The first $\mathcal{NP}$-complete problem

The proof of the first $\mathcal{NP}$-complete problem was done in the early 1970's by Stephen Cook and Leonid Levin. They proved that SAT is NP-Complete.

SAT = $\{\langle\phi\rangle \mid \phi$ is a satisfiable Boolean formula$\}$.

A **boolean formula** is a formula of boolean variables using AND, NOT, and OR operators. A formula is satisfiable if there is an assignation of true/false values (a "TVA": a "truth value assignment" to the boolean variables such that the formula evaluates to true.

- We can show that SAT $\in \mathcal{NP}$.
- For any problem $x \in \mathcal{NP}$, there must be an NTM N that decides x in time n^k.
- Therefore, for any input $w = w_1 w_2 ... w_n$, there is an accepting computation history on w of length $\leq n^k$.
- Additionally, the length of the worktape will never exceed n^k.
- Imagine an accepting computation history in an n^k by n^k grid, referred to as a **tableau**.

#	q_0	w_1	w_2	...	w_n	_	...	_	#
#	a	q_1	w_2	...	w_n	_	...	_	#
#	...								#

- The first row is the starting configuration, the next row is the second configuration, etc. By the n^kth configuration, we are guaranteed to be at an accepting configuration.
- Our algorithm for x is outlined as follows:
  1. Create a SAT formula ϕ (in polynomial time) that is satisifiable exactly when it describes a valid tableau for the problem.
  2. If we could solve SAT in polynomial time, then we'd be able to solve ϕ in polynomial time.
  3. The solution to ϕ tells us how to construct the tableau in polynomial time.
  4. With the tableau, we know which nondeterminisitic choices the machine makes. We simulate ϕ using the tableau as a roadmap to identify the solution to x in n^k time.

How to construct ϕ:

- Let $C = Q \cup \Gamma \cup \{\#\}$. These are the characters that can appear in a cell in the tableau.
- We will have $|C| \cdot (n^k)^2$ variables. $x_{i,j,s}$ is true exactly when the ith row and the jth column in the tableau contains $s \in C$.
- We'll break ϕ up into pieces: $\phi = \phi_{\text{cell}} \wedge \phi_{\text{start}} \wedge \phi_{\text{accept}} \wedge \phi_{\text{move}}$.
- Each cell in the tableau contains exactly one character. We describe this logic in ϕ_{cell}.

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

- The length of ϕ_{cell} (and the time it takes to construct it) is $\mathcal{O}((|C| \cdot n^k)^2)$.
- The first row in the tableau must be the starting configuration. We describe this logic in ϕ_{start}.

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

- The length of ϕ_{start} (and the time it takes to construct it) is $\mathcal{O}(n^k)$.
- Since this must be an accepting tableau, q_{accept} must show up somewhere. We describe this logic in ϕ_{accept}.

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

This leaves us to specify what constitutes a valid move. Does one configuration follow validly from the previous? We'll describe this logic in ϕ_{move}:

- We will look at every 2×3 window (2 rows, 3 columns) of the tableau, and verify that it is valid.
- If $w_{i,j}$ contains the logic to indicate whether the window whose top-left coordinate is (i, j) is valid, then our logic will look like:

$$\phi_{\text{move}} = \bigwedge_{1 \le i < n^k, 1 \le j < n^k - 1} w_{i,j}$$

- If the contents of the window are $a_1, a_2, ..., a_6$, then the window logic will look like this:

$$w_{i,j} = \bigvee_{\text{valid } a_1,...,a_6} x_{i,j,a_1} \wedge x_{i,j+1,a_2} \wedge x_{i,j+2,a_3} \wedge x_{i+1,j,a_4} \wedge x_{i+1,j+1,a_5} \wedge x_{i+1,j+2,a_6}$$

- Assuming $(q_2, c, L) \in \delta(q_1, b)$ and $(q_4, a, R) \in \delta(q_3, b)$, here are some examples of valid windows:

a	q_1	b
q_2	a	c

a	q_3	b
a	a	q_4

a	a	q_1
a	a	b

#	b	a
#	b	a

a	b	a
a	b	q_2

b	b	b
c	b	b

- Assuming $(q_2, a, L) \notin \delta(q_1, b)$, here are some examples of invalid windows:

a	b	a
a	a	a

a	q_1	b
q_1	a	a

q_1	a	a
a	a	q_2

a	a	a
a	q_1	a

a	q_1	a
a	a	a

b	q_1	b
q_2	b	q_2

- Why are we specifically looking at 2×3 windows?

- There are $|C|^6$ possible windows, each window can be described in length 6, so $w_{i,j}$ takes $\mathcal{O}(|C|^6)$ time to compute and to write.

- There are $(n^k)^2$ different $w_{i,j}$ formula, so the length of ϕ_{move} (and the time it takes to construct it) is $\mathcal{O}(|C|^6 \cdot (n^k)^2)$.

Therefore, this reduction takes polynomial-time, completing the proof.