Cook's Theorem: Satisfiability in NP-Complete

To be NP-complete we must show two properities:

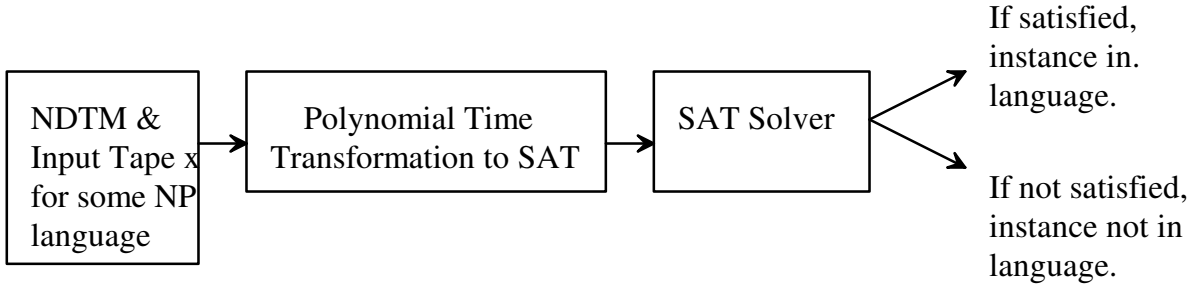
1) The problem is in NP, i.e., we can verify a claimed solution in polynomial time.

2) That every NP problem can be reduced to it in polynomial time.

For the Satisfiability problem (SAT), the first property is relatively easy to see. We just need to see if the claimed truth assignment for the boolean variables results makes every clause *true*, or not.

For the second property is much more difficult since there are an infinite number of diverse problems that are in NP. To show this, we will use the language level view of NP problems, i.e., any problem in NP has a nondeterministic Turing Machine (NDTM) program which solves it in polynomial time, specifically p(n). Therefore, each language in NP can be described in a standard way by giving a polynomial time NDTM program that recognizes it.

We will take this NDTM program and create from it an instance of SAT such that it is satisfiable if and only if the input string was in the language.



To complete the proof, we must show how to construct a polynomial-time transformation to SAT from an arbitrary polynomial time NDTM program M specified by:
Γ ("gamma") - set of tape symbols including a blank symbol *b,*
Σ ("sigma") - set of input symbols (a subset of Γ not including *b*),
Q - set of states,
q_0 - special state in Q called the start state,
q_Y - special state in Q that is a final state, called the *accepting state*,
q_N - special state in Q that is a final state, called the *non-accepting state*, and
δ ("delta") - the *next-move function* that maps the current state and current input-tape symbol under R/W head to the next state, replacement input-tape symbol, and direction to move R/W head (Left = -1 or Right = 1one spot on input tape).

If the input $x \in \Sigma^*$ is accepted by M, then we know that there is an accepting computation by M on x such that both the number of computation steps in the checking stage and the number of symbols written in the guessing string are bounded by a polynomial, say p(n), where n is the length of input

string *x*. Since M starts with its R/W head on tape square 1 and moves at most one square per move, then the computation could only involve tape squares -p(n) to p(n) + 1.

The transformation will use boolean variables to maintain the state of TM during the computation. Label the elements of Q as q_0, $q_1=q_Y$, $q_2=q_N$, q_3, ... , q_r, where r = |Q| - 1, and the elements of Γ as $s_0=b$, s_1, ... , s_v, where v = |Γ| - 1.

Boolean Variable	Range	Intended Meaning
Q[i, j]	0 <= i <= p(n) 0 <= j <= r	At time i, M is in state q_j.
H[i, j]	0 <= i <= p(n) -p(n)<= j <= p(n)+1	At time i, the R/W head is scanning tape square j.
S[i, j, k]	0 <= i <= p(n) -p(n)<= j <= p(n)+1 0 <= k <= v	At time i, the contents of tape square j is symbol s_k.

Note that there are a polynomial number of literals r p(n) + $2p(n)^2$ + 2vp(n).

An arbitrary true assignment to these boolean variables might have the computation in multiple states at the same time, have a given tape square containing multiple symbols, etc. All of which we want to prevent. The transformation will use clauses to restrict a truth assignment of the SAT expression to true iff an accepting computation of TM on input-string *x* completely.

The clauses of the transformation can be divided into six categories based on their role:

Clause Group	Restriction Imposed	Clauses in Group
G_1	At each time i, M is in exactly one state.	$(Q[i,0] \vee Q[i,1] \vee ... \vee Q[i,r]), 0 \le i \le p(n)$ $\left(\overline{Q[i,j]} \vee \overline{Q[i,j']}\right), 0 \le i \le p(n), 0 \le j < j' \le r$
G_2	At each time i, the R/W head is scanning exactly one tape square.	$(H[i,-p(n)] \vee H[i,-p(n)+1] \vee ... \vee H[i,p(n)+1]), 0 \le i \le p(n)$ $\left(\overline{H[i,j]} \vee \overline{H[i,j']}\right), 0 \le i \le p(n), -p(n) \le j < j' \le p(n)+1$
G_3	At each time i, each tape square contains exactly one symbol from Γ.	$(S[i,j,0] \vee S[i,j,1] \vee ... \vee S[i,j,v]), 0 \le i \le p(n), -p(n) \le j \le p(n)+1$ $\left(\overline{S[i,j,k]} \vee \overline{S[i,j,k']}\right), 0 \le i \le p(n), -p(n) \le j \le p(n)+1, 0 \le k < k \le v$
G_4	At time 0, the computation is in the initial configuration of its checking stage for input x.	$(Q[0,0]) \wedge (H[0,1]) \wedge (S[0,0,0]) \wedge$ $(S[0,1,k_1]) \wedge (S[0,2,k_2]) \wedge ... \wedge (S[0,n,k_n]) \wedge$ $(S[0,n+1,0]) \wedge (S[0,n+2,0]) \wedge ... \wedge (S[0,p(n)+1,0])$, where $x = s_{k_1} s_{k_2} ... s_{k_n}$

Clause Group	Restriction Imposed	Clauses in Group
G_5	By time p(n), M has entered state q_Y and hence has accepted x.	$(Q[p(n),1])$
G_6	For each time i, $0 \le i < p(n)$, the configuration of M at time i+1 follows by a single application of the transition function δ from the configuration at time i.	

The clauses for group G_6 guarantees that the changes from one configuration to the next are in accordance with the transition function δ for M. A configuration for M involves the time i, R/W head position j, current state k, and current tape symbol l, where $0 \le i < p(n), -p(n) \le j \le p(n)+1, 0 \le k < r$, and $0 \le l \le v$. For each possible configuration, three clauses are added
$\left(\overline{H[i,j]} \vee \overline{Q[i,k]} \vee \overline{S[i,j,l]} \vee H[i+1,j+\Delta]\right)$,
$\left(\overline{H[i,j]} \vee \overline{Q[i,k]} \vee \overline{S[i,j,l]} \vee Q[i+1,k']\right)$,
$\left(\overline{H[i,j]} \vee \overline{Q[i,k]} \vee \overline{S[i,j,l]} \vee S[i+1,j,l']\right)$, where if $q_k \in Q-\{q_Y, q_N\}$, then the values of Δ, k', and l' are such that $\delta(q_k, s_l) = (q_{k'}, s_{l'}, \Delta)$, and if $q_k \in \{q_Y, q_N\}$, then $\Delta = 0, k' = k$, and $l' = l$.

Thus, the SAT expression C formed by the conjunction of clauses G_1 through G_6 performs the previously stated goal: If x∈L, then there is an accepting computation of M on x of length p(n) or less, and this computation imposes a truth assignment that satifies all clauses. Conversely, the construction of this SAT is such that any satisfying truth assignment for it must correspond to an accepting computation of M on x.

All that remains is to show that transformation can be performed in polynomial time in length of x. The number of literals is $O(p(n)^2)$ and the number of clauses is $O(p(n)^2)$. A reasonable encoding of the SAT expression would involve $O(p(n)^4)$ items since each clause would contain at most each literal.

Reference:
Computers and Intractability: A Guide to the Theory of NP-Completeness, Michael R. Garey and David S. Johnson, W. H. Freeman and Co., 1979