Lecture 15

- Recap: NP, NP-hardness, NP-completeness
- Cook-Levin Theorem: SAT is NP-complete
- Other NP-complete languages

RECAP

\[ \phi \text{ NP} = \bigcup_{k=1}^{\infty} \text{NTIME}(n^k) \]

- \( L \in \text{NP} \) if it has a non-deterministic polynomial-time decider.
- \( A \leq_p B \) if there exists a poly-time computable function \( f \) such that \( \forall x \in A \text{ iff } f(x) \in B \).
- \( L \) is NP-hard if \( \forall B \in \text{NP}, B \leq_p L \).
- \( L \) is NP-complete if \( \exists L \in \text{NP} \) and \( L \) is NP-hard.

We now give an example of a specific NP-complete language. The proof that it is complete is the Cook-Levin Theorem.

\[ \text{SAT} = \{ \phi(x_1, \ldots, x_n) | \phi \text{ is a Boolean formula that is satisfiable, i.e., there is a truth assignment } a_1, \ldots, a_n \text{ such that } \phi(a_1, a_2, \ldots, a_n) \text{ evaluates to True} \} \]
Example: $\phi_1 = (x_1 \lor \overline{x}_2) \land x_3$ is satisfiable because it is true when $x_1 = T$, $x_2 = T$, $x_3 = T$.

$\phi_2 = x_1 \lor \overline{x}_1$ is not satisfiable because no matter the assignment to $x_1$, $x_1 \lor \overline{x}_1$ is false.

**THEOREM.** [Cook–Levin] SAT is NP-Complete.

(Proof Later.)

**Corollary 1**

A corollary to this is that TAUTOLOGY is coNP-complete. Recall that for a complexity class $C$ is a collection of languages such that if $A \leq_p B$ and $B \in C$, then $A \in C$. We say that $L$ is $C$-hard if $B \in C$, $B \leq_p L$. $L$ is $C$-complete if $L$ is $C$-hard and $L \in C$. Examples of complexity classes: NP, coNP, P, PSPACE, etc.

Recall that $\text{coNP} = \{ L : L^c \in \text{NP} \}$. 2/9
Proof of Corollary. (Tautology is coNP-complete)

Let \( B \in \text{coNP} \). Then \( B^c \in \text{NP} \). Therefore,
\[
B^c \leq_p \text{SAT}
\]

Then \( \exists \) a poly-time \( f \) s.t., \( x \in B^c \) iff \( f(x) \in \text{SAT} \). Then \( B \leq_p \text{SAT} \).

Therefore, \( \text{SAT}^c \) is coNP-complete.

TAUTOLOGY is also coNP-complete, b/c

1. \( \text{TAUT.} \in \text{coNP} \)
2. \( \forall B \in \text{coNP}, B \leq_p \text{TAUT.} \)

To show 1), on input \( \phi \), non-deterministically pick an assignment \( x_1, x_2, \ldots, x_n \) to \( X_1, X_2, \ldots, X_m \). Accept if \( \phi(x_1, \ldots, x_n) \) is false.

To show 2), since we know \( \text{SAT}^c \) is coNP-complete, so it suffices to show \( \text{SAT}^c \leq_p \text{TAUT} \). Use the reduction
\[
f(\phi) = \overline{\phi}.
\]

**Corollary 2**

3SAT is NP-complete.

Proof. You will show on the homework that \( \text{SAT} \leq_p \text{3SAT} \). We also can show \( \text{3SAT} \leq_p \text{SAT} \). Since \( \text{SAT} \) is NP-complete, so is \( \text{3SAT} \).
COOK-LEVIN THM PROOF

1. SAT \in NP.

On input \( \phi \), non-deterministically pick an assignment \( a_1, a_2, \ldots, a_n \). If \( \phi(a_1, \ldots, a_n) = T \), accept, else reject.

2. WTS: \( \forall B \in \text{NP}, B \in \text{P} \iff \text{SAT} \).

Suppose \( B \in \text{NP} \) is given. Let \( N \) be the poly-time NTM such that \( L(N) = B \). Let \( p(n) \) be its run-time.

\text{WANT:} poly-time computable \( f \) s.t. \( \forall x, x \in B \iff f(x) \in \text{SAT} \).

\text{ROADMAP:}

\( x \in B \iff N \text{ accepts } x \)
\( \iff \exists \text{ an accepting computational history of } N \text{ on input } x \) (of length \( \leq p(n) \))
\( \iff \exists \text{ an accepting tableau of } N \text{ on input } x \)
\( \iff \exists \text{ an assignment } a_1, a_2, \ldots, a_n \) such that \( \phi(a_1, \ldots, a_n) = T \) (\( \phi \) constructed from \( N \) and \( x \)).
\( \iff \phi \in \text{SAT} \).

We need \( \Phi \) to find \( \phi \) given \( x \), i.e. to represent the existence of an accepting tableau as existence of a satisfying assignment to \( \phi \).
A tableau is a table where each row represents a state of the tape. $T$ is a tableau for $N$ on input $x$ if

1. $T$ has $p(1|x_1) + 1$ rows and $p(1|x_1) + 3$ columns.

Row $i$ represents a configuration of $N$ at time $i$.

If in state $q_i$, and head pointing at $a$, where the tape reads $b_1, b_2, b_3, \ldots$, be a $c_1, c_2, \ldots, c_m$,

This row reads $\# b_1, b_2, b_3, \ldots, b \# q_i, a, c_1, c_2, \ldots, c_m$. Put with blank.
2. Row 0 is

\[ \# q_0 \ x_1 \ x_2 \ x_3 \ldots \ x_n \ \# \]

3. \( \forall i, \text{with } 1 \leq i \leq p(1x1) \), row i follows from \( i-1 \) using a valid transition of \( N_0 \).

4. A tableau is accepting if \( q_\text{accept} \) appears somewhere in the tableau.

Example Tableau

Row 0: \[ \# q_0 \ a \ x_2 \ldots \ x_n \ \# \]

Transition: \[ q_0 \rightarrow q_{13} \]

Row 1: \[ \# b \ q_{13} \ x_2 \ldots \ x_n \ \# \]

To decide whether row it follows from \( i \) you only need to check only all the \( 2 \times 3 \) windows. In the above example, you check

\[ \begin{array}{c|c|c|c}
q_0 & q_1 & q_2 \\
q_0 & q_{13} & q_{21} \\
\end{array} \]

etc.

\[ \begin{array}{c|c|c|c|c|c|c}
\text{LEGAL} & \text{ILLEGAL} & \text{ILLEGAL} & \text{ILLEGAL} & \text{ILLEGAL} \\
q \rightarrow r & \text{b} \ q \ a & \text{b} \ q \ a & \text{b} \ q \ a & \text{b} \ q \ a \\
r \rightarrow b b & \text{b} \ a \ b & \text{b} \ a \ b & \text{b} \ a \ b & \text{b} \ a \ b \\
\end{array} \]

If there is no transition \( q \rightarrow a \ u \) in \( N \).
So go back to the definition of $T$. Equivalent to (3) is condition (3) :

$$\forall 0 \leq i \leq p(1x1) - 1, \forall 1 \leq j \leq p(1x1)+1,$$

the $2 \times 3$ window that starts at position $(i,j)$ is legal.

Note: Since $\#s$ states in $N$ and $\#s$ symbols in $\Sigma$ are finite, there are only a finite, constant number of legal tableaus.

Next step: Translate Tableau to a Boolean Formula.

For every $i, j, s$, where $i$ is a row, $j$ is a column, and $s$ is a symbol or state or special symbol $\#$, make a variable $Y_{ij}s$ true if $(ij)$ contains $s$, false otherwise.

$\Delta = \text{set of symbols, states, and } \#$

$\phi$ has variables $\forall Y_{ij}s \mid 0 \leq i \leq p(1x1), 1 \leq j \leq p(1x1)+3, s \in \Delta^3$. 

7/9
Construct $\phi$ in parts.

**PART 1**. $\phi_{\text{cell}}$ ensures that each cell has exactly one symbol inside it.

$$\phi_{\text{cell}} = \bigwedge_{i=0}^{p(1|x)} \bigwedge_{j=1}^{p(1|x)+3} \phi_{\text{cell}_{i,j}}$$

Where $\phi_{\text{cell}_{ij}} = \left( \bigvee_{s \in \Delta} Y_{i,j,s} \right) \wedge \left( \bigwedge_{s,s' \in \Delta, s \neq s'} Y_{i,j,s} \vee Y_{i,j,s'} \right)$. At least one symbol inside cell $i,j$ and no more than one symbol inside cell $i,j$.

**PART 2**. $\phi_{\text{start}}$ is only satisfied by an assignment that has a correct start configuration.

$$\phi_{\text{start}} = Y_{0,1,1} \wedge Y_{0,2,q_0} \wedge Y_{0,3,1} \wedge Y_{0,4,1} \wedge \ldots \wedge Y_{0,1\times 1+2, x_n} \wedge Y_{0,1\times 1+3, l_1} \wedge \ldots \wedge Y_{0,p(1|x)+2,l_1} \wedge Y_{0,p(1|x)+3,\#}$$

(Refer to condition 2.)
PART 3: $\phi_{\text{move}}$ is only satisfied by tableaus where every 2 x 3 window is legal. (Refer to 3).

\[
\phi_{\text{move}} = \bigwedge_{i=0}^{p(1x)+1} \bigwedge_{j=1}^{p(1x)} \bigvee_{i,j,i',j'} Y_{i,j,i',j'} \wedge Y_{i,j+1,i,j+1} \wedge Y_{i+j+1,i,j+1} \\
\wedge Y_{i,j+1,i',j+1} \wedge Y_{i+j+1,i',j+1} \wedge Y_{i+j+1,i+j+1} \wedge Y_{i,j,i+j+1} \wedge Y_{i,j+1,i+j+1} \wedge Y_{i+j+1,i,j} \wedge Y_{i+j+1,i+1,j} \wedge Y_{i+j+1,i+1,j} \wedge Y_{i+j+1,i+1,j+1} \\
\text{Legal}
\]

\text{Finish set!} \\
\text{Constant size}

PART 4: $\phi_{\text{accept}}$ is only satisfied by accepting tableaus, i.e. $\phi_{\text{accept}}$ appears somewhere.

\[
\phi_{\text{accept}} = \bigvee_{i=0}^{p(1x)} \bigvee_{j=1}^{p(1x)+3} Y_{i,j} \phi_{\text{accept}}
\]

COMBINE.

Now we set $\phi(\exists Y_{i,j} \exists Y_{i,j}) = \phi_{\text{cell}} \land \phi_{\text{start}} \land \phi_{\text{move}} \land \phi_{\text{accept}}$. $\phi$, on input $x$, constructs $\phi_{\text{cell}}, \phi_{\text{start}}, \phi_{\text{move}},$ and $\phi_{\text{accept}}$, and returns $\phi$. All of these constructions take polynomial time.

Cook-Levin reduction $\phi$.

On input $x$, 
1. Compute the list of variables $Y_{i,j}$ (polynomial time).
2. Compute the list of valid tableaus (constant time).
3. Compute $\phi$ (each disjunct/conjunct only has a polynomial size).

$\square$