INF2080
Oblig 3

Rune Hovde, runehovd

Deadline: Friday April 20th 23:59

Hand-in and deadline

Hand in a single PDF file with your answers. You can scan written answers and compile the scans into a PDF file, but make sure the pages are correctly oriented, and that they are readable. Your answer may be in English or a Norwegian-like language.

Problem 1

At least one of the above languages is *NP*-complete. Identify them, and prove that they are in P.

DNFSAT and DNFUNSAT

The following algorithm runs in polynomial time, and is a decider for DNFSAT:

1. Check if input is on the form DNF. If no, reject.
2. Check for SAT. If a clause has both X and $\bar{X}$ (a literal and it's negation), this clause cannot be assigned to true. Go to next clause
3. If there are no more clauses, reject. If there are any clauses witch does not contain X and $\bar{X}$, accept.

For DNFUNSAT, the algorithm will be identical, except for step 3. On step 3 the reject and accept will be flipped (accept where reject and reject where accept in DNFSAT).

CNFTAUT

This algorithm is a decider for CNFTAUT:

1. Check if input is on the form CNF. If no, reject.
2. Check CNFTAUT. Check if a clause contains both X and $\bar{X}$. If no, reject. If yes, go to next clause.
3. If there are no more clauses, accept.

Runtime

I think step one will use linear time. The check for this is trivial.
Step 2 will use $x * ((y(y+1))/2)$ time, where x is the number of clauses and y is the number of literals.
Step 3 will use constant time.

These algorithms use polynomial time, therefore DNFSAT, CNFUNSAT and DNFTAUT is in P.

Problem 2:

At least one of the above languages is NP-complete. Identify them, and prove that they are *NP*-complete.

CNFSAT

The following algorithm runs in polynomial time and is a decider for CNFSAT with a certificate.
The certificate contains variables set to 0/1 that makes the CNFSAT true.

1. Check that the input is on the form CNF. If no, reject.
2. For all clauses, check that at least one literal in the clause is true(1). If no, reject.
3. If there are no more clauses, accept.

Runtime

Step 1 will take linear time.
Step 2 will take linear time, since all the algorithm has to do is look up what a variable is assigned to for each literal.
Step 3 will take constant time.

This algorithm runs in polynomial time with a certificate. CNFSAT is therefore in NP. In one if the lectures we have shown that we can reduce CNFSAT to 3SAT, that is NP-complete.
This is done by slitting clauses and add literals until the clauses contain three literals.
Ref. lecture, slide 16.

Problem 3:

At least one of the above languages is *coNP*-complete. Identify them, and prove that they are *coNP*-complete.

CNFUNSAT

The following algorithm runs in polynomial time and is a decider for CNFUNSAT with a certificate. The certificate is on the same form as above.

1. Check if the input is on the form CNF. If no, reject.
2. Check that at least one of the clauses evaluates to false if yes, accept, if no, reject.

This takes as long as the decider for CNFSAT. Polynomial time and checks no-instances. Since we have shown that CNFSAT is *NP*-complete, will CNFINSAT be in *coNP*-complete.

DNFTAUT

If we can reduce DNFTAUT to CNFUNSAT, DNFTAUT will also be *coNP*-complete.
The reduction can be done by taking the nagation of DNFTAUT, then apply De Morgans laws to get CNFUNSAT.
Negation of DNFTAYT takes linear time. De Morgans laws will also take linear time.

Therefore, DNFTAUT is *coNP*-complete.