunctive Normal Form

- A literal is an atomic proposition X or its negation $\neg X$.

More Propositional Logic: Conjunctive Normal Form

- A literal is an atomic proposition X or its negation $\neg X$.
- A clause is a disjunction of literals,
  e.g. $(X \vee \neg Y \vee Z)$

More Propositional Logic: Conjunctive Normal Form

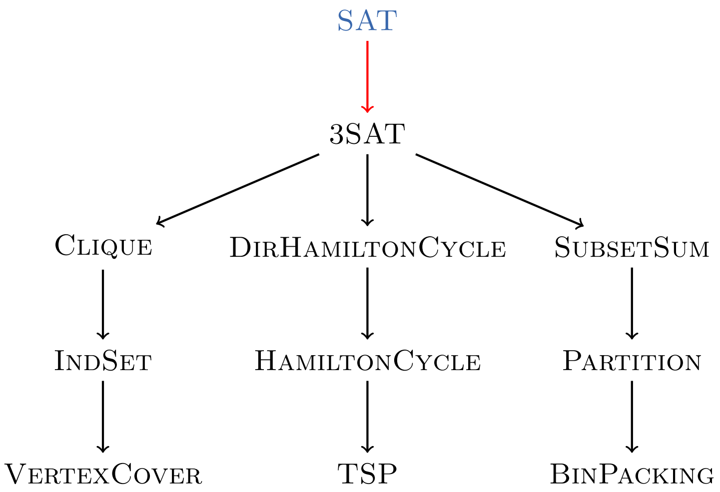
- A literal is an atomic proposition X or its negation $\neg X$.
- A clause is a disjunction of literals,
  e.g. $(X \lor \neg Y \lor Z)$
- A formula in conjunctive normal form
  is a conjunction of clauses,
  e.g. $((X \lor \neg Y \lor Z) \land (\neg X \lor \neg Z) \land (X \lor Y))$

Exercise (slido)

Which of the following formulas are in conjunctive normal form?

- $((X \land \neg Y \land Z) \lor (\neg X \land \neg Z))$
- $(X \lor \neg Y \lor Z)$
- $((\neg X \lor \neg Z) \land \neg(X \lor Y))$
- $((\neg Y \lor X) \land (Y \lor \neg Z))$

SAT $\leq_p$ 3SAT



SAT and 3SAT

Definition (Reminder: SAT)

The problem SAT (satisfiability) is defined as follows:

Given: a propositional logic formula φ

Question: Is φ satisfiable?

Definition (3SAT)

The problem 3SAT is defined as follows:

Given: a propositional logic formula φ in conjunctive normal form with at most three literals per clause

Question: Is φ satisfiable?

3SAT is NP-Complete (1)

Theorem (3SAT is NP-Complete)

3SAT *is NP-complete.*

3SAT is NP-Complete (2)

Proof.

$\mathrm{3SAT} \in$ NP: guess and check.

$\mathrm{3SAT}$ is NP-hard: We show $\mathrm{SAT} \leq_{\mathsf{p}} \mathrm{3SAT}$.

- Let φ be the given input for SAT. Let *Sub*(φ) denote the set of subformulas of φ, including φ itself.

$\mathrm{3SAT}$ is NP-Complete (2)

Proof.

$\mathrm{3SAT} \in$ NP: guess and check.

$\mathrm{3SAT}$ is NP-hard: We show $\mathrm{SAT} \leq_{\mathsf{p}} \mathrm{3SAT}$.

- Let φ be the given input for SAT. Let $\mathit{Sub}(\varphi)$ denote the set of subformulas of φ, including φ itself.
- For all $\psi \in \mathit{Sub}(\varphi)$, we introduce a new proposition X_{ψ}.

3SAT is NP-Complete (2)

Proof.

3SAT ∈ NP: guess and check.

3SAT is NP-hard: We show $\mathrm{SAT} \leq_{\mathrm{p}} \mathrm{3SAT}$.

- Let φ be the given input for SAT. Let $\mathit{Sub}(\varphi)$ denote the set of subformulas of φ, including φ itself.
- For all $\psi \in \mathit{Sub}(\varphi)$, we introduce a new proposition X_ψ.
- For each new proposition X_ψ, define the following auxiliary formula χ_ψ:
  - If $\psi = A$ for an atom A: $\chi_\psi = (X_\psi \leftrightarrow A)$
  - If $\psi = \neg\psi'$: $\chi_\psi = (X_\psi \leftrightarrow \neg X_{\psi'})$
  - If $\psi = (\psi' \wedge \psi'')$: $\chi_\psi = (X_\psi \leftrightarrow (X_{\psi'} \wedge X_{\psi''}))$
  - If $\psi = (\psi' \vee \psi'')$: $\chi_\psi = (X_\psi \leftrightarrow (X_{\psi'} \vee X_{\psi''}))$

...

3SAT is NP-Complete (3)

Proof (continued).

- Consider the conjunction of all these auxiliary formulas, $\chi_{\text{all}} := \bigwedge_{\psi \in \mathit{Sub}(\varphi)} \chi_\psi$.

3SAT is NP-Complete (3)

Proof (continued).

- Consider the conjunction of all these auxiliary formulas, $\chi_{\mathsf{all}} := \bigwedge_{\psi \in \mathit{Sub}(\varphi)} \chi_\psi$.
- Every variable assignment $\mathcal{I}$ for the original variables can be extended to a variable assignment $\mathcal{I}'$ under which χ_{all} is true in exactly one way: for each $\psi \in \mathit{Sub}(\varphi)$, set $\mathcal{I}'(X_\psi) = T$ iff $\mathcal{I} \models \psi$.

3SAT is NP-Complete (3)

Proof (continued).

- Consider the conjunction of all these auxiliary formulas, $\chi_{\mathsf{all}} := \bigwedge_{\psi \in Sub(\varphi)} \chi_\psi$.
- Every variable assignment $\mathcal{I}$ for the original variables can be extended to a variable assignment $\mathcal{I}'$ under which χ_{all} is true in exactly one way: for each $\psi \in Sub(\varphi)$, set $\mathcal{I}'(X_\psi) = T$ iff $\mathcal{I} \models \psi$.
- It follows that φ is satisfiable iff $(\chi_{\mathsf{all}} \wedge X_\varphi)$ is satisfiable.

3SAT is NP-Complete (3)

Proof (continued).

- Consider the conjunction of all these auxiliary formulas, $\chi_{\mathsf{all}} := \bigwedge_{\psi \in Sub(\varphi)} \chi_\psi$.
- Every variable assignment $\mathcal{I}$ for the original variables can be extended to a variable assignment $\mathcal{I}'$ under which χ_{all} is true in exactly one way: for each $\psi \in Sub(\varphi)$, set $\mathcal{I}'(X_\psi) = T$ iff $\mathcal{I} \models \psi$.
- It follows that φ is satisfiable iff $(\chi_{\mathsf{all}} \wedge X_\varphi)$ is satisfiable.
- This formula can be computed in linear time.

3SAT is NP-Complete (3)

Proof (continued).

- Consider the conjunction of all these auxiliary formulas, $\chi_{\mathsf{all}} := \bigwedge_{\psi \in \mathit{Sub}(\varphi)} \chi_\psi$.
- Every variable assignment $\mathcal{I}$ for the original variables can be extended to a variable assignment $\mathcal{I}'$ under which χ_{all} is true in exactly one way: for each $\psi \in \mathit{Sub}(\varphi)$, set $\mathcal{I}'(X_\psi) = T$ iff $\mathcal{I} \models \psi$.
- It follows that φ is satisfiable iff $(\chi_{\mathsf{all}} \wedge X_\varphi)$ is satisfiable.
- This formula can be computed in linear time.
- It can also be converted to 3-CNF in linear time because it is the conjunction of constant-size parts involving at most three variables each. (Each part can be converted to 3-CNF independently.)

3SAT is NP-Complete (3)

Proof (continued).

- Consider the conjunction of all these auxiliary formulas, $\chi_{\mathrm{all}} := \bigwedge_{\psi \in Sub(\varphi)} \chi_\psi$.
- Every variable assignment $\mathcal{I}$ for the original variables can be extended to a variable assignment $\mathcal{I}'$ under which χ_{all} is true in exactly one way: for each $\psi \in Sub(\varphi)$, set $\mathcal{I}'(X_\psi) = T$ iff $\mathcal{I} \models \psi$.
- It follows that φ is satisfiable iff $(\chi_{\mathrm{all}} \wedge X_\varphi)$ is satisfiable.
- This formula can be computed in linear time.
- It can also be converted to 3-CNF in linear time because it is the conjunction of constant-size parts involving at most three variables each. (Each part can be converted to 3-CNF independently.)
- Hence, this describes a polynomial-time reduction.

□

Restricted 3SAT

Note: 3SAT remains NP-complete if we also require that

- every clause contains exactly three literals and
- a clause may not contain the same literal twice

Restricted 3SAT

Note: 3SAT remains NP-complete if we also require that

- every clause contains exactly three literals and
- a clause may not contain the same literal twice

Idea:

- remove duplicated literals from each clause.

Restricted 3SAT

Note: 3SAT remains NP-complete if we also require that

- every clause contains exactly three literals and
- a clause may not contain the same literal twice

Idea:

- remove duplicated literals from each clause.
- add new variables: X, Y, Z
- add new clauses: $(X \lor Y \lor Z)$, $(X \lor Y \lor \neg Z)$, $(X \lor \neg Y \lor Z)$, $(\neg X \lor Y \lor Z)$, $(X \lor \neg Y \lor \neg Z)$, $(\neg X \lor Y \lor \neg Z)$, $(\neg X \lor \neg Y \lor Z)$

Restricted 3SAT

Note: 3SAT remains NP-complete if we also require that

- every clause contains exactly three literals and
- a clause may not contain the same literal twice

Idea:

- remove duplicated literals from each clause.
- add new variables: X, Y, Z
- add new clauses: $(X \vee Y \vee Z)$, $(X \vee Y \vee \neg Z)$, $(X \vee \neg Y \vee Z)$, $(\neg X \vee Y \vee Z)$, $(X \vee \neg Y \vee \neg Z)$, $(\neg X \vee Y \vee \neg Z)$, $(\neg X \vee \neg Y \vee Z)$

$\rightsquigarrow$ satisfied if and only if X, Y, Z are all true

Restricted 3SAT

Note: 3SAT remains NP-complete if we also require that

- every clause contains exactly three literals and
- a clause may not contain the same literal twice

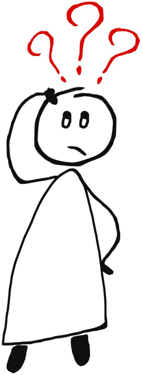
Idea:

- remove duplicated literals from each clause.
- add new variables: X, Y, Z
- add new clauses: $(X \vee Y \vee Z)$, $(X \vee Y \vee \neg Z)$, $(X \vee \neg Y \vee Z)$, $(\neg X \vee Y \vee Z)$, $(X \vee \neg Y \vee \neg Z)$, $(\neg X \vee Y \vee \neg Z)$, $(\neg X \vee \neg Y \vee Z)$

$\rightsquigarrow$ satisfied if and only if X, Y, Z are all true

- fill up clauses with fewer than three literals with $\neg X$ and if necessary additionally with $\neg Y$

Questions



Questions?

Summary

Summary

- Thousands of important problems are NP-complete.
- The satisfiability problem of propositional logic (SAT) is NP-complete.
- Proof idea for NP-hardness:
  - Every problem in NP can be solved by an NTM in polynomial time $p(|w|)$ for input w.
  - Given a word w, construct a propositional logic formula φ that encodes the computation steps of the NTM on input w.
  - Construct φ so that it is satisfiable if and only if there is an accepting computation of length $p(|w|)$.
- Usually (as seen for 3SAT), the easiest way to show that another problem is NP-complete is to
  - show that it is in NP with a guess-and-check algorithm, and
  - polynomially reduce a known NP-complete to it.