

Complexity Theory

Lecture 5

Anuj Dawar

University of Cambridge Computer Laboratory
Easter Term 2016

<http://www.cl.cam.ac.uk/teaching/1516/Complexity/>

SAT is NP-complete

Cook showed that the language **SAT** of satisfiable Boolean expressions is **NP**-complete.

To establish this, we need to show that for every language L in **NP**, there is a polynomial time reduction from L to **SAT**.

Since L is in **NP**, there is a nondeterministic Turing machine

$$M = (Q, \Sigma, s, \delta)$$

and a bound k such that a string x of length n is in L if, and only if, it is accepted by M within n^k steps.

Boolean Formula

We need to give, for each $x \in \Sigma^*$, a Boolean expression $f(x)$ which is satisfiable if, and only if, there is an accepting computation of M on input x .

$f(x)$ has the following variables:

$S_{i,q}$ for each $i \leq n^k$ and $q \in Q$

$T_{i,j,\sigma}$ for each $i, j \leq n^k$ and $\sigma \in \Sigma$

$H_{i,j}$ for each $i, j \leq n^k$

Intuitively, these variables are intended to mean:

- $S_{i,q}$ – the state of the machine at time i is q .
- $T_{i,j,\sigma}$ – at time i , the symbol at position j of the tape is σ .
- $H_{i,j}$ – at time i , the tape head is pointing at tape cell j .

We now have to see how to write the formula $f(x)$, so that it enforces these meanings.

Initial state is s and the head is initially at the beginning of the tape.

$$S_{1,s} \wedge H_{1,1}$$

The head is never in two places at once

$$\bigwedge_i \bigwedge_j (H_{i,j} \rightarrow \bigwedge_{j' \neq j} (\neg H_{i,j'}))$$

The machine is never in two states at once

$$\bigwedge_q \bigwedge_i (S_{i,q} \rightarrow \bigwedge_{q' \neq q} (\neg S_{i,q'}))$$

Each tape cell contains only one symbol

$$\bigwedge_i \bigwedge_j \bigwedge_\sigma (T_{i,j,\sigma} \rightarrow \bigwedge_{\sigma' \neq \sigma} (\neg T_{i,j,\sigma'}))$$

The initial tape contents are x

$$\bigwedge_{j \leq n} T_{1,j,x_j} \wedge \bigwedge_{n < j} T_{1,j,\sqcup}$$

The tape does not change except under the head

$$\bigwedge_i \bigwedge_j \bigwedge_{j' \neq j} \bigwedge_\sigma (H_{i,j} \wedge T_{i,j',\sigma}) \rightarrow T_{i+1,j',\sigma}$$

Each step is according to δ .

$$\begin{aligned} \bigwedge_i \bigwedge_j \bigwedge_\sigma \bigwedge_q (H_{i,j} \wedge S_{i,q} \wedge T_{i,j,\sigma}) \\ \rightarrow \bigvee_{\Delta} (H_{i+1,j'} \wedge S_{i+1,q'} \wedge T_{i+1,j,\sigma'}) \end{aligned}$$

where Δ is the set of all triples (q', σ', D) such that $((q, \sigma), (q', \sigma', D)) \in \delta$ and

$$j' = \begin{cases} j & \text{if } D = S \\ j - 1 & \text{if } D = L \\ j + 1 & \text{if } D = R \end{cases}$$

Finally, the accepting state is reached

$$\bigvee_i S_{i, \text{acc}}$$

CNF

A Boolean expression is in *conjunctive normal form* if it is the conjunction of a set of *clauses*, each of which is the disjunction of a set of *literals*, each of these being either a *variable* or the *negation* of a variable.

For any Boolean expression ϕ , there is an equivalent expression ψ in conjunctive normal form.

ψ can be exponentially longer than ϕ .

However, **CNF-SAT**, the collection of satisfiable **CNF** expressions, is **NP**-complete.

3SAT

A Boolean expression is in **3CNF** if it is in conjunctive normal form and each clause contains at most 3 literals.

3SAT is defined as the language consisting of those expressions in **3CNF** that are satisfiable.

3SAT is **NP**-complete, as there is a polynomial time reduction from **CNF-SAT** to **3SAT**.

Composing Reductions

Polynomial time reductions are clearly closed under composition.

So, if $L_1 \leq_P L_2$ and $L_2 \leq_P L_3$, then we also have $L_1 \leq_P L_3$.

If we show, for some problem A in **NP** that

$$\text{SAT} \leq_P A$$

or

$$3\text{SAT} \leq_P A$$

it follows that A is also **NP**-complete.

Independent Set

Given a graph $G = (V, E)$, a subset $X \subseteq V$ of the vertices is said to be an *independent set*, if there are no edges (u, v) for $u, v \in X$.

The natural algorithmic problem is, given a graph, find the largest independent set.

To turn this *optimisation problem* into a *decision problem*, we define **IND** as:

The set of pairs (G, K) , where G is a graph, and K is an integer, such that G contains an independent set with K or more vertices.

IND is clearly in **NP**. We now show it is **NP**-complete.