1 Introduction

We now want to complete the proof of Cook's Theorem (or the Cook-Levin theorem - independently discovered):

Theorem 1.1 (Cook, 1971)**.** SAT *is* **NP***-complete*

We've already established that SAT is in **NP**, so it remains to show that it's **NP**-hard. As noted previously, our plan to do that is to show how to convert the "movie" that shows a non-deterministic Turing machine operating for An^c steps into an instance of SAT by encoding its individual states (and the state of the tape) in boolean variables as "snapshots" and then linking it all together consistently

What's a snapshot?

- A record of the position of the read-write heads (main tape, oracle tape)
- The current state
- The complete contents of the tapes

2 Introducing the variables

The parameters of operation for our machine are:

- the current time (an integer, usually denoted t, between 0 and An^c),
- the current state (also an integer, usually denoted q in some bounded range)
- the current location of the read-write heads (two integers between 0 and An^c)
- the current contents of the tapes (one character per possible location per time step)

We'll capture the time within the variables we use. Each variable will have a name and some associated parameters. The intended interpretation is that when we make a truth

assignment, if the value of a variable is t (for true) then the configuration of the machine at the associated time will have that property. So, our variables will be:

Name	Interpretation
STATE(q, t)	M is in state q at time t.
M-HEAD(j, t)	The read/write head on the main tape is at position j at time t.
O-HEAD(j, t)	The read/write head on the oracle tape is at position j at time t.
M-SYMBOL(s, j, t)	Symbol s is on the main tape at position j at time t.
O-SYMBOL(s, j, t)	Symbol s is on the oracle tape at position j at time t.

If we observe a computation of M we can make truth assignments to these variables that correspond to it. For instance, all the variables with time parameter 0 would be determined by the initial state of the machine, the main input, and whatever the genie had decided to write on the oracle tape.

Conversely, given an assignment of truth values to all these variables it *might* represent a legitimate computation. Our whole goal is to write down a (polynomially-sized) family of clauses in such a way that any satisfying assignment to the clauses represents a valid computation that accepts. In that way, we will have produced a polynomial-time reduction of an instance of problem DECIDE-L (where L is the language accepted by M), i.e., "Does M accept a given word w" into an instance of SAT. Thus, DECIDE-L is polynomial-time reducible to SAT and, since L could be any language in **NP**, we conclude that SAT is **NP**-hard.

A computation of M would be represented by a truth assignment to these variables. The fact that it was a "real" computation could be enforced by clauses defining the restrictions imposed on the operation of M.

3 Getting started

Let's consider a couple of examples before we try to carefully pin down all the details.

How do we express the fact *Position* 3 *of the main tape contains exactly one symbol at time* 23?

We can break this down into two parts: there is a symbol there, and there is only one symbol there. The first part is captured by a single clause:

$$\bigvee_{s \in \Sigma} \text{M-SYMBOL}(s, 3, 23).$$

The latter part is really just "for any two distinct symbols at least one of them is false". That is, it is captured by the collection of all clauses of the form:

$$\neg\text{M-SYMBOL}(s, 3, 23) \vee \neg\text{M-SYMBOL}(s', 3, 23)$$

for all pairs $s, s' \in \Sigma$ with $s \neq s'$.

Note the first clause has size $|\Sigma|$ (which is a fixed constant) and the second group consists of $|\Sigma|(|\Sigma| - 1)/2$ clauses each of size 2. So, each family of clauses of this type (i.e., for other values of 3 and 23) is of constant total size.

What about *If position* 7 *is not under the oracle read/write head at time* 12 *then the symbol at that position is the same at time* 13?

In logical terms, for each $s \in \Sigma$ we assert:

$$\neg\text{O-HEAD}(7, 12) \wedge \text{O-SYMBOL}(s, 7, 12) \implies \text{O-SYMBOL}(s, 7, 13).$$

We haven't used the $\implies$ operator (for "implies") before but fortunately it turns out that:

$$x \implies y \text{ is equivalent to } \neg x \vee y.$$

So the condition above becomes the following clause (after also using one of DeMorgan's laws):

$$\text{O-HEAD}(7, 12) \vee \neg\text{O-SYMBOL}(s, 7, 12) \vee \text{O-SYMBOL}(s, 7, 13).$$

Just to check – let's see how that clause might be true. The oracle-head might be at position 7 at time 12 (in which case we're happy – anything can happen to the oracle symbol there at time 13). If not (so the first part of the clause is false) then either s is not on the oracle-tape in position 7 at time 12 (first remaining part of the clause) or it is, in

which case it must also be there at time 13. So, taking all those over all $s \in \Sigma$ does indeed capture the intended behaviour.

These two examples cover the main themes of the whole construction: making sure the static situation is valid (the first type) and then checking consistency through time (the second type). The rest is just (a lot of) bookkeeping.

4 Groups of clauses

Our reducer will convert M and the relevant part of the two tapes (i.e. only the first An^c characters) into an instance of SATISFIABILITY. There are several groups of clauses in this instance which play different roles:

- A group dealing with the representation being valid at each time step, i.e. things like:
  - There is a unique symbol at each point on the tape.
  - The read/write head is in a specific position.
  - The machine is in some state.
- A group representing the initial configuration.
- A group representing the final configuration.
- A group representing consistency between one frame (time t) and the next (time $t + 1$).

At all times we need to check that everything is bounded by a polynomial in the original input size ($n = |w|$). We'll do this as we go since the groups together just add up their sizes. Note that, since it is fixed throughout, we can treat any parameters relating to M, specifically the number of states and the size of the alphabet as constants and suppress them in O notation.

We will assume that M has states q_0 through q_m (denoted 0 through m in the parameters of our M-SYMBOL and O-SYMBOL variables, which include special states denoted a and

r which are the accepting and rejecting states respectively. These are modeled as "loop states" i.e. the machine simply idles in these states once it reaches one. So we can assume that a computation on input of length n is run for *exactly* An^c steps. Technically this machine is not halting but of course that's not really relevant. We will require that we terminate after An^c transitions in one of these two states and that's all that matters. We now consider the groups one at a time:

4.1 Valid tape

"There is a symbol at each point on each tape at each time"

For $0 \leq t \leq An^c$, and $0 \leq j \leq An^c$:

$$\bigvee_{s \in \Sigma} \text{M-SYMBOL}(s, j, t)$$
$$\bigvee_{s \in \Sigma} \text{O-SYMBOL}(s, j, t)$$

A total of $O(n^{2c})$ clauses each of the same size as the alphabet. So polynomial.

"There are never two symbols at the same point of the tape at the same time." In other words: "For any two symbols, at least one of them is not on a particular point of the tape at a particular time."

For $0 \leq t \leq An^c$, and $0 \leq j \leq An^c$, and for distinct $s, s' \in \Sigma$:

$$\neg\text{M-SYMBOL}(s, j, t) \vee \neg\text{M-SYMBOL}(s', j, t)$$
$$\neg\text{O-SYMBOL}(s, j, t) \vee \neg\text{O-SYMBOL}(s', j, t)$$

A total of $O(n^{2c})$ more clauses each of size 2 so polynomial.

So that's ensured that in a satisfying assignment we have exactly one variable set to true that indicates a particular symbol on each state at each point in time. That is, a satisfying assignment represents genuinely possible tapes.

4.2 Valid machine

"At each time step the tape heads are somewhere."

For $0 \leq t \leq An^c$

$$\bigvee_{0 \leq j \leq An^c} \text{M-HEAD}(j, t)$$
$$\bigvee_{0 \leq j \leq An^c} \text{O-HEAD}(j, t)$$

Two clauses each of size $O(n^c)$.

"And they're not in two places at once"

For each $0 \leq j < j' \leq An^c$ one clause:

$$\neg\text{M-HEAD}(j, t) \vee \neg\text{M-HEAD}(j', t)$$
$$\neg\text{O-HEAD}(j, t) \vee \neg\text{O-HEAD}(j', t)$$

That's $O(n^{3c})$ clauses (n^c from time, and n^{2c} from the (j, j') pairs) each of size 2.

"At each time step the machine is in some state"

For $0 \leq t \leq An^c$

$$\bigvee_{0 \leq i \leq m} \text{STATE}(i, t)$$

$O(n^c)$ clauses of constant size.

"And it's not in two states at once"

For $0 \leq t \leq An^c$, and each $0 \leq i < i' \leq m$:

$$\neg\text{STATE}(i, t) \vee \neg\text{STATE}(i', t)$$

Again $O(n^c)$ clauses of size 2.

5 Initial/Final

"The initial state is 0, the read/write head is at position 0 and the first An^c positions of the tape match w"

$$\begin{gathered}\textsc{State}(0,0)\\ \text{M-}\textsc{Head}(0,0)\\ \text{O-}\textsc{Head}(0,0)\\ \text{M-}\textsc{Symbol}(w_j,j,0)\quad \text{for } 0\le j\le An^c.\end{gathered}$$

That's $O(n^c)$ clauses of size 1.

"The final state is a"

$$\textsc{State}(a,An^c).$$

One clause of size 1.

At this point we have a static representation of a series of snapshots of a machine which in the correct initial configuration, and the correct final configuration and such that the intermediate configurations are all valid ones. What we don't have is any link between configurations at successive time steps.

6 Transitions

"Only the symbol under the read/write head can change"

or

"If the head is somewhere, then no symbol anywhere else can change".

For $0\le t<An^c$, $0\le j,j'\le An^c$, with $j\ne j'$ and all $s,s'\in\Sigma$ with $s\ne s'$:

$$\begin{gathered}\text{M-}\textsc{Head}(j,t)\Rightarrow\neg\left(\text{M-}\textsc{Symbol}(s,j',t)\wedge\text{M-}\textsc{Symbol}(s',j',t+1)\right)\\ \text{O-}\textsc{Head}(j,t)\Rightarrow\neg\left(\text{O-}\textsc{Symbol}(s,j',t)\wedge\text{O-}\textsc{Symbol}(s',j',t+1)\right)\end{gathered}$$

But $a \Rightarrow \neg(b \wedge c)$ is logically equivalent to $\neg a \vee \neg b \vee \neg c$ so this is a family of $O(n^{3c})$ clauses of size 3.

"If we are in state q at position j_m on the main tape and j_o on the oracle tape reading symbols s_m on the main tape and s_o on the oracle tape then we must move to the state which is determined by the transition from q, s_m and s_o"

Suppose that this transition does the following:

- Writes s'_m on the main tape and s'_o on the oracle tape.
- Moves the main head by $\epsilon_m \in \{-1, 0, 1\}$ and the oracle head by $\epsilon_o \in \{-1, 0, 1\}$
- Changes to state q'

Then we want to ensure that:

$$\bigwedge \left\{ \begin{array}{l} \textsc{State}(q, t) \\ \text{M-}\textsc{Head}(j_m, t) \\ \text{O-}\textsc{Head}(j_o, t) \\ \text{M-}\textsc{Symbol}(s_m, j_m, t) \\ \text{O-}\textsc{Symbol}(s_o, j_o, t) \end{array} \right\} \implies \bigwedge \left\{ \begin{array}{l} \text{M-}\textsc{Symbol}(s'_m, j_m, t+1) \\ \text{O-}\textsc{Symbol}(s'_o, j_o, t+1) \\ \text{M-}\textsc{Head}(j_m + \epsilon_m, t+1) \\ \text{O-}\textsc{Head}(j_o + \epsilon_o, t+1) \\ \textsc{State}(q', t+1) \end{array} \right\}$$

But $x \implies y_1 \wedge y_2 \wedge \ldots y_k$ is equivalent to $x \implies y_1$ and $x \implies y_2$ and ... and $x \implies y_k$, and as we've already seen, a conjunction on the left-hand side of an implication becomes a disjunction in the equivalent form. So each of thse implications can be replaced by five clauses of size six. We need to do this for every t, q, j_m, j_o, s_m and s_o but that's still only another $O(n^{3c})$ clauses of constant size.

And that's it!

7 Overview

We have described a mechanical procedure for converting in polynomial time, a non-deterministic Turing machine M with polynomial time complexity and an input word w into a formula $\Phi(M, w)$ in CNF, i.e. an instance of SAT. This translation is such that if

$\Phi(M, w)$ is satisfiable, then any satisfying assignment represents a series of snapshots of a valid accepting computation of M on input w.

Moreover, if such a computation exists, we can use it to set the variables of $\Phi(M, w)$ to produce a satisfying assignment.

So:

M accepts w if and only if $\Phi(M, w)$ is satisfiable.

That is, we have a polynomial time reduction from "The language accepted by M" to SAT. Since M was an *arbitrary* non-deterministic Turing machine with polynomial time-complexity this means that SAT is **NP**-hard, and therefore **NP**-complete (since we already know that it is in **NP**).

Phew.

8 Tutorial problems