7.37 Cook-Levin theorem

SAT is NP-complete

PROOF IDEA

Showing that SAT is in NP is easy, and we do so shortly. The hard part of the proof is showing that any language in NP is polynomial time reducible to SAT.

To do so, we construct a polynomial time reduction for each language A in NP to SAT. The reduction for A takes a string w and produces a Boolean formula ϕ that simulates the NP machine for A on input w. If the machine accepts, ϕ has a satisfying assignment that corresponds to the accepting computation. If the machine doesn't accept, no assignment satisfies ϕ. Therefore, w is in A if and only if ϕ is satisfiable.

Constructing the reduction to work in this way is a conceptually simple task, though we must cope with many details. A Boolean formula may contain the Boolean operations AND, OR, and NOT, and these operations form the basis for the circuitry used in electronic computers. Hence the fact that we can design a Boolean formula to simulate a Turing machine isn't surprising. The details are in the implementation of this idea.

PROOF

SAT is in NP

First, we show that SAT is in NP. A nondeterministic polynomial time machine can guess an assignment to a given formula ϕ and accept if the assignment satisfies ϕ.

SAT is NP-hard

Next, we take any language A in NP and show that A is polynomial time reducible to SAT. Let N be a nondeterministic Turing machine that decides A in n^k time for some constant k. (For convenience, we actually assume that N runs in time $n^k - 3$) The following notion helps to describe the reduction.

Tableau A tableau for N on w is an $n^k \times n^k$ table whose rows are the configurations of a branch of the computation of N on input w, as shown in the following figure.

![[Bachelor 2/Algorithms and complexity/images/Pasted image 20250530190053.png]]

For convenience later, we assume that each configuration starts and ends with a # symbol. Therefore, the first and last columns of a tableau are all #'s. The first row of the tableau is the starting configuration of N on w, and each row follows the previous one according to N's transition function. A tableau is accepting if any row of the tableau is an accepting configuration. Every accepting tableau for N on w corresponds to an accepting computation branch of N on w. Thus, the problem of determining whether N accepts w is equivalent to the problem of determining whether an accepting tableau for N on w exists.

Reduction Now we get to the description of the polynomial time reduction f from A to SAT. On input w, the reduction produces a formula ϕ. We begin by describing the variables of ϕ. Say that Q and Γ are the state set and tape alphabet of N, respectively. Let $C = Q \cup \Gamma \cup \#$. For each i and j between 1 and n^k and for each s in C, we have a variable, $x_{i,j,s}$.

Each of the $(n^k)^2$ entries of a tableau is called a cell. The cell in row i and column j is called cell$[i, j]$ and contains a symbol from C. We represent the contents of the cells with the variables of ϕ. If $x_{i,j,s}$ takes on the value 1, it means that cell$[i, j]$ contains an s.

Now we design ϕ so that a satisfying assignment to the variables does correspond to an accepting tableau for N on w. The formula ϕ is the AND of four parts:

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

Cel constraint The first thing we must guarantee in order to obtain a correspondence between an assignment and a tableau is that the assignment turns on exactly one variable for each cell. Formula ϕ cell ensures this requirement by expressing it in terms of Boolean operations:

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

where $C = s_1, s_2, ..., s_l$. Hence ϕ cell is actually a large expression that contains a fragment for each cell in the tableau because i and j range from 1 to n^k.

The first part of ϕ_{cell} inside the brackets stipulates that at least one variable that is associated with each cell is on. The second part stipulates that no more than one variable is on for each cell. Any assignment to the variables that satisfies ϕ (and therefore ϕ cell) must have exactly one variable on for every cell. Thus, any satisfying assignment specifies one symbol in each cell of the table.

Start constraint Formula ϕ_{start} ensures that the first row of the table is the starting configuration of N on w by explicitly stipulating that the corresponding variables are on:

$$\phi_{start} = x_{1,1,s_1} \wedge x_{1,2,s_2} \wedge ... \wedge x_{1,n^k,s_{n^k}}$$

where s_j is the j-th symbol of w. The first row of the tableau is thus completely specified by ϕ_{start} .

Accept constraint Formula ϕ_{accept} guarantees that an accepting configuration occurs in the tableau. It ensures that q_{accept}, the symbol for the accept state, appears in one of the cells of the tableau by stipulating that one of the corresponding variables is on:

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

This part of the formula ensures that at least one row of the tableau is an accepting configuration.

Move constraint Formula ϕ move guarantees that each row of the tableau corresponds to a configuration that legally follows the preceding row's configuration according to N's rules. It does so by ensuring that each 2×3 window of cells is legal. We say that a 2×3 window is legal if that window does not violate the actions specified by N's transition function. In other words, a window is legal if it might appear when one configuration correctly follows another. For example, say that a, b, and c are members of the tape alphabet, and q_1 and q_2 are states of N. Assume that when in state q_1 with the head reading an a, N writes a b, stays in state q_1, and moves right; and that when in state q_1 with the head reading a b, N nondeterministically either

1. writes a c, enters q_2, and moves to the left, or
2. writes an a, enters q_2, and moves to the right.

Expressed formally, $\delta(q_1, a) = (q_1, b, R)$ and $\delta(q_1, b) = (q_2, c, L), (q, a, R)$.

If the top row of the tableau is the start configuration and every window in the tableau is legal, each row of the tableau is a configuration that legally follows the preceding one.

We prove this claim by considering any two adjacent configurations in the tableau, called the upper configuration and the lower configuration. In the upper configuration, every cell that contains a tape symbol and isn't adjacent to a state symbol is the center top cell in a window whose top row contains no states. Therefore, that symbol must appear unchanged in the center bottom of the window. Hence it appears in the same position in the bottom configuration.

The window containing the state symbol in the center top cell guarantees that the corresponding three positions are updated consistently with the transition function. Therefore, if the upper configuration is a legal configuration, so is the lower configuration, and the lower one follows the upper one according to N's rules. Note that this proof, though straightforward, depends crucially on our choice of a 2×3 window size.

ϕ_{move} stipulates that all the windows in the tableau are legal. Each window contains six cells, which may be set in a fixed number of ways to yield a legal window. Formula ϕ move says that the settings of those six cells must be one of these ways, or

$$\phi_{move} = \bigwedge_{1 \le i < n^k,\ 1 < j < n^k} \left[\bigvee_{\substack{a_1, \cdots, a_6 \\ \text{is a legal window}}} (x_{i,j-1,a_1} \wedge x_{i,j,a_2} \wedge x_{i,j+1,a_3} \wedge x_{i+1,j-1,a_4} \wedge x_{i+1,j,a_5} \wedge x_{i+1,j+1,a_6}) \right]$$

Complexity of the Reduction Next, we analyze the complexity of the reduction to show that it operates in polynomial time. To do so, we examine the size of ϕ. First, we estimate the number of variables it has. Recall that the tableau is an $n^k \times n^k$ table, so it contains n^{2k} cells. Each cell has l variables associated with it, where l is the number of symbols in C. Because l depends only on the TM N and not on the length of the input n, the total number of variables is $O(n^{2k})$.

We estimate the size of each of the parts of ϕ. Formula ϕ_{cell} contains a fixed-size fragment of the formula for each cell of the tableau, so its size is $O(n^{2k})$. Formula ϕ_{start} has a fragment for each cell in the top row, so its size is $O(n^k)$. Formulas ϕ_{move} and ϕ_{accept} each contain a fixed-size fragment of the formula for each cell of the tableau, so their size is $O(n^{2k})$. Thus, ϕ's total size is $O(n^{2k})$. That bound is sufficient for our purposes because it shows that the size of ϕ is polynomial in n. If it were more than polynomial, the reduction wouldn't have any chance of generating it in polynomial time.

To see that we can generate the formula in polynomial time, observe its highly repetitive nature. Each component of the formula is composed of many nearly identical fragments, which differ only at the indices in a simple way. Therefore, we may easily construct a reduction that produces ϕ in polynomial time from the input w.

Thus, we have concluded the proof of the Cook–Levin theorem, showing that SAT is NP-complete.

■