AC32008 Theory of Computation

SATISFIABILITY is NP-complete

School of Science and Engineering – Computing
University of Dundee

In this video:

- Cook's theorem – SATISFIABILITY is **NP**−complete

SATISFIABILITY - the “first” NP-complete problem

It is easy to see that SATISFIABILITY $\in$ **NP**, since a NDTM merely needs to guess a satisfying assignment and check that it works, i.e., that all clauses are satisfied.

However, the much more significant (and long) result is that SATISFIABILITY is **NP**$-$complete. This was proved by Stephen Cook in 1971.

Negation of Literals

The literals are of two types, variables themselves, e.g. u, and the negation of a variable, $\bar{u}$.

So if x is any literal, then

- if x is a variable u, then $\bar{x}$ is just $\bar{u}$.
- if x is a the negation of a variable, $\bar{u}$, then $\bar{x}$ is $\bar{\bar{u}}$ which is just u – the double negation cancels out.

Cook's Theorem: Types of Clauses

The proof often uses a set of clauses to achieve a particular requirement. These are of two types.

Exactly one thing true

Often we want to assert that exactly one of some set of literals, say w, x, y, z, must be true in any satisfying assignment. To do this we use clauses as follows:

- $\{w, x, y, z\}$ - this says that *at least* one of the literals must be true.
- Then to ensure that two or more can't be true, we take a clause for each pair of the literals, negated:

$$\{\bar{w}, \bar{x}\} \ \ \{\bar{w}, \bar{y}\} \ \ \{\bar{w}, \bar{z}\} \ \ \{\bar{x}, \bar{y}\} \ \ \{\bar{x}, \bar{z}\} \ \ \{\bar{y}, \bar{z}\}$$

Now, if say w and y are both true, then the clause $\{\bar{w}, \bar{y}\}$ has both literals false and so is not satisfied.

Cook's Theorem: Types of Clauses

Implications

Often we want to say that **if** some set of literals, say w, x, y, z are all true, **then** some other literal p must also be true in any satisfying assignment.

This is how we can say that **if** the machine is in some configuration at time t, **then** it will be in a certain configuration at time $t + 1$, after the next step.

To do this we use the clause

$$\{\bar{w}, \bar{x}, \bar{y}, \bar{z}, p\}$$

Then if all of w, x, y, z are true, then $\bar{w}, \bar{x}, \bar{y}, \bar{z}$ are all false, so p must be true.

Cook's theorem: SATISFIABILITY is NP-complete

Theorem (Cook, 1971)

The language SATISFIABILITY is **NP**−complete.

Proof: Let L be a language in **NP**. We show there is a polynomial transformation from L to SATISFIABILITY.

Since $L \in$ **NP**, L is recognised by a polynomial-time NDTM M, with alphabet Γ, input alphabet Σ, states Q, transition function δ and special states q_0, q_Y, q_N. Also there is a polynomial $p(n)$ such that $T_M(n) \leq p(n)$ for all n. (We also assume that $p(n) \geq n$.)

We construct a function f_L which maps strings (inputs to M) to instances of SATISFIABILITY, such that $x \in L$ if and only if $f_L(x)$ has a satisfying assignment.

Cook's theorem: SATISFIABILITY is NP-complete

Idea of the proof: construct instance of SATISFIABILITY, $f_L(x)$, which describes the possible computations of M on x, and has a satisfying assignment if and only if at least one of these computations is accepting.

Cook's theorem: SATISFIABILITY is NP-complete

We know that if M accepts x, there is an accepting computation of at most $p(n)$ steps, and so this computation uses (at most) the tape cells numbered $-p(n), \ldots, p(n) + 1$.

At any stage in such a computation, the machine can be completely described by giving the contents of these cells, the tape-head position and the internal state (the Instantaneous Description).

Since the checking stage is completely deterministic, and there are only $p(n) + 1$ distinct "times" during the computation, we can describe the whole of the checking stage by an instance of SATISFIABILITY.

Cook's theorem: SATISFIABILITY is NP-complete

Let $Q = \{q_0, q_1 = q_Y, q_2 = q_N, q_3, \ldots, q_r\}$, and let $\Gamma = \{s_0 = B, s_1, \ldots, s_\nu\}$. We use the following boolean variables:

Variable		Intended meaning
$Q[t, q]$	$0 \leq t \leq p(n)$, $q \in Q$	After t steps (of checking stage), M is in state q
$H[t, c]$	$0 \leq t \leq p(n)$, $-p(n) \leq c \leq p(n) + 1$	After t steps (of checking stage), the head is scanning tape cell c
$S[t, c, s]$	$0 \leq t \leq p(n)$, $-p(n) \leq c \leq p(n) + 1$ $s \in \Gamma$	After t steps (of checking stage), tape cell c contains symbol s

Cook's theorem: SATISFIABILITY is NP-complete

In addition, there is a set of clauses, divided into 6 groups whose object is to mimic the computation.

The clauses are designed so that once the variables which specify the negatively indexed part of the tape at time 0 are assigned to (thus completely determining the guess), then the rest of the variables have to be assigned in a way which describes the checking stage (otherwise some clause is not satisfied).

But in addition, in order to get a complete satisfying assignment, the variable $Q[p(n), q_Y]$ must be true (since it is in a clause on its own), and this will happen precisely if some guess gives an accepting computation. Hence a satisfying assignment can be found if and only if some guess of length at most $p(n)$ gives an accepting computation, i.e., $x \in L$.

Cook's Theorem: Clauses – Group 1

$\{Q[t, q_0], Q[t, q_1], \ldots, Q[t, q_r]\}$, $0 \leq t \leq p(n)$

$\{\overline{Q[t, q]}, \overline{Q[t, q']}\}$, $0 \leq t \leq p(n)$, $q \neq q'$

At each time t, M is in exactly 1 state.

- The first clause says that at each time t, the machine must be in at least one state.
- The remaining clauses say that at each time t and for each pair of states, the machine cannot be in both of these states.)

Cook's Theorem: Clauses – Group 2

$\{H[t, -p(n)], H[t, -p(n)+1], \ldots, H[t, p(n)+1]\}$, $0 \leq t \leq p(n)$

$\{\overline{H[t,c]}, \overline{H[t,c']}\}$, $0 \leq t \leq p(n), -p(n) \leq c < c' \leq p(n)+1$

Similarly, at each time t, the read-write head is scanning exactly one tape square.

Cook's Theorem: Clauses – Group 3

$\{S[t, c, s_0], S[t, c, s_1], \ldots, S[t, c, s_\nu]\}$,
$0 \leq t \leq p(n), -p(n) \leq c \leq p(n) + 1$

$\{\overline{S[t, c, s]}, \overline{S[t, c, s']}\}$, $0 \leq t \leq p(n), -p(n) \leq c \leq p(n) + 1, s \neq s'$

Similarly, at each time t, each tape square contains exactly one tape symbol.

Cook's Theorem: Clauses – Group 4

$\{Q[0, q_0]\}, \{H[0, 1]\}, \{S[0, 0, B]\}$,

$\{S[0, 1, x_1]\}, \{S[0, 2, x_2]\}, \ldots, \{S[0, n, x_n]\}$,

$\{S[0, n+1, B]\}, \{S[0, n+2, B]\}, \ldots, \{S[0, p(n)+1, B]\}$,
where $x = x_1 x_2 \ldots x_n$.

At time 0, the computation is in the initial configuration of its checking stage for input x.

Cook's Theorem: Clauses – Group 5

$\{Q[p(n), q_Y]\}$

M has entered state q_Y at or before time $p(n)$.

Cook's Theorem: Clauses – Group 6(i)

$\{\overline{S[t, c, s]}, H[t, c], S[t+1, c, s]\}$

for $0 \leq t < p(n), -p(n) \leq c \leq p(n) + 1, s \in \Gamma$.

These clauses ensure that any tape cell *not* scanned by the tape head at time t remains unchanged at time $t + 1$.

Cook's Theorem: Clauses – Group 6(ii)

$\{\overline{H[t,c]}, \overline{Q[t,q]}, \overline{S[t,c,s]}, H[t+1, c+D]\}$

$\{\overline{H[t,c]}, \overline{Q[t,q]}, \overline{S[t,c,s]}, Q[t+1, q']\}$

$\{\overline{H[t,c]}, \overline{Q[t,q]}, \overline{S[t,c,s]}, S[t+1, c, s']\}$

for $0 \leq t < p(n), -p(n) \leq c \leq p(n)+1, q \in Q$ and $s \in \Gamma$, and where if $q \neq q_Y, q_N$ then q', s', D are given by

$$\delta(q,s) = (q', s', D)$$

while if $q = q_Y$ or q_N, then $D = 0, q' = q$ and $s' = s$.

This group of clauses ensures that the change from one configuration to the next is precisely as given by the transition function δ.

Remarks on Cook's Theorem

1. The proof above gives a polynomial transformation f which takes a string x and produces an instance $f(x)$ of SATISFIABILITY such that $x \in L$ if and only if $f(x)$ has a satisfying assignment. Using a suitable coding scheme for SATISFIABILITY to give a language corresponding to the informally stated problem, we would get a transformation from the language L to the language SATISFIABILITY.
2. The number of variables used is

   $$(p(n) + 1)(|Q|) + 2(p(n) + 1)^2(1 + |\Gamma|)$$

   and the number of clauses is $O(p(n)^2)$ since $|Q|$ and $|\Gamma|$ are constants.

   Since the calculation of $f(x)$ is straightforward, it is easy to see that the time taken, in terms of the length of x, is polynomial. Hence f is a polynomial transformation.