CS154, Lecture 15: Cook-Levin Theorem

\begin{itemize}
  \item \textbf{P}
  \item \textbf{NP}
  \item \textbf{SAT, 3SAT}
\end{itemize}
Definition: A language \( B \) is \( \text{NP} \)-complete if:

1. \( B \in \text{NP} \)

2. Every \( A \) in \( \text{NP} \) is poly-time reducible to \( B \)
   That is, \( A \leq_p B \)
   When this is true, we say “\( B \) is \( \text{NP} \)-hard”

On homework, you showed
A language \( L \) is recognizable iff \( L \leq_m A_{\text{TM}} \)

\( A_{\text{TM}} \) is “complete for recognizable languages”:
\( A_{\text{TM}} \) is recognizable, and for all recognizable \( L \), \( L \leq_m A_{\text{TM}} \)
Suppose \( L \) is NP-Complete...

If \( L \in P \), then \( P = NP \)

If \( L \notin P \), then \( P \neq NP \)
Suppose $L$ is NP-Complete...

Then assuming the conjecture $P \neq NP$, $L$ is not decidable in $n^k$ time, for every $k$. 
The Cook-Levin Theorem: SAT and 3SAT are NP-complete

1. **3SAT ∈ NP**
   A satisfying assignment is a “proof” that a 3cnf formula is satisfiable

2. **3SAT is NP-hard**
   Every language in NP can be polynomial-time reduced to 3SAT (complex logical formula)

**Corollary**: 3SAT ∈ P if and only if P = NP
Theorem (Cook-Levin): 3SAT is NP-complete

Proof Idea:

(1) 3SAT ∈ NP (done)

(2) Every language A in NP is polynomial time reducible to 3SAT (this is the challenge)

We give a poly-time reduction from A to SAT.

The reduction converts a string w into a 3cnf formula φ such that w ∈ A iff φ ∈ 3SAT

For any A ∈ NP, let N be a nondeterministic TM deciding A in n^k time.

φ will simulate N on w.
Deterministic Computation

Nondeterministic Computation

accept or reject

$\exp(n^k)$
Let $L(N) \in \text{NTIME}(n^k)$. A tableau for $N$ on $w$ is an $n^k \times n^k$ table whose rows are the configurations of some possible computation history of $N$ on $w$. Each “cell” contains a $\sigma \in Q \cup \Gamma \cup \{\#\}$. 

<table>
<thead>
<tr>
<th></th>
<th>$q_0$</th>
<th>$w_1$</th>
<th>$w_2$</th>
<th>...</th>
<th>$w_n$</th>
<th>$\square$</th>
<th>...</th>
<th>$\square$</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>#</td>
</tr>
<tr>
<td>#</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>#</td>
</tr>
<tr>
<td>#</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>#</td>
</tr>
<tr>
<td>#</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>#</td>
</tr>
<tr>
<td>#</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>#</td>
</tr>
</tbody>
</table>
A tableau is accepting if the last row of the tableau is an accepting configuration.

$N$ accepts $w$ if and only if there is an accepting tableau for $N$ on $w$.

Given $w$, we’ll construct a 3cnf formula $\phi$ with $O(|w|^{2k})$ clauses, describing logical constraints that any accepting tableau for $N$ on $w$ must satisfy.

The 3cnf formula $\phi$ will be satisfiable if and only if there is an accepting tableau for $N$ on $w$. 
Variables of formula $\phi$ will encode a tableau

Let $C = Q \cup \Gamma \cup \{ \# \}$

Each of the $(n^k)^2$ entries of a tableau is a cell containing value in $C$

$\text{cell}[i,j] = \text{value of the cell at row } i \text{ and column } j$

$= \text{the } j\text{th symbol in the } i\text{th configuration}$

For every $i$ and $j$ ($1 \leq i, j \leq n^k$) and for every $s \in C$
we have a Boolean variable $x_{i,j,s}$ in $\phi$

Total number of variables $= |C|n^{2k}$, which is $O(n^{2k})$

These $x_{i,j,s}$ are the variables of $\phi$ and represent the contents of the cells

We will have: for all $i,j,s$, $x_{i,j,s} = 1 \iff \text{cell}[i,j] = s$
Idea: Make $\phi$ so that every *satisfying assignment* to the variables $x_{i,j,s}$ corresponds to an *accepting tableau* for $N$ on $w$ (an assignment to all $cell[i,j]$’s of the tableau)

The formula $\phi$ will be the AND of four CNF formulas:

$$\phi = \phi_{cell} \land \phi_{start} \land \phi_{accept} \land \phi_{move}$$

$\phi_{cell}$: for all $i, j$, there is a unique $s \in C$ with $x_{i,j,s} = 1$

$\phi_{start}$: the first row of the table equals the *start* configuration of $N$ on $w$

$\phi_{accept}$: the last row of the table has an accept state

$\phi_{move}$: every row is a configuration that yields the configuration on the next row
\( \phi_{\text{cell}} \): for all \( i, j \), there is a unique \( s \in C \) with \( x_{i,j,s} = 1 \)

\[
\phi_{\text{cell}} = \bigwedge_{1 \leq i, j \leq n^k} \left[ \left( \bigvee_{s \in C} x_{i,j,s} \right) \land \left( \bigwedge_{s,t \in C, s \neq t} (\neg x_{i,j,s} \lor \neg x_{i,j,t}) \right) \right]
\]

for all \( i, j \) \quad at least one \( x_{i,j,s} \) is set to 1

at most one \( x_{i,j,s} \) is set to 1
\( \phi_{\text{start}} \): the first row of the table equals the \textit{start} configuration of \( N \) on \( w \)

\[
\phi_{\text{start}} = X_{1,1,\#} \land X_{1,2,q_0} \land X_{1,3,w_1} \land X_{1,4,w_2} \land \ldots \land X_{1,n+2,w_n} \land X_{1,n+3,\square} \land \ldots \land X_{1,n^k-1,\square} \land X_{1,n^k,\#}
\]

<table>
<thead>
<tr>
<th></th>
<th>q₀</th>
<th>w₁</th>
<th>w₂</th>
<th>...</th>
<th>wₙ</th>
<th>□</th>
<th>...</th>
<th>□</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
\[ \phi_{\text{accept}} : \text{the last row of the table has an accept state} \]

\[ \phi_{\text{accept}} = \bigvee_{1 \leq j \leq n^k} x_{n^k,j}, q_{\text{accept}} \]
$\phi_{\text{move}}$: every row is a configuration that yields the configuration on the next row

**Key Question:** If one row yields the next row, how many cells can be different between the two rows?

**Answer:** at most three cells
$\phi_{\text{move}}$: every row is a configuration that yields the configuration on the next row

Idea: check that every $2 \times 3$ “window” of cells is legal (consistent with the transition function of $N$)
Example: Let \( N = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}}) \)

Suppose \( a, b, c \in \Gamma, \ q_1, q_2 \in Q \) and
\[
\delta(q_1, a) = \{(q_1, b, R)\}
\]
\[
\delta(q_1, b) = \{(q_2, c, L), (q_2, a, R)\}
\]

Legal = Consistent with \( N \)’s transition function

Illegal = Inconsistent with \( N \)’s transition function
Key Lemma:
IF   Every window of the tableau is legal, and  
    The top row is the start configuration
THEN  Each row of the tableau is a configuration that yields the 
    next row on the tableau

Proof Sketch: (Strong) induction on the rows.  
The top row is a configuration. If it does not yield the next row, 
then there is a \(2 \times 3\) window that is “illegal”

Suppose the first \(1, \ldots, k\) rows are configurations which yield the next, 
and assume every window is legal.

If row \(k+1\) did not yield row \(k+2\), then there must be a \(2 \times 3\) window 
along those two rows which is “illegal” – contradiction.
The \((i, j)\) window of a tableau is the tuple \((a_1, \ldots, a_6) \in \mathbb{C}^6\) such that:

\[
\begin{array}{ccc}
\text{col. } j & \text{col. } j+1 & \text{col. } j+2 \\
\hline
\text{row } i & a_1 & a_2 & a_3 \\
\text{row } i+1 & a_4 & a_5 & a_6 \\
\end{array}
\]
\( \phi_{\text{move}} : \) every row is a configuration that legally follows from the previous configuration

\[
\phi_{\text{move}} = \bigwedge \quad ( \text{the (i, j) window is legal})
\]

\[
1 \leq i \leq n^k - 1
\]
\[
1 \leq j \leq n^k - 2
\]

\[(\text{the (i, j) window is legal}) =
\]

\[
\bigvee \quad (x_{i,j,a_1} \land x_{i,j+1,a_2} \land x_{i,j+2,a_3} \land x_{i+1,j,a_4} \land x_{i+1,j+1,a_5} \land x_{i+1,j+2,a_6})
\]

\((a_1, \ldots, a_6)\) is a legal window

\[
\equiv \bigwedge \quad (x_{i,j,a_1} \lor x_{i,j+1,a_2} \lor x_{i,j+2,a_3} \lor x_{i+1,j,a_4} \lor x_{i+1,j+1,a_5} \lor x_{i+1,j+2,a_6})
\]

\((a_1, \ldots, a_6)\) is NOT a legal window
How do we get 3SAT?

We had some long clauses in there... how do we convert the whole thing into a 3-cnf formula?

Everything was an AND of ORs (a CNF). We just need to make those ORs small

\((a_1 \lor a_2 \lor ... \lor a_t)\) is equivalent to

\((a_1 \lor a_2 \lor z_1) \land (\neg z_1 \lor a_3 \lor z_2) \land (\neg z_2 \lor a_4 \lor z_3) \dots \land (\neg z_{t-3} \lor a_{t-1} \lor a_t)\)

\((\text{SAT is polynomial time reducible to 3SAT})\)
What’s the total length of $\phi$?

$\phi = \phi_{\text{cell}} \land \phi_{\text{start}} \land \phi_{\text{accept}} \land \phi_{\text{move}}$

- $O(n^{2k})$ clauses
- $O(n^k)$ clauses
- $O(n^k)$ clauses
- $O(n^{2k})$ clauses

Table:

<table>
<thead>
<tr>
<th>#</th>
<th>q_0</th>
<th>w_1</th>
<th>w_2</th>
<th>...</th>
<th>w_n</th>
<th>#</th>
<th>...</th>
<th>...</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>#</td>
<td></td>
<td></td>
<td>#</td>
</tr>
<tr>
<td>#</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>#</td>
<td></td>
<td></td>
<td>#</td>
</tr>
<tr>
<td>#</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>#</td>
<td></td>
<td></td>
<td>#</td>
</tr>
</tbody>
</table>

$n^k$ x $n^k$ matrix
Summary. We wanted to prove:
Every $A$ in NP has a polynomial time reduction to 3SAT

For every $A$ in NP, we know $A$ is decided by some nondeterministic $n^k$-time Turing machine $N$

We gave a generic method to reduce $(N, w)$ to a 3CNF formula $\phi$ of $O(|w|^{2k})$ clauses such that satisfying assignments to the variables of $\phi$ directly correspond to accepting computation histories of $N$ on $w$

The formula $\phi$ is the AND of four 3CNF formulas:
$\phi = \phi_{\text{cell}} \land \phi_{\text{start}} \land \phi_{\text{accept}} \land \phi_{\text{move}}$
Reading Assignment

Read Luca Trevisan’s notes for an alternative proof of the Cook-Levin Theorem

Sketch:
1. Define CIRCUIT-SAT: Given a logical circuit $C(y)$, is there an input $a$ such that $C(a)=1$?
2. Show that CIRCUIT-SAT is NP-hard: The $n^k \times n^k$ tableau for $N$ on $w$ can be simulated using a logical circuit of $O(n^{2k})$ gates
3. Reduce CIRCUIT-SAT to 3SAT in polytime
4. Conclude 3SAT is also NP-hard
Theorem (Cook-Levin):
SAT and 3SAT are NP-complete

Corollary: SAT ∈ P if and only if P = NP