SAT and its variants

Denis Trystram

February 11, 2019

Motivation. The satisfiability problem *SAT* was the first problem to be NP-complete. The study of its variants allows to introduce some proof techniques and also to stress the frontier between polynomial and NP-complete problems.

1 Preliminaries: Presentation of the SAT problem

SAT is a problem coming from the first-order logic (predicates). Let us recall that a predicate is defined over a set of logic variables using one of the three following elementary operations:

- negation NOT ($\neg x$ also denoted by $\bar{x}$)
- conjunction AND ($x \wedge y$)
- disjonction OR ($x \vee y$).

A *literal* is a predicate composed of a single logic variable x (or its negation). A *clause* is a particular predicate formed by disjunctions of literal, for instance $C = x \vee \bar{y} \vee z$.
SAT consists in deciding if there exists a truth assignment that satisfies all the clauses of a given set of clauses.

SAT (Satisfiability)
Instance. A set of n boolean variables (literals) and a collection of m clauses C_i expressed from these lliterals
Question. Is there a satisfying truth assignment for $C_1 \wedge C_2 \wedge ... \wedge C_m$?

For instance: $(x \vee \bar{y} \vee z) \wedge (\bar{x} \vee \bar{y}) \wedge (x \vee \bar{z})$ is satisfiable with the truth assignment $\pi(x) = \mathit{true}$ et $\pi(y) = \mathit{false}$.

2 Brief overview of the Cook theorem

Cook proved in 1971 that there is one particular problem in $\mathcal{NP}$-complete. This problem is the satisfiability problem (called SAT) and it has the property

that every other problem in $\mathcal{NP}$ can be polynomially reduced to it. This means that if any problem in $\mathcal{NP}$ is intractable, then, SAT is also intractable. Notice here that if SAT can be solved in polynomial time, this will be the case for any other problem in $\mathcal{NP}$! The question of whether or not NP-complete problems are intractable is considered since the 70ties to be one of the most challenging open question in theoretical Computer Science. Today, most researchers are conjecturing that NP-complete problems are intractable.
The idea for proving that SAT is in $\mathcal{NP}$-complete was to use a smart coding of the execution of any algorithm on a non-deterministic Turing Machine by a boolean expression written as conjunctions of clauses. In other words:
$\forall L \in \mathcal{NP}$ there exists a polynomial reduction of L to SAT.
First, it is easy to verify that SAT is in $\mathcal{NP}$. A certificate is obtained by a truth assignment of the n variables, which can be coded in a n-bits vector. The verification is clearly polynomial (more precisely, it is in $\mathcal{O}(nm)$).
The difficulty of the reduction is that we should prove it for all the languages L in $\mathcal{NP}$ where the only clue is the existence of a non-deterministic Turing Machine that accepts it in polynomial time.

Let us recall that the notion of execution time on a non-deterministic TM on a word $\omega \in L$ is the minimum of the execution times among ALL the executions that accept ω. Coding the execution on a TM that accepts L as a logic expression (set of clauses) needs to code the data of L, all the informations including the states transitions, the read-write header and the tape.
Cook showed that there exists a transformation that associates an instance L_{SAT} to each word ω and to each language (*i.e.* problem) $L \in \mathcal{NP}$ iif $\omega \in L$.

3 A first variant: kSAT

We denote by kSAT the variant of SAT where all the clauses are composed by exactly k literals ($k \geq 2$).

3.1 3SAT

The variant obtained for $k = 3$ is particularly popular and useful. Let us prove the (simple) following result:

Theorem 1 *3SAT is NP-complete.*

Proof 1 *Let us first verify that 3SAT $\in \mathcal{NP}$, which is straightforward since it is a subproblem of* SAT *and* SAT $\in \mathcal{NP}$.
The idea of the reduction is to rewrite each clause of SAT as a clause of cardinality 3 *by adding adequate variables.*

- *For the clauses with 2 literals, we add a new variable and we create two new clauses that do not change the evaluation of the formula.*

  For instance for the clause $(x_1 \vee x_2)$ of cardinality 2, we add the variable y and we rewrite the initial clause as:

  $(x_1 \vee x_2 \vee y) \wedge (x_1 \vee x_2 \vee \bar{y})$.

  For the clauses with only one literal, we add similarly 2 variables y and z.

- *For the clauses of cardinality p ($p > 3$), we add $p-3$ boolean variables and as many clauses.*

  For instance, for the clause $C = (x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5)$ of cardinality $p = 5$, we introduce 2 new variables y_1 and y_2 as follows:

  $(x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5) = (x_1 \vee x_2 \vee y_1) \wedge (\bar{y_1} \vee x_3 \vee y_2) \wedge (\bar{y_2} \vee x_4 \vee x_5)$.

  It keeps the same value.

The whole reduction from SAT to 3SAT is easy to verify and it is left to the reader.

3.2 3SAT NAE

3SAT NAE (Non All Equal)
Instance. m clauses $C_i = (x_{i,1} \vee x_{i,2} \vee x_{i,3})$ expressed from n literals.
Question. Is there a satisfying truth assignment for $C_1 \wedge C_2 \wedge ... \wedge C_m$ where the literals within the clauses can not all take the same value?

The reduction transforms an instance of 3SAT (with m clauses $C_i = x_{i,1} \vee x_{i,2} \vee x_{i,3}$) into the following instance of 3SAT NAE:
keep the same boolean variables $x_{i,j}$, add m variables y_i plus one variable denoted by f.
$2m$ clauses $C'_i = (x_{i,1} \vee x_{i,2} \vee y_i)$ and $(C''_i = x_{i,3} \vee \bar{y_i} \vee f)$

Theorem 2 *3SAT NAE is NP-complete.*

Proof 2 *Let first show that any positive instance of SAT is transformed to a positive instance of 3SAT NAE by the previous reduction.*
We start by assuming that there exists a truth assignment of 3SAT. We keep the same assignment for these variables and complete it by $y_i = \neg(x_{i,1} \vee x_{i,2})$ and set $f = false$.
This assignment is valid for the transformed instance of 3SAT NAE since

- *if $x_{i,1}$ or $x_{i,2}$ are true then the first clause C'_i is true and y_i is false (one or two literals are true).*

  The second clause C''_i is also true since $\neg y_i$ is true, f is false and again one or two literals are true.

- *if both $x_{i,1}$ or $x_{i,2}$ are false, then $x_{i,3}$ is true according to 3SAT and y is true while f is false.*

The reciprocity is obtained by assuming that there exists a truth assignment for 3SAT NAE.

- *if f is false we take the complementary assignment for each $x_{i,j}$ in 3SAT. Indeed, in this case, either $x_{i,3}$ or $\neg y_i$ are true according to C_i''.*
- *if f is true we take the complementary assignment for each $x_{i,j}$ in 3SAT.*

  Both $x_{i,3}$ and $\neg y_i$ can not be true simultaneously in C_i'' and according to the clause C_i' either $\neg(x_{i,1} \vee x_{i,2})$ or y_i are true this leads to a false assignment of all x variables.

3.3 EO-3SAT

EO-3SAT (Exactly One 3SAT)
Instance. m clauses $C_i = x_{i,1} \vee x_{i,2} \vee x_{i,3}$ expressed from n boolean variables (literals)
Question. Is there a satisfying truth assignment for $C_1 \wedge C_2 \wedge ... \wedge C_m$ where exactly one literal per clause is *true*?

The core of the reduction is to transform each clause $C_i = (x_{i,1} \vee x_{i,2} \vee x_{i,3})$ of an instance of 3SAT into three clauses as follows:
$(x_{i,1} \vee \alpha_i \vee \beta_i)$, $(\bar{x_{i,2}} \vee \alpha_i \vee \gamma_i)$ and $(\bar{x_{i,3}} \vee \beta_i \vee \theta_i)$.
The number of variables is equal to $n+4m$ (the same as in 3SAT augmented by 4 new variables per clause).

The proof follows the same pattern as the previous one and it is left to the reader.

4 Variants of 2SAT

2SAT
Instance. m clauses C_i composed of at most two literals with n literals
Question. Is there a truth assignment that satisfies $C_1 \wedge C_2 \wedge ... \wedge C_m$?

Theorem 3 *2SAT is polynomial*

Proof 3 *The proof is constructive.*
We consider any variable, say x. Let replace x in all the clauses where it appears as follows.

- *if x appears as it it, the clause is satisfied.*

- *if the variable appears as $\bar{x}$ in a clause, we fix the truth assignment of the other variable of the clause in order to satisfy it. This determines a set of variables, denoted by $\mathcal{X}$.*

  There are two possibilities then: if all these clauses are satisfied, we resume the process with the remaining variables (those which do not belong to $\mathcal{X}$).

  If the clauses are not all satisfied, we start again with the value $x = false$.

  If again the clauses are not satisfied, the answer of 2SAT is NO.

Notice that there exists another construction which transforms the formula $\Phi = C_1 \wedge C_2 \wedge ... \wedge C_m$ into a graph $G(\Phi)$ with $2n$ vertices (each vertex corresponds to variables in the two states x and $\bar{x}$).
Notons pour finir que cette formulation permet également de montrer que 2SAT est NL-complet.

5 Max2SAT

Instance: m clauses C_i composed of at most two literals with n variables, an integer K.
Question. Is there a satisfying truth assignment for at least K clauses?

Theorem 4 *max2SAT is NP-complete.*

Proof 4 *Let us first remark that it is simple to verify that Max2SAT is in $\mathcal{NP}$.*
The proof of the theorem is obtained by a reduction from 3SAT, it illustrates the widget technique. A widget is a sub-set of the instance for which a property remains valid. Here, the widget is a particular 2SAT composed of 10 clauses where at most 7 clauses are satisfied when at least one literal is true. The verification is done by a case by case analysis (see below).
For each clause $C_i = (x_{i,1} \vee x_{i,2} \vee x_{i,3})$, we consider the 3variables x plus a new literal (denoted by p_i) and the 10 following clauses of cardinality at most 2:

- *$(x_{i,1})$, $(x_{i,2})$, $(x_{i,3})$, (p_i)*
- *$(\bar{x}_{i,1} \vee \bar{x}_{i,2})$, $(\bar{x}_{i,2} \vee \bar{x}_{i,3})$, $(\bar{x}_{i,1} \vee \bar{x}_{i,3})$*
- *$(x_{i,1} \vee \bar{p}_i)$, $(x_{i,2} \vee \bar{p}_i)$, $(x_{i,3} \vee \bar{p}_i)$*

There is no way to satisfy all the clauses simultaneously.
As the widget is symmetric in regard to $x_{i,1}$, $x_{i,2}$ and $x_{i,3}$, it is sufficient to verify the following cases:

- *if all the variables x are true, we loose the three clauses of the second row, but the three first clauses of the first row and all the clauses of the third row are satisfiable.*

  Now, let discuss the instantiation of p_i.

  If it is true, we gain one clause in the first row (thus, 7 clauses are satisfied otherwise there are only 6).

  If it is false, we loose one clause in the first row (thus, 6 are satisfied).

- *if two of the three x variables are true, we also satisfy* 7 *clauses whatever the value of p_i.*

- *if only one x variable is true,* 7 *clauses are satisfied if p_i is false and* 6 *if it is true.*

- *if the three x variables are false, we can not satify more than* 6.

These last 10 clauses have the interesting property that any truth assignment that satisfies a clause $(x_1 \vee x_2 \vee x_3)$ can be extended to satisfy at most 7 *clauses of the widget. This suggests immediately a reduction of Max2SAT from 3SAT.*

6 One step further: SAT and the space complexity

Classe NL (problèmes qui s'exécutent en espace poly-logarithmique en la taille de l'instance sur une TM non-déterministe). L'opérateur de réduction s'entend ici au sens ...
De même que pour NP, on montre qu'il existe un problème NL-complet (c'est le Graph accessibility Problem – ou GAP).

GAP
Instance. un graphe G=(V,E) orienté et deux sommets x et y appartenant à V.
Question. existe-t-il un chemin de x à y ?

On peut montrer que 2SAT appartient à NL car peut être réduit (au sens de la réduction poly-log) à partir du problème GAP. La réduction utilise la transformation d'une formule logique en graphe.

7 Synthesis

- SAT is NP-complete(Cook 1971)
- kSAT is NP-completepour $k \geq 3$ (en particulier 3SAT)
- 3SAT NAE and EO-3SAT are NP-complete

- 2SAT is polynomial
- 2SAT is NL-complete
- max2SAT is NP-complete
- there exists an efficient randomized algorithm for solving SAT