Theory of computation **Fall 2015**

Lecture 22: Satisfiability Problem

Faculty: K.R. Chowdhary *: Professor of CS*

22.1 Satisfiability Problem

It is concerned with truth values of propositional formulas formulas in propositional logic. It was shown to be the first decision problem that is NP-complete. The objective of satisfiability problem is to determine whether there is an assignment of truth values to propositions that make the CNF (conjunctive normal form) formulas as true. A CNF formula is $u_1 \wedge u_2 \wedge \wedge \cdots \wedge u_n$, where u_i is a clause (i.e., consisting of disjunction of variables, say, $u_i = V_{i_1} \vee V_{i_2} \vee \cdots \vee V_{i_k}$, where V_{i_j} is an atom).

A deterministic solution to the satisfiability problem can be obtained by checking every truth assignment. For n number of Boolean variables, the number of truth assignments are 2^n, which results to exhaustive search, and the complexity is exponential.

Theorem 22.1 *The SAT (Satisfiability) problem is in NP.*

Proof. Let the set of variables be $\{x_1, x_2, \ldots, x_n\}$. The encoding rule followed for positive and negative variables is as follows:

Literal	Encoding
x_i	$\bar{i}\#1$
$\neg x_i$	$\bar{i}\#0$

where $\bar{i}$ is encoding for x_i. Thus, encoding for $(x_1 \vee \neg x_2) \wedge (\neg x_3 \vee x_4)$ is $1\#1 \vee 10\#0 \wedge 11\#0 \vee 100\#1$. The input to the machine is: encoded list of variables appearing in the formula, followed with encoding of the formula itself. Thus, input to TM as per above format is:

$$1\#10\#11\#100 \;\#\#\; 1\#1 \vee 10\#0 \wedge 11\#0 \vee 100\#1,$$

which is called an instance of the SAT problem. With alphabet set as $\Sigma = \{0, 1, \wedge, \vee, \#\}$, the language L_{SAT} is:

$$L_{SAT} = \{w \mid w \in \Sigma^*, \text{ and } w \text{ is in standard form as } CNF\}.$$

A two tape NDTM M that solves SAT problem is described below. The M employs "guess and check" strategy. The guessing generates nondeterministically, a truth assignment. The configurations corresponding to the computation initiated with the input string $w = (x_1 \vee \neg x_2) \wedge (\neg x_3 \vee x_4)$ are demonstrated in the following, to test the satisfiability of a general CNF expression. The initial configuration is:

Tape 2: BB

Tape 1: B1#10#11#100 ## 1#1 $\vee$ 10#0 $\wedge$ 11#0 $\vee$ 100#1B

1. If input on Tape 1 is not in standard format of CNF, computation halts, and input is rejected
2. The encoding of the first variable on tape 1 is copied on tape 2. This is followed with # and nondeterministically writing 0 or 1. This is repeated for the remaining variables. After this ## is written. If t is truth assignment, then for the variable x_i its value is $t(x_i)$. Now, we have following on the tapes:

   Tape 2: B1#$t(x_1)$##10#$t(x_2)$##11#$t(x_3)$##100#$t(x_4)$B

   Tape 1: B1#10#11#100 ## 1#1 $\vee$ 10#0 $\wedge$ 11#0 $\vee$ 100#1B
3. The tape head on Tape 2 is repositioned at the beginning of the tape, and head on tape 1 is moved past ## to first variable in the formula.

   Note: The generating of truth assignment on tape 2 is the only instance of nondeterminism for M. The remainder of the computation checks whether the formula is satisfied by the nondeterministically selected truth assignment.
4. Next, when encoding of x_i is scanned on tape 1, the truth assignment for this variable is found on tape 2. The subsequent action of the machine are determined by the result of comparing value of the Boolean variable following x_i on tape 1 (0 for x_i negative and 1 for positive) with the $t(x_i)$ on tape 2.
5. If they above do not match, current literal is not satisfied by the truth assignment. If the symbol following the literal on tape 1 is $\wedge$ or B, every literal in the current clause has been examined and has failed. (A clause is disjunctive expression of atoms: $X_1 \vee \cdots \vee X_n$). In this case, the truth assignment does not satisfy the formula, and M halts in a non-accepting state.
6. If "$\vee$" is read at tape 1, the tape heads are positioned to examine the next literal in the clause (go to step 4).
7. If the values match, the literal and current clause are satisfied by the truth assignment. The head on tape 1 moves to right to the next $\vee$ or B. If a B is encountered, the computation halts and accept the input. Otherwise, the next clause is processed by returning to step 4.

The matching procedure in step 4 determines the rate of growth of time complexity of computation. In the worst case, the matching requires computing the variables on tape 1 with each of the variables on Tape 2 to discover the match. This is done in time $O(k.n^n)$, where n is number of variables, and k is number of literals in the input.

Since the solution using NDTM is in **P**, the SAT is **NP**.

This proves the Theorem. □