INFOF408 - Computability and Complexity
Homework

Hakim Boulahya
Université Libre de Bruxelles

December 17, 2017

1 ≠SAT is NP-complete

1.1 Proof Idea

≠SAT is a 3cnf-formula and we know that 3SAT is NP-complete. It is trivial that the best way to proof that ≠SAT is NP-complete is to find a way reduce in polynomial time 3SAT to ≠SAT. Therefore the goal is to find a way to construct an equivalent ≠SAT formula for any 3SAT formula. The ≠SAT formula must be satisfies iff the corresponding 3SAT formula is satistied.

We will proof that for any ≠assignment that satisfy the ≠SAT formula, the negation of this assignment also satisfy the formula. Therefore we have to find an ≠assignment that satifsfy the newly constructed ≠SAT formula.

1.2 Notations

Let ϕ be a 3cnf formula:

$$\phi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge ... \wedge (a_{k-1} \vee b_{k-1} \vee c_{k-1}) \wedge (a_k \vee b_k \vee c_k)$$

1.3 Negation of any ≠assignment is also a ≠assignment

If ϕ has a ≠assignment, then it means that there exists an assignment such that ϕ is satified and there is at least one literal for each clause that is set to false, and not all literals of this clause are set false. By negating the assignment, the literal set to false will change to true, which will still satisfy the clause.

1.4 Proof

Let ϕ represents any 3SAT formula. It is satisfied if all clauses are satified. To construct a ≠SAT formula, we will create, for each initial clause $(a_i \vee b_i \vee c_i)$, two clauses and add a new literal x_i such that:

$$A_i = (a_i \vee b_i \vee x_i)$$
$$B_i = (c_i \vee \neg x_i \vee \perp)$$

For each 3SAT formula, we have a ≠SAT formula:

$$\phi_{\neq} = A_1 \wedge B_1 \wedge A_2 \wedge B_2 \wedge ... \wedge A_{k-1} \wedge B_{k-1} \wedge A_k \wedge B_k$$

To make sure that $\phi_{\neq}$ is a $\neq$SAT formula it is necessary that: (1) each constructed clauses must be satisfied only if the initial clause is satisfied (2) the new clauses are satifisied with a $\neq$assignment.

The construction is made as follow: if the assignment that satisfy the initial clause is when $c_i = \top$ and either $a_i = \top$, $b_i = \top$ or $a_i = \bot$ and $b_i = \bot$, *i.e.* a_i and b_i must not be true together when c_i is true, then set $x_i = \top$. For all other assignments set $x_i = \bot$. The construction is shown in Fig. 1.

a_i	b_i	c_i	x_i
1	0	1	1
0	1	1	1
0	0	1	1
1	1	1	0
1	1	0	0
1	0	0	0
0	1	0	0
0	0	0	0

Figure 1: Construction of x_i based on the initial clause

First let's prove that the construction in Fig. 1 satisfy the new clauses only when the initial clause is satisfy. The initial clause is not sat when all literals are set to false: x_i is set to false in that case, therefore A_i is not sat *i.e.* $\phi_{\neq}$ will not be sat. When the initial clause is sat, then the new clauses are both sat: if the assignment that satisfy the initial clause is based on a_i or b_i (*i.e.* they are true), A_i is true, and B_i is true because of the negation of x_i. And if the sat assignment is only based on c_i, then x_i is set to true, which sat B_i and A_i.

Second we have to prove that that the construction in Fig. 1 is a $\neq$assignment for $A_i \wedge B_i$. It is obvious that B_i is satisfied with a $\neq$assignment because of the constant literal $\bot$. We can see that any assignment in Fig. 1 that satisfy A_i, still satisfy A_i when using their negation (see 1.3).

This construction is obviously polynomial because we add a new clause per initial clause from ϕ, and we only assign values the the x literal based on constant complexity conditions.

We proved that $3SAT \leq_P \neq SAT$ *i.e.* $\neq$SAT is NP-complete.