Satisfiability problem and NP Completeness

Vitaly Parnas

March 16, 2018

We survey the classic CNF (Conjunction Normal Form) Satisfiability problem and then proceed to demonstrate it's NP Completeness by means of a comparatively terse proof in contrast to the traditional Cook-Levin[1, 2, 3], yet explanatory in certain points that might not otherwise seem intuitive.

The Satisfiability (SAT) problem is defined by of a boolean formula in Conjunction Normal Form (CNF). The CNF form requires a formula to contain a conjunction of clauses, each a disjunction of literals. Here's one example:

$$\underbrace{(x_1 \vee x_2 \vee x_3)}_{c_1} \wedge \underbrace{(\overline{x_2} \vee x_3 \vee x_4)}_{c_2} \wedge \underbrace{(x_1 \vee \overline{x_3})}_{c_3}$$

The above example consists of *clauses* c_1, c_2, c_3, *variables* x_1, x_2, x_3, x_4, and *literals* $x_1, x_2, \overline{x_2}, x_3, \overline{x_3}, x_4$.

The formula is *satisfiable* if a variable truth assignment exists that causes the formula to evaluate to true. For example, the assignment $x_1 = True, x_2 = False$ causes the above formula to evaluate to true irrespective of the other variable assignments.

The decision version of the SAT problem simply determines whether *some* truth assignment exists.

Theorem. *SAT is NP Complete.*

Proof. 1) SAT $\in$ NP: Construct a Nondeterministic Turing Machine (NTM) to nondeterministically explore each possible truth assignment t of the SAT

formula Ψ. For each t, a polynomial computation trivially suffices to determine if t satisfies the formula.

2) SAT is NP Hard: Let's generate a polynomial reduction from all languages $L \in NP$ to SAT by leveraging a NTM M deciding L and creating a series of SAT clauses corresponding to all of the valid configurations and transitions of M. We'll proceed to show that $L \in NP$ if and only the resulting formula Ψ has a satisfying truth assignment, or rather, M accepts input w if and only Ψ evaluates to true.

First, provided that $L \in NP$, we can assume that based on input of length n, the NTM M will require a polynomial number steps to execute, $p(n)$. Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$, defining the states, input alphabet, tape alphabet, transition function, initial state, accepting state, and reject state. Now, let's construct a SAT formula Ψ in terms of the components of M and the function p.

States At step i M must reside in precisely one state. Let a variable Q_{ij} be true if and only if at step i M is in state j, for $i \in \{0, ..., p(n)\}$ and $j \in \{1, ..., |Q|\}$. From here on, let us also assume, without loss of generality, that M remains on the positive side of the tape[1].

The complexity described to the right reflects the number of tape symbols necessary to encode the corresponding clauses. We exclude the M-defined constants $|Q|$, and later $|\Gamma|$, from the complexity, and require $O(\log n)$ bits to encode each literal. In fact, the reduction time complexity corresponds nearly to the encoding length complexity but without the logarithmic factor, and omitted from further analysis.

Clause	Encoding Complexity
$(Q_{i0} \vee Q_{i1} \vee ... \vee Q_{i\|Q\|}) \quad \forall i$	$O(\|Q\|p(n)\log n) = O(p(n)\log n)$.
$(\overline{Qij} \vee \overline{Q_{ij'}}) \quad \forall i, \forall j \neq j'$	$O(\|Q\|^2 p(n)\log n) = O(p(n)\log n)$

Tape Head At step i, the tape head can only reside in one position $j \in \{0, ..., p(n)\}$. Let the variable H_{ij} designate this position. Note that after $p(n)$ steps, the tape head could have moved at most $p(n)$ cells to the right.

Clause	Encoding Complexity
$(H_{i0} \vee H_{i1} \vee ... \vee H_{i,p(n)}) \quad \forall i$	$O(p^2(n)\log n)$
$(\overline{Hij} \vee \overline{H_{ij'}}) \quad \forall i, \forall j \neq j'$	$O(p^3(n)\log n)$

[1] To the contrary, we would consider $2p(n)$ possible cell positions, a growth of only a constant factor

Tape The tape must contain only one symbol per square per step. Let S_{ijk} be true if at step i the tape contains symbol $k \in \{0, ..., |\Gamma|\}$ at square j.

Clause	Encoding Complexity
$(S_{ij0} \vee S_{ij1} \vee ... \vee S_{ij|\Gamma|}) \quad \forall i, j \in \{0, ..., p(n)\}$	$O(\|\Gamma\|p^2(n) \log n) = O(p^2(n) \log n)$
$(\overline{Sijk} \vee \overline{S_{ijk'}}) \quad \forall i, j \in \{0, ..., p(n)\}, \forall k, k'.k \neq k'$	$O(\|\Gamma\|^2 p^2(n) \log n) = O(p^2(n) \log n)$

Initial Configuration At step 0 M contains the initial configuration, including the initial state q_0, head position over first square, the input w written on the first part of the tape, and the rest of the tape blank.

Clause	Encoding Complexity
(Q_{0,q_0})	$O(\log n)$
(H_{01})	$O(\log n)$
$(S_{01w_1} \vee S_{02w_2} \vee ... \vee S_{0\|w\|w_{\|w\|}})$	$O(\|w\| \log n) = O(\log n)$
$(S_{0,\|w\|+1,0} \vee ... \vee S_{0,p(n),0})$	$O(p(n) \log n)$

Terminates in accepting state By step $p(n)$, M has entered an accepting state q_a.

Clause	Encoding Complexity
$(Q_{p(n),q_a})$	$O(\log n)$

Transitions Here we construct clauses corresponding to accepting configurations by successive steps in M. The first series of clauses insures that a cell symbol does not change unless the tape head resided in that cell at the previous step.

The second series of clauses insures that an invalid transition cannot occur. For example, with M in state k at step i, and the head over cell j containing symbol l (the first three literals unsatisfied), for any invalid transition $(q_{k'}, s_{l'}, \Delta) \notin \delta(q_k, s_l)$, at least one of the second three literals must be satisfied. By satisfying these clauses, we leave the possibility of only valid transitions having occurred.

Clauses:

$(H_{ij} \vee \overline{S_{ijl}} \vee S_{i+1,j,l})$

$(\overline{H_{ij}} \vee \overline{Q_{ik}} \vee \overline{S_{ijl}} \vee \overline{H_{(i+1)(j+\Delta)}} \vee \overline{Q_{(i+1)k'}} \vee \overline{S_{(i+1)jl'}}$

$\forall i, j \in \{0, ..., p(n)\}, k \in \{1, ..., |Q|\}, l \in \{0, ..., |\Gamma|\}$, and any $(q_{k'}, s_{l'}, \Delta) \notin \delta(q_k, s_l)$

Encoding Complexity: $O(|Q|^2|\Gamma|^2 p^2(n) \log n) = O(p^2(n) \log n)$

The above conjunction of clauses in formula Ψ is satisfied if and only if $w \in L$, that is, machine M yields an accepting computation on input w. We have demonstrated the polynomial reduction encoding length with respect to input of size n, with the respective $O(\log n)$ factor added to account for encoding each literal. The reduction computation itself also incurs the same polynomial cost (without the log factor) to iterate all combinations of steps and cells and generate the appropriate clauses, with Q and Γ constant.

□

References

[1] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[2] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM.

[3] B. A. Trakhtenbrot. A survey of russian approaches to perebor (brute-force searches) algorithms. *Annals of the History of Computing*, 6(4):384–400, Oct 1984.