Alex Hubers
CSC:341 Spring 2024

Lecture Notes #12A

#1 NP, etc

Recall that a language B is NP-complete if

- $B \in \mathsf{NP}$, and
- B is NP-hard. (for all $A \in \mathsf{NP}$, $A \leq_p B$).

We develop a notion of completeness, really, to serve the following two theorems:

- (Theorem 7.35). If B is NP-complete and $B \in P$, then $P = \mathsf{NP}$.
- (Theorem 7.36). If B is NP-complete and $C \in \mathsf{NP}$ is, then C is NP-complete.

(Both theorems follow from transitivity of $\leq_p$.)
7.35 means that a solution for *ANY* NP-complete problem is a solution for all problems in P; 7.36 means that we can show problems are NP-complete by reducing to them from other NP-complete problems. In other words: we only need to actually *prove* NP-hardness for one problem.
That problem is called SAT.

The claim that SAT is NP-complete is called the Cook-Levin Theorem. We have two obligations:

- Prove that SAT $\in \mathsf{NP}$. (You did this in Lab 11B.)
- Prove that, for all $A \in \mathsf{NP}$, $A \leq_p$ SAT.

The second part is what we will do for the rest of this lecture.

#2 The Cook-Levin Theorem

Here we go.

Theorem 1. SAT *is* NP-*hard.*

Let $A \in \mathsf{NP}$ be arbitrary. As $A \in \mathsf{NP}$, we know A can be decided in nondeterministic poly-time by some N. Suppose N runs in time n^k. We represent each input to N as a tableau. A tableau can be thought of as a table of configurations on a single branch of N. A tableau is an accepting tableau if any row on the tableau is an accepting configuration.

Picture goes here (See Figure 7.38).

Notes:

1. each tableau is an $n^k \times n^k$ table.
2. We represent configurations how Sipser does. So the string abq_3c means that abc is on the tape, the head is over c, and we're in state q_3.

Observe that each $w \in \Sigma^*$ has a tableau of computation with the first row filled in. In particular, the tableau's other rows have *many* ways to be "filled in"—one way for each nondeterministic branch of N on w. The crux of our proof relies on encoding each tableau as a boolean formula. Thus, our *reduction* takes a tableau C and builds a formula ϕ such that:

- if C is a tableau with an accepting configuration possible on N, then ϕ is satisfiable,
- if C is not accepting on any valid trace of configurations, then ϕ is not satisfiable.

In essence, we encode the idea that "C is an accepting tableau" as a formula ϕ. If ϕ is satisfiable, we know that C has *some* valid trace.

Let me be very clear. Here is the reduction:

1. f takes a string $w \in \Sigma^*$.
2. f builds a tableau with $\# q_0 w_1 ... w_n \square ... \#$ as the first row. All other cells are $\square$.
3. f encodes this tableau as a (big) logical formula.

And the argument for correctness goes like this:

1. if $w \in A$, then the tableau that f builds will have *some* trace resulting accept state. This trace will correspond to a satisfying assignment to ϕ.
2. if $w \notin A$, then the tableau will *not* have a trace resulting in an accept state. So ϕ will be unsatisfiable.

You should convince yourself now that this is a proper reduction: take $w \in \Sigma^*$, build tableau, turn tableau to formula. if $w \in A$ then its tableau should have a valid trace, which makes ϕ satisfiable. So $f(w) \in$ SAT... And so forth.

#3 The formula

Let $C = Q \cup \Gamma \cup \{\#\}$. Then our big formula will be the conjunction of four components:

$$\phi_{cell} \wedge \phi_{start} \wedge \phi_{move} \wedge \phi_{accept}$$

where each cell of the tableau is encoded as a variable:

$$x_{i,j,s}$$

for $1 \le i, j \le n^k$ and $s \in C$. If $x_{i,j,s}$ is true, then the cell at position i, j has symbol s written. For example, we expect $x_{1,1,\#}$ and $x_{1,1,q_0}$ to be true.

1. ϕ_{cell} is sort of a correctness assertion, which states that (i) each cell in the table has a symbol;
2. ϕ_{start} says that the starting configuration is $\# q_0 w_1 ... w_n \square ... \square \#$.
3. ϕ_{move} ensures that each row of the tableau legally follows from the preceding configuration according to δ.
4. ϕ_{accept} says that at least *one* row in the table is an accept state.

Among these three, ϕ_{move} is the most painful to think about.

#4 Part 1: ϕ_{start}

I'm going to go out of order.

We want to assert that the first row of the tableau is a start configuration—that is, the start state with the input on the tape.

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

#5 Part 2: ϕ_{accept}

We want to assert that at least *some* row is an accepting configuration.

$$\phi_{accept} = \bigvee_{1\le i,j\le n^k} x_{i,j,q_{accept}}$$

#6 Part 3: ϕ_{cell}

We want to assert that each cell has exactly one inhabitant.

$$\phi_{cell} = \bigwedge_{1\le i,j\le n^k} \left[\left(\bigvee_{s\in C} x_{i,j,s} \right) \wedge \left(\bigwedge_{s,t\in C, s\neq t} (\neg x_{i,j,s} \vee \neg x_{i,j,t}) \right) \right]$$

1. The left part says that *each cell has at least one symbol* and the right part says that *each cell has exactly one symbol.*

#7 Part 4: ϕ_{move}

Alrighty, here things get tricky.

The role of ϕ_{move} is to assert that each configuration follows from the one before it. Note that this is according to a non-deterministic δ.
We break this problem down into 2×3 windows. We then assert that every window is legal. A window is a 2×3 snapshot of 6 cells. Here's an example.
Suppose that

$$\delta(q_1, a) = \{(q_1, b, R)\}$$
$$\delta(q_1, b) = \{(q_2, c, L), (q_2, a, R)\}$$

Now, here are some valid windows:

a	q_1	b
q_2	a	c

a	a	q_1
a	a	b

b	b	b
c	b	b

We now assert that we can only consider windows to assert the correctness of transitions.

Claim 1. *If the top row of the tableau is the start configuration and every window in the tableau is legal, each row of the tableau is a configuration that legally follows the preceding one.*

Proof. I don't have time, but see the top two paragraphs of pp. 309. ☐

Finally, the encoding of ϕ_{move}.

We want to stipulate that all windows in the cell are legal.

$$\phi_{move} = \bigwedge_{1 \leq i < n^k, 1 < k < n^k} (\text{the } (i,j)\text{-window is legal})$$

where the (i,j)-window has the cell at row i and column j as the upper central position. This is the actual logic that fills in the "(The (i,j)-window is legal)" above:

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

Note that each window has *many* valid choices for $a_1 \ldots a_6$. We are asserting that at least one is chosen.

#8 Time complexity

It's easiest to consider the size of ϕ with respect to the input tableau. Firstly, the tableau has n^{2k} cells, and each cell has $|C|$ options for what it could be. so that is $\mathcal{O}(n^{2k})$ variables. Then

1. ϕ_{start} consists only have variables from the first row, so that's $\mathcal{O}(n^k)$.
2. ϕ_{cell}, ϕ_{move}, and ϕ_{accept} concern a fixed-size fragment of the formula for each cell of the whole tableau, so $\mathcal{O}(n^{2k})$ for each.

Thus we can say ϕ is polynomial in size w.r.t. n. No real time analysis is given in Sipser, and I'll be damned if I'm going to come up with it in my notes right here.