

CS-E4530

Computational Complexity Theory

Janne H. Korhonen

Aalto University

Spring 2018

Lecture 6:

Cook–Levin Theorem

NP-complete Problems

- **Last lecture:**
 - We saw that TMSAT is NP-complete
 - Definition tied directly to the definition of NP
 - Does not really tell us anything new about NP
- **This lecture:**
 - Prove that a problem called **CNF-SAT** is NP-complete
 - First example of a **natural** NP-complete problem
 - Starting point for further NP-completeness proofs

Cook-Levin Theorem

- **One of the founding results of computational complexity**
 - CNF-SAT is NP-complete
 - Named after Stephen Cook and Leonid Levin
 - Both independently proved the theorem around 1971



Stephen Cook



Leonid Levin

Boolean Satisfiability

Boolean Formulas

- **Boolean formula is built from the following primitives:**
 - **Variables** x_1, x_2, \dots, x_n
 - **Operators** AND (\wedge), OR (\vee), and NOT (\neg)
 - **Example:** $\varphi = (x_1 \wedge x_2) \vee (x_2 \wedge x_3) \vee (x_3 \wedge x_1)$

Definition (Boolean formulas, recursive definition)

The set of **Boolean formulas** over variables x_1, x_2, \dots, x_k is defined as follows:

- x_i is a Boolean formula for any $i = 1, 2, \dots, n$.
- If φ is Boolean formula, then $\neg\varphi$ is Boolean formula.
- If φ and ψ are Boolean formulas, then $\varphi \wedge \psi$ and $\varphi \vee \psi$ are Boolean formulas.

Value of a Boolean Formula

- An assignment gives value 1 (true) or 0 (false) to each variable
 - Semantics of NOT, AND and OR are defined in the obvious way

Definition (Value of a boolean formula)

Let $z = (z_1, z_2, \dots, z_n) \in \{0, 1\}^n$ be an assignment. The **value $\varphi(z)$ of formula φ under assignment z** is defined as follows:

- If $\varphi = x_i$, then $\varphi(z) = z_i$.
- If $\varphi = \neg\psi$, then $\varphi(z) = 1 - \psi(z)$.
- If $\varphi = \psi_1 \wedge \psi_2$, then $\varphi(z) = 1$ if $\psi_1(z) = \psi_2(z) = 1$, and $\varphi(z) = 0$ otherwise.
- If $\varphi = \psi_1 \vee \psi_2$, then $\varphi(z) = 1$ if $\psi_1(z) = 1$ or $\psi_2(z) = 1$, and $\varphi(z) = 0$ otherwise.

Value of a Boolean Formula

- **Assignment** z **satisfies** formula φ if $\varphi(z) = 1$
 - A formula is **satisfiable** if there is a satisfying assignment
 - A formula is **unsatisfiable** otherwise

- **Examples:**
 - $\varphi = (x_1 \wedge x_2) \vee (x_2 \wedge x_3) \vee (x_3 \wedge x_1)$
 - φ is satisfiable

 - $\psi = (x_1 \vee \neg x_2) \wedge \neg x_1 \wedge x_2$
 - ψ is unsatisfiable

Conjunctive Normal Form

- A formula in **conjunctive normal form** is a formula that is an **AND** of **ORs**:

- **Example:** $(x_1 \vee x_2) \wedge (x_2 \vee \neg x_3) \wedge (x_3 \vee \neg x_1)$

- **Formally:**

- A CNF formula is a formula of form

$$\bigwedge_{i=1}^m \left(\bigvee_{j=1}^k \ell_{i,j} \right),$$

where each $\ell_{i,j}$ is either a x or $\neg x$ for some variable x

- Terms $\ell_{i,j}$ are called **literals**
 - Terms $\bigvee_{j=1}^k \ell_{i,j}$ are called **clauses**

CNF-SAT and k-SAT

Definition (CNF-SAT)

- **Instance:** A CNF formula φ .
- **Question:** Is φ satisfiable?

Definition (k-SAT)

- **Instance:** A CNF formula φ such that clause in φ has at most k literals.
- **Question:** Is φ satisfiable?

- **2-SAT instance:** $(x_1 \vee x_2) \wedge (x_2 \vee \neg x_3) \wedge (x_3 \vee \neg x_1)$
- **3-SAT instance:** $(x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4)$
- **4-SAT instance:** $(x_1 \vee x_2 \vee x_3 \vee \neg x_4) \wedge (x_2 \vee \neg x_3 \vee x_4)$

CNF-SAT and NP

Theorem

CNF-SAT is in NP.

- **Proof:** CNF-SAT has a polynomial-time verifier
 - **Input:** a formula φ over variables x_1, x_2, \dots, x_n
 - **Certificate:** an assignment $z \in \{0, 1\}^n$
 - **Verification algorithm:** compute the value $\varphi(z)$, accept if $\varphi(z) = 1$

Corollary

For any fixed $k \geq 1$, k -SAT is in NP.

CNF Formulas and Boolean Functions

Universality of CNF Formulas

- **CNF formulas can express all Boolean functions**
 - May require exponential number of clauses
 - This does not matter: we want to use this construction for **constant** number of variables

Lemma

Let $f: \{0, 1\}^n \rightarrow \{0, 1\}$ be a Boolean function. Then there is a CNF formula φ over n variables with at most 2^n clauses such that $\varphi(z) = f(z)$ for all $z \in \{0, 1\}^n$.

Universality of CNF Formulas: Proof

- **For each $z \in \{0, 1\}^n$, we construct clause C_z :**
 - Let $\ell_i = x_i$ if $z_i = 0$, and $\ell_i = \neg x_i$ if $z_i = 1$
 - Let $C_z = \bigvee_{i=1}^n \ell_i$
 - We now have $C_z(y) = 0$ if $z = y$, and $C_z(y) = 1$ if $z \neq y$
- **For any $f: \{0, 1\}^n \rightarrow \{0, 1\}$, we construct formula φ :**
 - Let $\varphi_f = \bigwedge_{z: f(z)=0} C_z$
 - If $f(y) = 0$, then y does not satisfy the clause C_y in φ_f
 - If $f(y) = 1$, then y satisfies all clauses C_y in φ_f
 - Thus, we have $\varphi_f(y) = f(y)$ for all $y \in \{0, 1\}^n$

Cook–Levin Theorem

Cook-Levin Theorem

Theorem

CNF-SAT is NP-complete.

- **We have:** CNF-SAT is in NP
- **Next:** CNF-SAT is NP-hard

Cook–Levin Theorem: Proof

- **General template for the proof:**
 - Let $L \in \text{NP}$ be a language
 - We prove that there is a polynomial-time reduction from L to CNF-SAT
- **The only thing we know about L is that it is in NP**
 - There exists a verifier M for L
 - For any $x \in L$, there is a certificate for x of length at most $q(|x|)$, for some polynomial q
 - M runs on input (x, u) in time $p(|x|)$ for some polynomial p with $q(n) \leq p(n)$
 - M uses at most $p(|x|)$ positions on each tape
 - We may assume M has one working tape, uses alphabet $\{\triangleright, \square, 0, 1\}$

Execution Tables

- **Execution of M on input (x, u) can be viewed as a table:**
 - Row i describes the state of M , the positions of heads and the contents of the tapes after step i
 - Since M runs in time $p(|x|)$, each row needs to store at most $1 + 2 \cdot 3 \cdot p(|x|)$ entries
 - At most $p(|x|)$ rows
- **Table can be encoded as binary:**
 - Use $|Q|$ bits for state
 - Use 2 bits for each tape position
 - Use 1 bit for each possible head position

Execution Tables

- **Execution table is accepting if:**
 - State bit on the last row corresponds to the halting state
 - The encoding of the output tape on the last row corresponds to $\triangleright 1 \square \square \dots$
- **By definition:**
 - M has accepting execution table if and only if M accepts

Cook–Levin Theorem: Proof

- **Proof overview:**

- Let x be an instance of L
- We construct a CNF-SAT formula φ_x over $S = p(|x|) \cdot (|Q| + 9p(|x|))$ variables
- Assignment z to φ_x encodes an execution table of M :
 - φ_x is satisfied by z only if z encodes a valid execution table
 - φ_x is satisfied by z only if z is an execution table on input (x, u) for some $u \in \{0, 1\}^*$
 - φ_x is satisfied by z only if z is an accepting execution table

Cook–Levin Theorem: Proof

- **Clauses of φ_x are constructed to locally enforce the constraints**
 - We could use the universality lemma to directly construct a CNF formula to enforce that the variables encode an accepting execution table
 - This would give **exponential** size in terms of $|x|$
 - Need to be more careful to get polynomial size
- **Basic idea: encode local constraints**

Cook–Levin Theorem: Proof

- **Clauses of φ_x enforce the starting and halting:**
 - x is written to the beginning of input tape
 - All heads start at position 1 and first state is q_0
 - State on the last row is q_h
 - Content of the output tape on the last row is $\triangleright 1 \square \square \dots$
- **These can be encoded by a conjunction of $O(p(|x|))$ width-1 clauses**

Cook-Levin Theorem: Proof

- **Clauses of φ_x enforce consistency of the table:**
 - Only single head position and state for each row
 - If head is not at position j , then the tape symbols at position j do not change between steps
 - If head is at position j , the tape symbols and the head positions change correctly between steps
 - The machine state is changes correctly between steps
- **Each of these conditions can be viewed as a Boolean function on **constant** number of variables**
 - At most $c = 2|Q| + 3 \cdot 2 \cdot 6$ variables per constraint
 - Encode as a CNF with 2^c clauses using universality lemma
 - About $O(p(|x|)^2)$ constraints needed

Cook–Levin Theorem: Proof

- **The final CNF formula φ_x is conjunction of all constraints:**
 - A conjunction of CNF formulas is a CNF formula
- **By construction, this gives us a reduction from L to CNF-SAT**
 - $x \in L$ if and only if φ_x is satisfiable
 - φ_x can be constructed in polynomial time

Closing Remarks

CNF-SAT: Discussion

- **CNF-SAT is a relevant problem in practice**
 - **Problem-specific** reductions to CNF-SAT can be much more compact than the general reduction given by Cook–Levin theorem
 - E.g. Intel has used CNF-SAT solvers to verify and optimise processor designs
 - Highly efficient CNF-SAT solvers are available as open-source software
 - For many difficult optimisation problems, reducing to CNF-SAT and applying off-the-shelf solvers can be much faster than anything you implement yourself

Lecture 6: Summary

- CNF-SAT and k -SAT
- CNF-SAT is NP-complete