

## Cook's Theorem

### The Turing Machine

A Turing Machine TM is described by the septuple

$$TM = \{Q, \Sigma, b, \delta, q_0, q_Y, q_N\} \quad \text{where}$$

$Q$  = a finite set of states including a start state  $q_0$  and accepting/rejecting states  $q_Y$ , and  $q_N$

$\Sigma$  = a set of symbols that are part of the language to be decided

$b$  = the blank symbol

$\delta$  = a set of transitions  $\{ (q_i, s_k) \rightarrow (q_j, s_h, inc) \}$  where

$q_i$  is the initial state,  $q_j$  the new state

$s_k$  the symbol under the tape head before the transition

$s_h$  the symbol in that square after the transition, and

$inc = \{ \rightarrow, \leftarrow \}$  the direction the tape head moves

**Church's Thesis** – any problem solvable in polynomial time on a *physically realizable* computer can be solved in polynomial time on a Turing Machine.

### The Satisfiability Problem (SAT)

Given a set of  $n$  Boolean variables,  $\{x_1, x_2, \dots, x_n\}$  and a set of  $m$  CNF clauses composed of literals of these variables, is there an assignment of truth values (T and F) to these variables such that every clause is satisfied?

A literal is either one of the variables  $x_i$  or its negation  $\overline{x_i}$

A clause in CNF form is a string of literals separated by disjunction (or) with all of the clauses separated by conjunction (and). In our notation, we will leave out the or operators when writing a clause.

Consider the set of variables  $\{x_1, x_2, x_3, x_4\}$  and the clauses

$$F = (x_1 x_3 \overline{x_4}) \wedge (\overline{x_1} \overline{x_2} x_4) \wedge (x_1 \overline{x_2} \overline{x_3} x_4) \wedge (x_2 x_3)$$

Is there an assignment of values T and F for the variables  $x_i$  such that F is satisfied?

Although efficient algorithms exist for deciding many instances of SAT, the only known algorithm for deciding *all* instances of this problem is an exhaustive search of all  $2^n$  possible assignments for the  $x_i$ . In the above example, we can readily see that F is satisfied when  $\{x_1 = T, x_2 = F, x_3 = F, x_4 = T\}$ . (A clause is satisfied when any one of the literals in the disjunction is True.)

While the problem of deciding an instance  $I$  of SAT is “hard” (no known polynomial-time solution), given the statement of the problem instance,  $I$ , and a certificate,  $C(I)$ , the certificate can be readily verified in polynomial-time.

Thus SAT is in the class NP.

### **SAT is in the class NP-Complete (NPC)**

To show that SAT is in NPC, we must show that

1. SAT is in NP (discussed above)
2. For every  $L \subset NP$ ,  $L \leq_P SAT$ . Every language in NP is polynomially reducible to SAT.

Since every  $L \subset NP$  can be verified in  $P(n)$  – time polynomial in  $n$ , the size of the problem instance – on a Turing Machine. All of NP is reducible to a language that describes the verification process on the Turing Machine.

Let  $Q$  be some problem in NP, and let  $I$  be an instance of this problem. Let TMQ be a Turing machine that accepts encoded instance of  $Q$  in polynomial time if accompanied by a suitable certificate. Let  $P(n)$  be a polynomial in  $n$  with the property that TMQ recognizes every pair  $(x, C(x))$  in the language  $Q$ , where  $x$  is a string in the language and  $C(x)$  is its certificate, in time  $\leq P(n)$  where  $n$  is the length of  $x$  (number of symbols in the string  $x$ ).

We need to “reduce” every string  $I$  in the language  $Q$  to an instance  $f(I)$  OF SAT.

The strategy is this – the instance of SAT will be a set of clauses that together express the fact that there exists a certificate that causes TMQ to do an accepting calculation. An instance  $I$  of  $Q$  is verified if and only if the set of clauses in SAT are all true.

First we need to decide which Boolean variables will be used.

Let  $Q_{i,k}$  = the state of the machine is in  $q_k$  after the  $i^{\text{th}}$  step of the checking calculation.

$S_{i,j,h}$  = after step  $i$ , the symbol in square  $j$  is  $h$ .

$T_{i,j}$  = after step  $I$ , the tape head is positioned over square  $j$

How many variables have we just introduced? Since TMQ does its calculation in time  $P(n)$ , the tape head will never venture more than  $|P(n)|$  from its starting position. Thus the index  $j$  that runs through the number of tape squares takes at most  $O(P(n))$  different values.

The index  $i$  runs over the steps of the accepting calculation, and takes at most  $O(P(n))$  different values.

The index  $k$  indexes the (finite) number of states of TMQ and is some fixed number  $K$ . The symbols  $h$ , will be either 0, 1, or blank.

Hence there are all together  $O(P(n)^2)$  variables – a polynomial number of them. We must now list the conditions under which a set of values assigned to these variables represent an accepting condition.

**At each step TMQ is in at least one state**

$$\{Q_{i,0}, Q_{i,1}, \dots, Q_{i,K}\}$$

since  $i$  is  $O(P(n))$  there are  $O(P(n))$  such clauses

**At each step  $i$ , TMQ is in not more than one state**

$$\text{for each step } i, \text{ and each pair } j, j' \text{ of states } \{ \bar{Q}_{i,j}, \bar{Q}_{i,j'} \}$$

another  $O(P(n))$  clauses to add to the list

Note that  $Q_{i,j} \rightarrow \bar{Q}_{i,j'}$  is equivalent to the clause shown above.

**At each step  $i$ , each tape square contains exactly one symbol from  $\Sigma$**

This leads to 2 lists of clauses – there is at least one symbol in each square

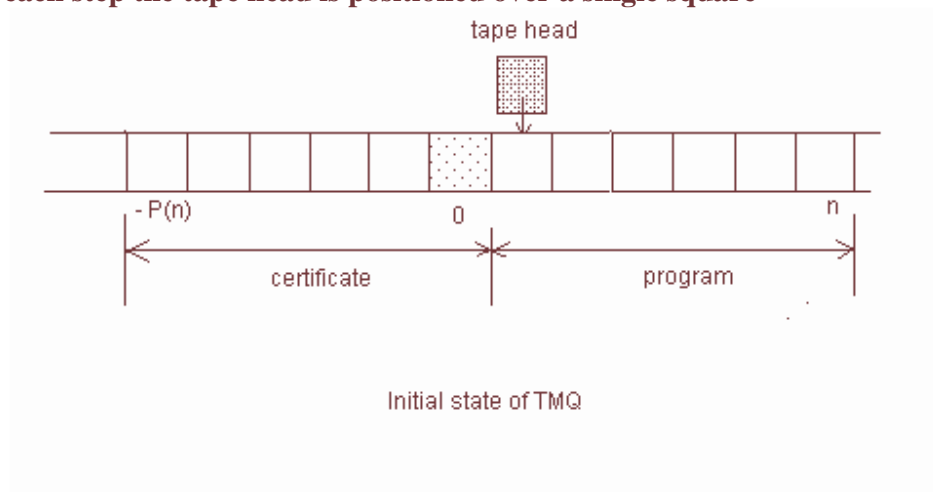
$$\{S_{i,j,0}, S_{i,j,1}, S_{i,j,b}\}$$

and there is not more than one symbol in each square after step  $i$

$$S_{i,j,k} \rightarrow \bar{S}_{i,j,k'} \text{ which is equivalent to } \{ \bar{S}_{i,j,k}, \bar{S}_{i,j,k'} \}$$

with  $O(P(n)^2)$  such clauses

**At each step the tape head is positioned over a single square**



(You add the clauses now)

**At step  $P(n)$  the machine is in state  $q_Y$**

(your turn again)

**At each step the machine moves to its next (state, symbol, head position) in accordance with the application of the program to its previous (state, symbol)**

$$\{ T_{i,j}, \bar{S}_{i,j,k}, S_{i+1,j,k} \}$$

The tape head must be positioned over square j for the symbol written there to change

$$\begin{aligned} & \{ \bar{T}_{i,j}, \bar{Q}_{i,k}, \bar{S}_{i,j,t}, T_{i+1,j+inc} \} \\ & \{ \bar{T}_{i,j}, \bar{Q}_{i,k}, \bar{S}_{i,j,t}, Q_{i+1,k'} \} \\ & \{ T_{i,j}, Q_{i,k}, S_{i,j,t}, S_{i+1,j,t'} \} \end{aligned}$$

In the last 3 sets of clauses we read : either the tape head is not positioned at square j, or the present state is not  $q_k$ , or the symbol just read is not t, but if they are then ...

Then there is a clause for each step of the operation i, for each square j,  $-P(n) \leq j \leq P(n)$  of the tape, for each symbol t of the alphabet, and for each possible state  $q_k$  of TMQ – a polynomial number of clauses in all.

Now if we execute a recognizing computation on TMQ of a string x and its certificate,  $C(x)$ , in time at most  $P(n)$ , then this computation determines an assignment of values T and F to the variables listed above such that all of the clauses are simultaneously satisfied.

Conversely, if we have a set of values of the SAT variables in which all of the clauses are simultaneously satisfied, then that set of values describes a certificate that would cause TMQ to do a recognizing calculation on string x of Q.

**We now have our first member of the class NP-Complete!!!!**