COOK'S THEOREM

1 Cook's Theorem

Cook's theorem shows that the satisfiability problem is NP-complete. Without loss of generality, we assume that languages in NP are over the alphabet \{0, 1\}*. Lemma 1, useful for the proof, states that we can restrict the form of a computation of a NTM that accepts languages in NP.

**Lemma 1** If \( L \in \text{NP} \), then \( L \) is accepted by a 1-tape NTM \( N \) with alphabet \{0, 1\} such that for some polynomial \( p(n) \), the following properties hold:

- \( N \)'s computation is composed of two phases, the guessing phase and the checking phase.
- In the guessing phase, \( N \) nondeterministically writes a string \( \cup j \) directly after the input string, and in the checking phase, \( N \) behaves deterministically.
- \( N \) uses at most \( p(n) \) tape cells, never moves its head to the left of \( w \), and takes exactly \( p(n) \) steps in the checking phase.

A Boolean formula \( f \) over variable set \( V \) is in conjunctive normal form (CNF) if

\[
 f = \bigwedge_{i=1}^{m} \bigvee_{j=1}^{k_i} l_{ij}
\]

for some values of \( m \) and \( k_i, 1 \leq i \leq m \), where literal \( l_{ij} \) is either \( x \) or \( \overline{x} \) for some \( x \in V \). For each \( k_i \), the term \( \bigvee_{j=1}^{k_i} l_{ij} \) is called a clause of the formula. \( f \) is satisfiable if there exists a truth assignment to the variables in \( V \) that sets \( f \) to true. CNFSAT is the set of satisfiable Boolean formulas in CNF.

**Theorem 1 (Cook's Theorem)** CNFSAT is NP-complete.

Proof sketch: It is not hard to show that CNFSAT \( \in \text{NP} \). To prove that CNFSAT is NP-complete, we show that for any language \( L \in \text{NP} \), \( L \leq_{p} \text{CNFSAT} \).

Let \( L \in \text{NP} \) and let \( N \) be a NTM accepting \( L \) that satisfies the properties of Lemma 1. Let the transition function of \( N \) be \( \delta \). Let the states of \( N \) be \( \{0_0, 0_1, \ldots, q_r\} \). Let \( q_0, q_1, q_2 \) denote 0, 1, \( \cup \), respectively. Assume that the tape cells are numbered consecutively from the left end of the input, starting at 0. On input \( w \) of length \( n \), we show how to construct a formula in CNF form \( f_w \), which is satisfiable if and only if \( w \) is accepted by \( N \). The variables of \( f_w \) are as follows:

<table>
<thead>
<tr>
<th>Variables</th>
<th>Range</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Q[i,k] )</td>
<td>( 0 \leq i \leq p(n) )</td>
<td>At step ( i ) of the checking phase, the state of ( N ) is ( q_2 ).</td>
</tr>
<tr>
<td>( H[i,j] )</td>
<td>( 0 \leq i \leq p(n) )</td>
<td>At step ( i ) of the checking phase, the head of ( N ) is on tape square ( j ).</td>
</tr>
<tr>
<td>( S[i,j,l] )</td>
<td>( 0 \leq i \leq p(n) )</td>
<td>At step ( i ) of the checking phase, the symbol in square ( j ) is ( s_l ).</td>
</tr>
</tbody>
</table>
A computation of $N$ naturally corresponds to an assignment of truth values to the variables. Other assignments to the variables may be meaningless. For example, an assignment with $Q[i, k] = k' = k$, would imply that $N$ is in two different states at step $i$, which is impossible.

Our goal is to construct $f_w$ so that it is satisfied only by assignments to the variables that correspond to accepting computations of $N$ on $w$. The clauses of $f_w$ are constructed to ensure that the following conditions are satisfied:

1. At each step $i$ of the checking phase, $N$ is in exactly 1 state.
2. At each step $i$, the head is on exactly one tape square.
3. At each step $i$, there is exactly 1 symbol in each tape square.
4. At step 0 of the checking phase, the state is the initial state of $N$ in its checking phase, and the tape contents are $w$. 
5. At step $p(n)$ of the checking phase, $N$ is in an accepting state.
6. The configuration of $N$ at the $(i+1)$th step follows from that at the $i$th step, by applying the transition function of $N$.

Consider condition 1. For each $i$, we have the following clause:

$$Q[i, 0] \lor Q[i, 1] \lor \ldots \lor Q[i, r].$$

This clause ensures that the machine is in at least 1 state at step $i$. We also need clauses to ensure that $N$ is not both in state $q_j$ and $q_j'$:

$$Q[i, j] \lor Q[i, j'] \text{ for each } j \neq j', 0 \leq j, j' \leq r.$$ Conditions 2 and 3 are handled similarly. Conditions 4 and 5 are quite easy. Finally, consider condition 6. For each $(i, j, k, l)$, we add clauses that ensure the following: If at step $i$, the tape head of $N$ is pointing to the $j$th tape cell, $N$ is in state $q_k$, $s_l$ is the symbol under the tape head, and $(q_k, s_l, q_{k'}, s_{l'}) \in \delta$, where $X \in \{L, R\}$ then at step $i+1$, the tape head is pointing to the $(j+y)$th tape cell where $y = 1$ if $X = R$ and $y = -1$ if $X = L$, $N$ is in state $q_{k'}$ and the symbol in cell $j$ is $s_{l'}$. The following clauses ensure this:

$$Q[i, k] \lor H[i, j] \lor S[i, j, l] \lor Q[i+1, k']$$

$$Q[i, k] \lor H[i, j] \lor S[i, j, l] \lor H[i+1, j + y]$$

$$Q[i, k] \lor H[i, j] \lor S[i, j, l] \lor S[i+1, j, l']$$

All of the clauses for condition 1 to 6 can be computed in polynomial time (how many clauses are there?). Moreover, $w$ is accepted by $N$ if and only if $f_w$ is satisfiable.