1 Cook-Levin Theorem: SAT is NP-complete

(Answner from Sipser textbook)

The Cook-Levin Theorem: **SAT is NP-complete**

To proof that we have to:

1. SAT $\in$ NP
2. any laguage in NP is $\leq_p$ to SAT

(1) A nondeterministic polynomial time machine can guess an assignment to a given formula ϕ and accept if the assignment satisfies ϕ.

(2) Let A a language in NP and N be the NTM (Nondeterministric Turing Machine) that decides A. The machine N decides A in n^k time for some constant k.

We will construct a tableau for N on w of size $n^k \times n^k$. The tableau is constructed following these propreties:

- Each row represent the configurations of a branch of computation of N on w
- Each row starts and ends with the symbol #
- The first row is the starting configuration and each row follows the previous according to N's transition function
- A tableau is **accepting** if any row of the tableau is an accepting configuraton

The reduction build tableau based on the computation tree, and every accepting tableau corresponds to an accepting computation branch of N on w.

1.1 Reduction

On input w the reduction produces a formula ϕ.

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

This formula is composed of a set of literals $x_{i,j,s}$ where $1 \leq i,j \leq n^k$ and $s \in C$ with $C = Q \cup \cup\Gamma \cup \{\#\}$ (i.e. a symbol that can be in the tableau).

ϕ_{cell} Ensure that each cell contains one and only one character:

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

The idea is the following: for each cell, at least one symbol must be true (first clause) and two symbols cannot be true together (second clause) i.e. one symbol can be true for a cell.

ϕ_{start} Ensure that the first row is the start configuration of N on w:

$$\phi_{start} = x_{1,1,\#} \wedge x_{1,2,q_0} \wedge x_{1,3,w_1} \wedge x_{1,4,w_2} \wedge ... \wedge x_{1,n+2,w_n} \wedge x_{1,n+3,\sqcup} \wedge ... \wedge \wedge x_{1,n^k-1,\sqcup} \wedge x_{1,n^k,\#}$$

ϕ_{accept} Guarantees that an accepting configurations occurs in the tableau:

$$\bigvee_{1 \leq i,j \leq n^k} x_{i,j,q_{accept}}$$

ϕ_{move} Guarantees that each row of the tableau corresponds to a configuration that legally follows the configuration of the preceding row according to N's rules. It does so by ensuring that each 2×3 window of the tableau is **legal**.

A windows is legal when it respects the following transitions: $\delta(q_1, a) = \{(q_1, b, R)\}$ and $\delta(q_1, b) = \{(q_2, c, L), (q_2, a, R)\}$. With a_1, ..., a_6 a legal window, we have the following formula:

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

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

Next, we analyze the complexity of the reduction by looking at the size of ϕ. The tableau contains n^{2k} cells. Each cells has l variables associated with l being the size of C. Because l depends only on the TM N and not on the input size n, the total number of variables in $O(n^{2k})$. ϕ_{cell}, ϕ_{accept} and ϕ_{move} contains a fixed size formula for each cell *i.e.* their size is $O(n^{2k})$. ϕ_{start} has a fragment for each cell in the top row, so its size is $O(n^k)$. Total size of ϕ is $O(n^{2k})$. This is sufficient and shows that the size of ϕ is polynomial in n, thus the reduction can be done in polynomial time.