The Cook-Levin Theorem

CSC 463

February 28, 2020

NP-Completeness

- A problem A is **NP-Complete** if $A \in NP$ and every problem in NP reduces to A.
- Showing that A is **NP-Complete** provides evidence that A cannot have efficient (polynomial-time) algorithm.
- We saw a sequence of reductions that proved that various problems are NP-Complete, assuming the NP-completeness of 3SAT.

Cook-Levin Theorem

- A Boolean formula is satisfiable if you can assign truth values to $x_1, \ldots, x_n$ so that $\phi(x_1, \ldots, x_n)$ is true.
- Recall that a Boolean formula ϕ is in conjunctive normal form of $\phi(x_1, \ldots, x_n) = \bigwedge_{i=1}^{m} \phi_i$ where each ϕ_i is an OR of literals (a variable x or its complement $\bar{x}$). Each ϕ_i is called a **clause.**
- We remove the 3 variable per clause and conjunctive normal form restrictions for now and add it in later.

Theorem (SAT is NP-Complete)

Determining if a Boolean formula ϕ is satisfiable or not is an NP-Complete problem.

The Main Ideas

- SAT $\in NP$ since given a truth assignment for $x_1, \ldots, x_n$, you can check if $\phi(x_1, \ldots, x_n) = 1$ in polynomial time by evaluating the formula on a given assignment.
- We now need to show that there is a polynomial-time reduction $A \leq_p SAT$ for every A in NP.
- $A \in NP$ means that there is a non-deterministic Turing machine N running in $O(n^k)$ time that decides A. We will construct a Boolean formula ϕ that is satisfiable if and only if some branch of N's computation accepts a given input w.

The Main Ideas

- A **tableau** for non-deterministic TM N is a table listing its configurations on some branch of its computation tree.
- So determining if $w \in A$ is equivalent to whether or not there is a tableau using encoding an accepting computation of N on input w.

#	q_0	w_1	w_2	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	w_n	#
#	w_1'	q_1	w_2	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	w_n	#
#	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	#
#	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	#
#	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	#

Figure: Part of a tableau

The Main Ideas: Encoding the Tableau as a Formula

- Each entry of a tableau T of the tableau can be a state q_i of the TM Q, an element of the tape alphabet Γ or $\#$. Let $C = Q \cup \Gamma \cup \{\#\}$. We define a propositional variable $x_{i,j,s}$ for every cell in row i, column j, and element $s \in C$.
- We interpret $x_{i,j,s}$ as true iff T[i,j] = s.
- N accepts w iff
  1. Each cell is well-defined.
  2. The first row is an initial configuration with w as the input.
  3. Each row follows from the previous row using the transition function given by N.
  4. Some row has a cell that includes an accepting state q_{accept}.

  We can express each of these conditions using propositional logic in the variables $x_{i,j,s}$.

Condition 1: Well-defined Tableau

- A well-defined tableau means that every cell $T[i,j]$ in the tableau is filled with exactly one element (possibly the blank symbol).
- In propositional logic cell $T[i,j]$ being filled with exactly one element is equivalent to the proposition

$$\phi_{ij} = \left(\bigvee_{s \in C} x_{i,j,s} \right) \wedge \left(\bigwedge_{s,t \in C, s \neq t} (\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}}) \right)$$

  being true.
- We have a well-defined tableau iff

$$\phi_{cell} = \bigwedge_{i,j} \phi_{ij}$$

  is true.

Condition 2: The Initial Configuration

- The formula

$$\begin{aligned}\phi_{start} = x_{1,1,\#} \wedge x_{1,2,q_0} \wedge x_{1,3,w_1} \wedge x_{1,4,w_2} \wedge \ldots x_{1,n+2,w_n} \\ \wedge x_{1,n+3,\sqcup} \wedge \cdots \wedge x_{1,O(n^k)-1,\sqcup} \cdots x_{1,O(n^k),\#}\end{aligned} \tag{1}$$

is true iff $w = w_1 \ldots w_n$ is given as the input.

Condition 3: Valid Transitions

- A **window** in the tableau is a 2x3 piece with adjacent rows and columns.

a_1	a_2	a_3
a_4	a_5	a_6

- A window is **legal** if it does not violate transition function of N. Determining which windows are legal can be done by case analysis.
- Example: assuming that tape alphabet is $\{a, b, c\}$

a	b	c
a	c	c

is never a legal window for any Turing machine.

Condition 3: Valid Transitions

- Example: suppose the TM N has a transition function $\delta(q_1, b) = \{(q_2, c, L), (q_2, a, R)\}$ then

a	q_1	b
q_2	a	c

a	q_1	b
a	a	q_2

a	c	q_1
a	c	a

a	b	a
a	b	a

are all legal windows for N's computation but

a	q_1	b
q_1	b	b

cannot be.

Condition 3: Valid Transitions

- **Observation 1:** Each row in the tableau is a configuration following the previous row according to N if and only if each window in the tableau is legal.
  - Proof Sketch: For any row i, the configuration in row $i+1$ can differ from row i in at most 3 consecutive positions so checking all legal windows is the same is checking that the tableau is valid according to N.
- **Observation 2:** The number of legal windows is finite ($\leq |C|^6$.)

Condition 3: Valid Transitions

- Hence the condition that each row follows from the previous according to N can be expressed as the condition:

$$\phi_{move} = \bigwedge_{1 \leq i,j < O(n^k)} \phi_{window,i,j}$$

where $\phi_{window,i,j}$ expresses the condition that the window with cells $(a_1, \ldots, a_6)$ with top middle cell at (i, j) is legal.

$$\phi_{window,i,j} = \bigvee_{(a_1,\ldots,a_6) \text{ is legal}} (x_{i,j-1,a_1} \wedge x_{i,j,a_2} \wedge x_{i,j+1,a_3} \wedge$$
$$x_{i+1,j-1,a_4} \wedge x_{i+1,j,a_5} \wedge x_{i+1,j+1,a_6})$$

Condition 4: Accepting Configuration

- The tableau is accepting iff some cell in the tableau contains an accepting state.
- $$\phi_{accept} = \bigvee_{ij} x_{i,j,q_{accept}}$$

  iff the tableau is accepting.

Putting it Together

- Given a non-deterministic Turing machine N and some input w we have shown that there is a propositional formula ϕ defined by

$$\phi_{N,w} = \phi_{cell} \wedge \phi_{start} \wedge \phi_{move} \wedge \phi_{accept}$$

  that is satisfiable if and only N accepts w.
- The subformulas encode the 4 conditions needed there be an accepting tableau for the computation of N on input w.
- It remains to show that the reduction is computable in polynomial time.

Polynomial Time Reduction

- We assumed that the N runs in $O(n^k)$ time on inputs of length n so the tableau has $O(n^k)$ rows and $O(n^k)$ columns.
- The formula constructed by the reduction has $O(n^{2k})$ literals, since there is a constant size formula for each cell of the tableau.
- The formula for each cell can be generated efficiently from a description of NDTM N.
- All together this gives a reduction with runtime poly(n).
- This completes the reduction $A \leq_p SAT$. We can produce a formula $\phi_{N,w}$ in polynomial time that, which is satisfiable iff $w \in A$.

Reducing SAT to CNF-3SAT

- Converting an Boolean formula to one in CNF-form that preserves satisfiability can be done in polynomial time. (See Sipser for details)
- Now suppose we have a clause $\phi = l_1 \vee \cdots \vee l_n$ with $n > 3$. Introduce a new variable z and rewrite the clause as

$$(l_1 \vee l_2 \vee z) \wedge (\bar{z} \vee \cdots \vee l_n).$$

  Do this recursively until all clauses have 3 variables.
- Example with $n = 5$,

$$(l_1 \vee l_2 \vee z_1) \wedge (\bar{z}_1 \vee l_3 \vee z_2) \wedge (\bar{z}_2 \vee l_4 \vee l_5).$$

- **Claim:** This procedure can be done in poly-time and preserves satisfiability. So we have shown that SAT $\leq_p$ CNF-3SAT.

The Tree of Reductions

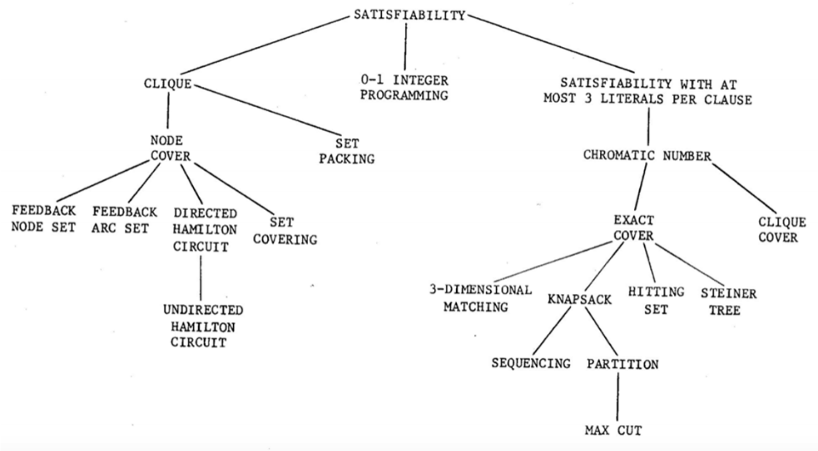
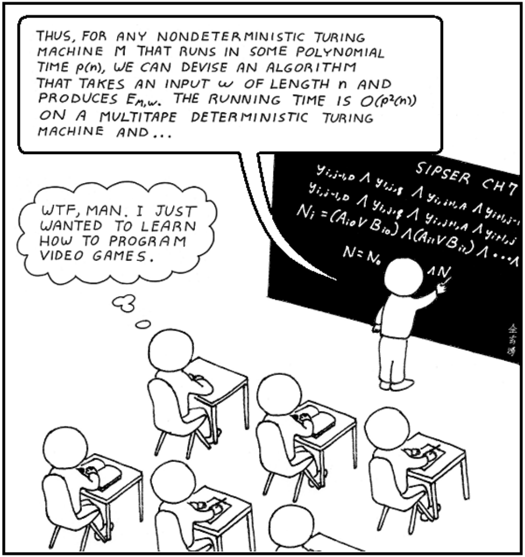


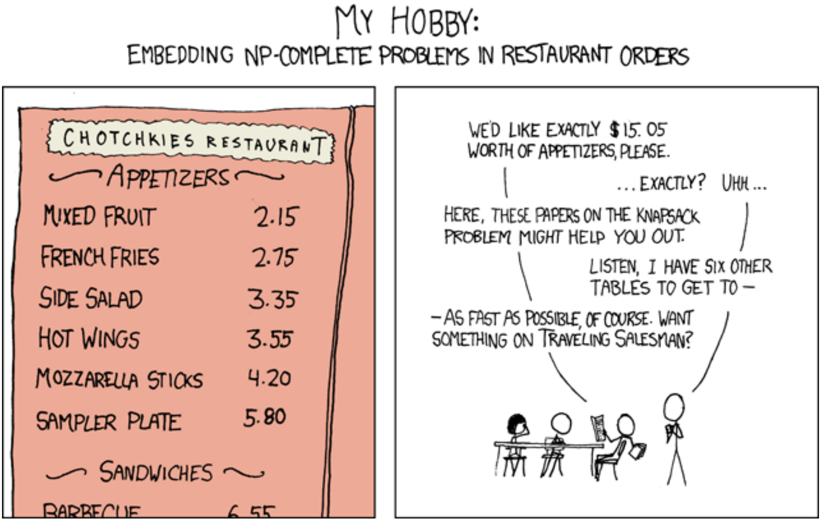
Figure: Karp (1972): Reducibility among Combinatorial Problems

So now you know how to prove the Cook-Levin Theorem! [1]



[1] Comic from abtrusegoose.com

Next week: more examples of NP-Complete problems [2]



[2] Comic from xkcd