

Theory of Computer Science

E3. Cook-Levin Theorem

Malte Helmert

University of Basel

May 25, 2016

Overview: Course

contents of this course:

- logic ✓
 - ▷ How can knowledge be represented?
How can reasoning be automated?
- automata theory and formal languages ✓
 - ▷ What is a computation?
- computability theory ✓
 - ▷ What can be computed at all?
- complexity theory
 - ▷ What can be computed efficiently?

Overview: Complexity Theory

Complexity Theory

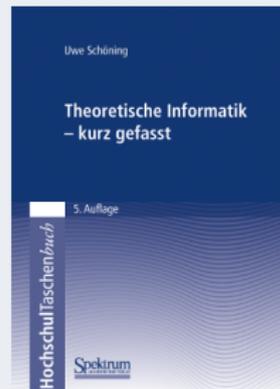
- E1. Motivation and Introduction
- E2. P, NP and Polynomial Reductions
- E3. **Cook-Levin Theorem**
- E4. Some NP-Complete Problems, Part I
- E5. Some NP-Complete Problems, Part II

Further Reading (German)

Literature for this Chapter (German)

Theoretische Informatik – kurz gefasst
by Uwe Schöning (5th edition)

- Chapter 3.2

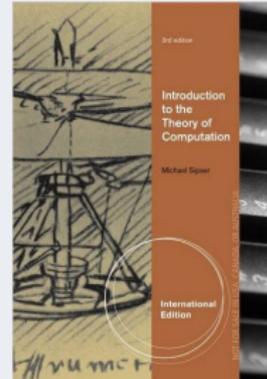


Further Reading (English)

Literature for this Chapter (English)

Introduction to the Theory of Computation
by Michael Sipser (3rd edition)

- Chapter 7.4



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: $A \leq_p \text{SAT}$ for all $A \in \text{NP}$.

NP-hardness of SAT (1)

Proof (continued).

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

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 $A \in \text{NP}$, there is an NTM M and a polynomial p such that M accepts 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 end configuration can be reached** in this time. ...

NP-hardness of SAT (3)

Proof (continued).

Let $M = \langle Q, \Sigma, \Gamma, \delta, q_0, \square, E \rangle$ be an NTM for A ,
and let p be a polynomial bounding the computation time of M .

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

NP-hardness of SAT (3)

Proof (continued).

Let $M = \langle Q, \Sigma, \Gamma, \delta, q_0, \square, E \rangle$ be an NTM for A ,
and let p be a polynomial bounding the computation time of M .

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

We number the tape positions with integers (positive and negative) such that the TM head initially is on position 1.

NP-hardness of SAT (3)

Proof (continued).

Let $M = \langle Q, \Sigma, \Gamma, \delta, q_0, \square, E \rangle$ be an NTM for A ,
and let p be a polynomial bounding the computation time of M .

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

We number the tape positions with integers (positive and negative) 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

$Pos = \{-p(n) + 1, -p(n) + 2, \dots, -1, 0, 1, \dots, p(n) + 1\}$.

NP-hardness of SAT (3)

Proof (continued).

Let $M = \langle Q, \Sigma, \Gamma, \delta, q_0, \square, E \rangle$ be an NTM for A ,
and let p be a polynomial bounding the computation time of M .

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

We number the tape positions with integers (positive and negative) 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

$$Pos = \{-p(n) + 1, -p(n) + 2, \dots, -1, 0, 1, \dots, p(n) + 1\}.$$

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

↔ 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 TM configurations**
- that **begins with the start configuration**,
- **reaches an accepting configuration**
- and **follows the TM rules in δ**

NP-hardness of SAT (6)

Proof (continued).

Auxiliary formula:

$$\text{oneof } X := \left(\bigvee_{x \in X} x \right) \wedge \neg \left(\bigvee_{x \in X} \bigvee_{y \in X \setminus \{x\}} (x \wedge y) \right)$$

Note: we use the symbol \perp to refer to some formula which is false under every interpretation (e.g., $(A \wedge \neg A)$, where A is an arbitrary proposition).

...

NP-hardness of SAT (7)

Proof (continued).

1. describe the configurations of the TM:

$$\text{Valid} := \bigwedge_{t \in \text{Steps}} \left(\text{oneof} \{ \text{state}_{t,q} \mid q \in Q \} \wedge \right. \\ \left. \text{oneof} \{ \text{head}_{t,i} \mid i \in \text{Pos} \} \wedge \right. \\ \left. \bigwedge_{i \in \text{Pos}} \text{oneof} \{ \text{tape}_{t,i,a} \mid a \in \Gamma \} \right)$$

...

NP-hardness of SAT (8)

Proof (continued).

2. begin in the start configuration

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

...

NP-hardness of SAT (9)

Proof (continued).

3. reach an accepting configuration

$$\text{Accept} := \bigvee_{t \in \text{Steps}} \bigvee_{q_e \in E} \text{state}_{t, q_e}$$

...

NP-hardness of SAT (10)

Proof (continued).

4. follow the rules in δ :

$$Trans := \bigwedge_{t \in Steps} \left(\bigvee_{q_e \in E} state_{t, q_e} \vee \bigvee_{R \in \delta} Rule_{t, R} \right)$$

where...

...

NP-hardness of SAT (11)

Proof (continued).

4. follow the rules in δ (continued): $Rule_{t, \langle \langle q, a \rangle, \langle q', a', y \rangle \rangle} :=$ $state_{t,q} \wedge state_{t+1,q'} \wedge$ $\bigwedge_{i \in Pos} (head_{t,i} \rightarrow tape_{t,i,a} \wedge head_{t+1,i+y} \wedge tape_{t+1,i,a'}) \wedge$ $\bigwedge_{i \in Pos} \bigwedge_{a'' \in \Gamma} (\neg head_{t,i} \wedge tape_{t,i,a''} \rightarrow tape_{t+1,i,a''})$

- For y , interpret $L \rightsquigarrow -1$, $N \rightsquigarrow 0$, $R \rightsquigarrow +1$.
- **special case:** $tape$ and $head$ variables with a tape index $i + y$ outside of Pos are replaced by \perp ; likewise all variables with a time index outside of $Steps$.

NP-hardness of SAT (12)

Proof (continued).

Putting the pieces together:

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

NP-hardness of SAT (12)

Proof (continued).

Putting the pieces together:

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

- $f(w)$ 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 $f(w) \in \text{SAT}$

$\rightsquigarrow A \leq_p \text{SAT}$

NP-hardness of SAT (12)

Proof (continued).

Putting the pieces together:

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

- $f(w)$ 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 $f(w) \in \text{SAT}$

$\rightsquigarrow A \leq_p \text{SAT}$

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

NP-hardness of SAT (12)

Proof (continued).

Putting the pieces together:

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

- $f(w)$ 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 $f(w) \in \text{SAT}$

$\rightsquigarrow A \leq_p \text{SAT}$

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

Questions



Questions?

Summary

Summary

- 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 $p(|w|)$.