Complexity IBC028, Lecture 7

H. Geuvers

Institute for Computing and Information Sciences
Radboud University Nijmegen

Version: spring 2021

Outline

The Cook Levin Theorem

Theorem

SAT is NP-complete

Proof

- SAT $\in$ NP: for φ a boolean formula, the certificate is the satisfying assignment v; v is polynomial in $|\varphi|$ and checking $v(\varphi) = 1$ is also polynomial.
- SAT $\in$ NPH. For every $L \in$ NP we should find a polynomial f such that

$$\forall x (x \in L \iff f(x) \in \text{SAT}).$$

Let $L \in$ NP, so there is a polynomial A such that

$$x \in L \iff \exists y \in \{0,1\}^* (|y| \text{ polynomial in } |x| \wedge A(x,y) = 1)$$

The f we construct will mimick A.

Encoding a Turing Machine as a boolean formula (I)

A is given by a Turing Machine $M = (Q, \Sigma, \delta)$ and we have

$$A(x, y) = 1 \iff M \text{ halts in state } q_F \text{ on input } (x, y).$$

We will encode the operation of M on (x, y) as a boolean formula.

- A configuration of M is given by: a state q and tape content $a_1 \ldots a_k a_{k+1} \ldots a_n$ with q reading a_k. We encode this by

$$a_1 \ldots a_k q a_{k+1} \ldots a_n \in (Q \cup \Sigma)^*$$

- A is polynomial in $|x|$, so there is a polynomial P such that
  - computation of M on (x, y) takes $\leq P(|x|)$ steps,
  - computation of M on (x, y) uses $\leq P(|x|)$ symbols on tape.
- Introduce boolean variables to describe the configuration of M after i steps. Intended meaning:

$$p_{i,j,a} = \text{true} \iff \text{after } i \text{ steps, there is an } a \text{ on position } j$$

- The number of boolean variables is bound by $P|x| \times (P|x| + 1) \times (|\Sigma| + |Q|)$, so polynomial in $|x|$.

Encoding a Turing Machine as a boolean formula (II)

We encode the intended meaning of $p_{i,j,a}$ by writing a (vast) number of boolean formulas.

- For readability, we also use $\to$ as a boolean connective.
- We use $v(p_{i,j,a}) \in \{\mathrm{true}, \mathrm{false}\}$ to distinguish the satisfiability problem we construct from the tape content.

We have three groups of formulas.

1. formulas that describe properties that a tape configuration should obey
2. formulas describing the transition function δ of the Turing Machine
3. formulas that describe the initial cofiguration of the Turing Machine, with input on the tape, and the final accepting configuration

Encoding a Turing Machine as a boolean formula (III)

(1) Boolean formulas to describe tape configurations

$$\bigwedge_{i,j}((\bigvee_{a\in\Sigma\cap Q} p_{i,j,a}) \wedge \bigwedge_{a,b\in\Sigma\cup Q, a\neq b} (\neg p_{i,j,a} \vee \neg p_{i,j,b}))$$

- On every i (every time step) each j (every tape location) holds an $a \in \Sigma \cup Q$,
- On every i (every time step) each j (every tape location) holds at most one $a \in \Sigma \cup Q$.

Note that both i and j are bound by $P(|x|)$, so the size of this formula is polynomial in $|x|$.

Encoding a Turing Machine as a boolean formula (IV)

(2) Boolean formulas describing the transition function δ.

Suppose that we have $\delta(q, a) = (q', b, R)$.

We add, for every i, j and every $c \in \Sigma$ the formula

$$(p_{i,j,a} \wedge p_{i,j+1,q} \wedge p_{i,j+2,c}) \rightarrow (p_{i+1,j,b} \wedge p_{i+1,j+1,c} \wedge p_{i+1,j+2,q'})$$

The rest of the tape remains intact so we add, for every $d \in \Sigma$, and for every $k < j$ and every $k > j+2$ the formula

$$(p_{i,j,a} \wedge p_{i,j+1,q} \wedge p_{i,j+2,c}) \rightarrow (p_{i+1,k,d} \leftrightarrow p_{i,k,d})$$

Note that again, i, j and k are bound by $P(|x|)$, so the size of this formula is polynomial in $|x|$.

This is repeated for all transition steps of δ.

Encoding a Turing Machine as a boolean formula (V)

(3) Boolean formulas describing the initial configuration of the Turing Machine with input x (and certificate y "to be guessed"), and the accepting condition.

- $p_{0,1,q_0}$
- $p_{0,j+1,0}$ for all j-positions in x for which $x_j = 0$
- $p_{0,j+1,1}$ for all j-positions in x for which $x_j = 1$
- $p_{0,|x|+2,M}$ marking the end of input x, for marking symbol M
- $p_{0,|x|+2+j,0} \vee p_{0,|x|+2+j,1}$ for all j-positions in y, which should be either 0 or 1
- $p_{0,j,\sqcup}$ for all other tape positions, for the "blank" symbol $\sqcup$.
- $\bigvee_{i,j} p_{i,j,q_F}$ describing that M has reached the final state q_F.

Note that again, i, j are bound by $P(|x|)$, so the size of this formula is polynomial in $|x|$.

Encoding a Turing Machine as a boolean formula (VI)

Given Turing Machine M (that implements algorithm A), and input x, we denote by $f(x)$ the Boolean formula that is the conjunction of all the formulas that we have just described.

We have the following:

$f(x) \in \text{SAT}$

$\iff$ the $p_{0,j,a}$ describe a valid initial configuration with x as input and some choice for y
and $\forall i > 0$, the $p_{i,j,a}$ describe a configuration of M after i steps
and $\bigvee_{i,j} p_{i,j,q_F} = \text{true}$
(at a certain point we arrive at state q_F)

$\iff$ $\exists y(M$ with tape input (x, y) halts in $q_F)$

$\iff$ $\exists y(A(x, y) = 1)$.

So: For every $L \in \text{NP}(L \leq_P \text{SAT})$.

So: $\text{SAT} \in \text{NPH}$ and so $\text{SAT} \in \text{NPC}$.

CNF-SAT is NP-complete

The construction of f in the Cook-Levin proof can be adapted a bit so that $f(x)$ is a CNF-formula.

Steps (1) and (3) already create a CNF. For Step (2):

$$(p_{i,j,a} \wedge p_{i,j+1,q} \wedge p_{i,j+2,c}) \rightarrow (p_{i+1,j,b} \wedge p_{i+1,j+1,c} \wedge p_{i+1,j+2,q'})$$

is equivalent to the three clauses

$$\neg p_{i,j,a} \vee \neg p_{i,j+1,q} \vee \neg p_{i,j+2,c} \vee p_{i+1,j,b}$$
$$\neg p_{i,j,a} \vee \neg p_{i,j+1,q} \vee \neg p_{i,j+2,c} \vee p_{i+1,j+1,c}$$
$$\neg p_{i,j,a} \vee \neg p_{i,j+1,q} \vee \neg p_{i,j+2,c} \vee p_{i+1,j+2,q'}$$

$$(p_{i,j,a} \wedge p_{i,j+1,q} \wedge p_{i,j+2,c}) \rightarrow (p_{i+1,k,d} \leftrightarrow p_{i,k,d})$$

is equivalent to the two clauses

$$\neg p_{i,j,a} \vee \neg p_{i,j+1,q} \vee \neg p_{i,j+2,c} \vee p_{i+1,k,d} \vee \neg p_{i,k,d}$$
$$\neg p_{i,j,a} \vee \neg p_{i,j+1,q} \vee \neg p_{i,j+2,c} \vee \neg p_{i+1,k,d} \vee p_{i,k,d}$$

So, for every $L \in \text{NP}(L \leq_P \text{CNF-SAT})$ and so: CNF-SAT $\in$ NPH.

Why SAT is important

SAT is NP-complete, but

- nevertheless there are very powerful tools that can solve large SAT problems (and even a bit more) very quickly
- many decision problems can be cast as a satisfiability problem

Example: Bounded Model Checking

Consider the following algorithm that sorts a triple of booleans.

$$\begin{array}{llll} \texttt{if} & a_1 > a_2 & \texttt{then} & \texttt{swap}(a_1, a_2); \\ \texttt{if} & a_2 > a_3 & \texttt{then} & \texttt{swap}(a_2, a_3); \\ \texttt{if} & a_1 > a_2 & \texttt{then} & \texttt{swap}(a_1, a_2) \end{array}$$

Question: is this a correct sorting algorithm?
Introduce variables $a_{i,j}$ as values of a_i after j steps ($j = 0, 1, 2, 3$) and introduce boolean formulas to denote the steps in the algorithm. For the first step:

$$\begin{array}{rcl} (a_{1,0} \wedge \neg a_{2,0}) & \to & (a_{1,1} \leftrightarrow a_{2,0} \wedge a_{2,1} \leftrightarrow a_{1,0} \wedge a_{3,1} \leftrightarrow a_{3,0}) \\ \neg(a_{1,0} \wedge \neg a_{2,0}) & \to & (a_{1,1} \leftrightarrow a_{1,0} \wedge a_{2,1} \leftrightarrow a_{2,0} \wedge a_{3,1} \leftrightarrow a_{3,0}) \end{array}$$

Add a boolean formula that states that the algorithm is incorrect:

$$(a_{1,3} \wedge \neg a_{2,3}) \vee (a_{2,3} \wedge \neg a_{3,3})$$

The conjunction of these formulas is not satisfiable, so the algorithm is correct.

Course overview(I)

1. Recursive programs
   $\mathrm{fib}(n) = \Theta(\varphi^n)$ with $\varphi = \frac{1+\sqrt{5}}{2} \approx 1.618$
   Application: AVL-trees with k nodes have depth $\Theta(\log k)$.
2. Divide and Conquer algorithms
   If #steps on input of size n is $T(n)$, we have

$$T(n) = \Sigma_{\text{some } k, k<n} T(k) + f(n)$$

   How to derive a $g(n)$ such that $T(n) = \mathcal{O}(g(n))$?
   - Substitution method
   - Recursion tree method
   - Master Theorem method, especially for $T(n) = aT(\frac{n}{b}) + f(n)$.

   Applications:
   - Karatsuba multiplication of numbers: $\Theta(n^{\log_2 3}) \approx \Theta(n^{1.58})$.
   - The median of a list of numbers of length n, in $\Theta(n)$.
   - Matrix multiplication (and inversion): $\Theta(n^{\log_2 7}) \approx \Theta(n^{2.8})$.

Course overview(II)

3 P and NP; NP-hard, NP-complete

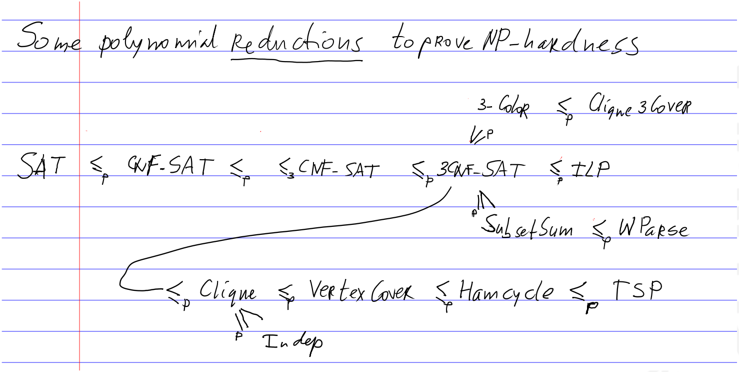
$$\text{P} := \{L \subseteq \{0,1\}^* \mid \exists A, A \text{ polynomial}, w \in L \iff A(w) = 1\}$$

$$\text{NP} := \{L \subseteq \{0,1\}^* \mid \exists A, A \text{ polynomial}, w \in L \iff \exists y \in \{0,1\}^*(|y| \text{ polynomial in } |w| \wedge A(w,y) = 1)\}$$

- $\text{NPH} := \{L \mid \forall L' \in \text{NP}(L' \leq_P L)\}$
- $\text{NPC} := \text{NP} \cap \text{NPH}$
- $L_1 \leq_P L_2$ if $\exists$ polynomial $f : \{0,1\}^* \to \{0,1\}^*(x \in L_1 \iff f(x) \in L_2)$
- (Theorem) If $L' \leq_P L$ and $L' \in \text{NPH}$, then $L \in \text{NPH}$.
- (Theorem) $\text{SAT} \in \text{NPC}$
- Whole list of NPC-problems

Course overview(III)

- Overview of NPC-problems



4 PSPACE

- Definition of PSPACE-problem, PSPACE-complete
- QBF and variants are PSPACE-complete