Computationally Hard Problems

Student name and id: Anders H. Opstrup (s160148)

Collaborator name(s) and id(s):

Hand-in for week: 2

Exercise 1

Given is a disjunctive form consisting of k monomials $m_1, ..., m_k$ over n boolean variables $x_1, ..., x_n$. The task is to decide if there is a truth assignment to the variables such that the truth value of the disjunctive form is false.

You are given a decision algorithm A_d that solves this problem, i. e., for each instance to REFUTATION, A_d answers YES if there is an assignment that makes the truth value of the disjunctive form false; otherwise it answers NO.

a)

Describe an algorithm A_o which solves the optimization problem, that is, which finds a truth assignment making the disjunctive form false if one exists. The running time has to be polynomial in the input size and A_o may make calls to A_d. Such calls count as one basic computational step.

Phase 1)

Ask A_d if there is an assignment that makes the truth value of the disjunctive form false. If A_d answers YES we proceed.

Phase 2)

When we know there is at least one assignment that makes the truth value of the disjunctive form false, we can proceed with:

1. Step 1) We remove one of the monomials m_k
2. Step 2) We ask the decision algorithm A_d again, if the answer is NO = we have found the assignment; else we repeat step 1.

b)

Argue that your algorithm is correct.

The algorithm is correct for the following reason: As long as the decision algorithm returns YES, there is one assignment that makes the truth value of the disjunctive form false. If the decision algorithm answers NO, it must have answered YES in the previous call, and the monomial removed must be the one we are looking for. If the disjunctive form contains more than one monomial which meets our requirements, the decision algorithm will simply answer YES if we remove the monomial, which is fine because we are just looking for given solution. If we want to find all the solutions we could run the algorithm again after removing the last monomial.

c)

Prove that the running time of the algorithm is bounded from above by a polynomial. Any polynomial is sufficient; you need not look for a polynomial of minimal degree. Recall that a call to A_d counts as one step.

Given A_d counts as one step:

The algorithm makes first one call to A_d and calls A_d one time in each loop n times. There for the running time of the algorithm is $1 + n(n-1) \rightarrow O(n^2)$.

Note: The input of REFUTATION is disjunctive form of monomials over n boolean variables, nothing else. In particular, a legal input cannot specify specific settings of variables.