Formal Methods
Fall 2017

S08

Professor : Ultes-Nitsche Ulrich
Assistant : Christophe Stammet

Submitted by Sylvain Julmy

Exercise 1

In order to show that 4TA-SAT is $\mathcal{NP}$-complete , we first have to show that 4TA-SAT is in $\mathcal{NP}$, then we proof that 4TA-SAT is $\mathcal{NP}$-complete by reducing it in polynomial time to the known $\mathcal{NP}$-complete problem SAT.

4TA-SAT is in $\mathcal{NP}$

We can construct a non-deterministic Turing Machine that solve the 4TA-SAT problem in polynomial time. The TM can check all the truth assignments to the propositional formula in parallel and find 4 evaluation of the formula in polynomial time.

4TA-SAT is $\mathcal{NP}$-complete

In order to use an 4TA-SAT algorithm to solve SAT, we are going to transform a formula *phi* which is in CNF and the input of the SAT algorithm. 4TA-SAT is *true* if and only if there are 4 different interpretations which satisfies ϕ.
We use the following transformation, where *phi′* is the transformed formula *phi* for the 4TA-SAT algorithm :

$$\phi' = \phi \lor (x \land y)$$

Were x and y are two new variables that does not occur in ϕ.
Now we consider the following case :

- ϕ is not satisfiable, then the 4TA-SAT algorithm would find only 1 case where ϕ' is satisfiable, where $x = true$ and $y = true$, then there would be only 1 satisfiable formula and the 4TA-SAT would return $false$
- only one interpretation satisfies ϕ, then the 4TA-SAT would find 4 different ways of satisfying ϕ. I is the interpretation that satisfies ϕ, then ϕ' would be satisfied by the four following interpretation :
  - $I' = I \cup (\{x \mapsto false, y \mapsto false\})$
  - $I' = I \cup (\{x \mapsto false, y \mapsto true\})$
  - $I' = I \cup (\{x \mapsto true, y \mapsto false\})$
  - $I' = I \cup (\{x \mapsto true, y \mapsto true\})$

The reduction of ϕ to ϕ' is in polynomial time because we just add two variables.

Exercise 2

In order to show that NAE-4-SAT is $\mathcal{NP}$-complete , we first show that NAE-4-SAT is in $\mathcal{NP}$, then we proof that the 3-SAT is reducible to NAE-4-SAT in a polynomial time. Because 3-SAT is $\mathcal{NP}$-complete , therefore NAE-4-SAT would be $\mathcal{NP}$-complete too.

NAE-4-SAT is in $\mathcal{NP}$

NAE-4-SAT is in $\mathcal{NP}$ since we can every clause in polynomial time with a non-deterministic Turing Machine, we check if the interpretation is satisfiable (polynomial time) and if the interpretation does not contain a clause where all of her literals are *true* (polynomial time).

NAE-4-SAT is $\mathcal{NP}$-complete

In order to reduce 3-SAT to NAE-4-SAT , we transform a 3-SAT formula ϕ into an equivalent ones which is accepted by the NAE-4-SAT problem.
For each clause $(\alpha_i \vee \alpha_j \vee \alpha_k)$ of ϕ, we represent it by a new NAE-4-SAT clause $(\beta_i, \beta_j, \beta_k, \lambda)$ which is the input of the NAE-4-SAT algorithm.
Each α_i variable of ϕ is represented by a variable β_i in the transformed formula ϕ'. α_i would be *true* if $\beta_i \neq \lambda$ and false otherwise. The clause $(\beta_i, \beta_j, \beta_k, \lambda)$ is satisfied only if one of the β is different from λ, so if we found a satisfying interpretation of ϕ', the λ variable would make it correct w.r.t the NAE-4-SAT constraint.
Then, ϕ is satisfiable if and only if phi' is satisfiable, because λ is shared with all the clauses of NAE-4-SAT .
The transformation is polynomial because we just use the same variable of each clause and add a new one (always the same).