
Logic in Computer Science, November 2nd, 2022. Time: 1h30min. No books or lecture notes allowed.

-Insert your answers on the dotted lines ... below, and only there.
-Do NOT modify the problems or the @nota lines.
-When finished, upload this file with the same name: exam.txt
-Use the text symbols: & v - -> |= A E
   for AND OR NOT IMPLIES "SATISFIES" FORALL EXISTS etc.,
like in:
   I |= p & (q v -r) (the interpretation I satisfies the formula p & (q v -r)).
   You can write subindices using "_". For example write x_i to denote x-sub-i.

Problem 1. (3 points). @n@nota1:

1a) Given two propositional formulas F and G, is it true that F -> G is tautology and
    F satisfiable, then G is satisfiable?
    Prove it using only the formal definitions of propositional logic.

Answer:

    It is true:

    F satisfiable implies
    E I s.t. I |= F implies [definition of satisfiability]
    E I s.t. I |= F and I |= F -> G implies [F -> G is tautology, definition of tautology]
    E I s.t. I |= F and I |= -F v G implies [definition of ->]
    E I s.t. eval_I(F) = 1 and eval_I(-F v G) = 1 implies [definition of satisfaction]
    E I s.t. eval_I(F) = 1 and max(1 - eval_I(F), eval_I(G)) = 1 implies [definition of evaluation]
    E I s.t. max(1 - 1, eval_I(G)) = 1 implies [simplification]
    E I s.t. max(0, eval_I(G)) = 1 implies [simplification]
    E I s.t. eval_I(G) = 1 implies [definition of satisfiability]
    G satisfiable

1b) Give an example of formulas F1, F2, and F3 such that F1 & F2 & F3 is unsatisfiable
    and any conjuction of two of then is satisfiable.

Answer:
    For example, consider P={p,q} and the formulas:
      F1 = p v q
      F2 = -p
      F3 = -q
    The interpretation I(p)=0 I(q)=1 satisfies F1 & F2
    The interpretation I(p)=1 I(q)=0 satisfies F1 & F3
    The interpretation I(p)=0 I(q)=0 satisfies F2 & F3
    But F1 & F2 & F3 is unsatisfiable

Problem 2. (2.5 points). @n@nota2:

We define the problem NEG-SAT as follows:
given a propositional formula F, to determine whether there exists I such that I |= -F.

a) Describe a linear-time algorithm for NEG-SAT when the input formula
   is in CNF. Justify its correctness and its cost.

   Hint: you can assume that, given a clause C, detecting if

C contains contradictory literals, i.e., p and -p for
some variable p, can be done in linear time.

Answer:

The algorithm works as follows. Let F be the input CNF. If F is the empty set of clauses, then F is a tautology, and hence we return NO. Otherwise we take a clause C of F. If there is a pair of contradictory literals in C, then C is a tautology that can be discarded, and the algorithm starts again with the resulting formula. Otherwise we can build an interpretation I that falsifies C by setting to false all the literals in C. This interpretation also falsifies F, and so we return YES.

The algorithm is correct because when we return YES, we have indeed found an interpretation that falsifies F. For the reverse implication, if there exists I such that I |= -F, then there exists C ∈ F such that I |= -C. The algorithm will return YES when this clause is processed.

Since detecting contradictory literals can be done in linear time and we just need to make one pass over the formula, the algorithm is linear.

b) Let us call CNF-NEG-SAT the linear-time algorithm of the previous exercise for NEG-SAT when the input formula is in CNF:

```
Algorithm CNF-NEG-SAT
---------------------

Input: propositional formula F in CNF

Output:
  YES if there exists I such that I |= -F,
  NO otherwise
```

Consider now the following algorithm for solving the SAT problem for arbitrary formulas:

```
Algorithm MY-SAT
----------------

Input: propositional formula F

Output:
  YES if there exists I such that I |= F,
  NO otherwise

Step 1. G := Tseitin_transformation_of(-F)
Step 2. return CNF-NEG-SAT(G)
```

The algorithm MY-SAT is NOT correct. Prove it giving a counterexample.

Answer:

First of all notice that the Tseitin transformation G of any formula is of the form p & G', where G' is some CNF and p is the auxiliary propositional variable representing the root of the formula tree. So for any interpretation I such that I(p) = 0 we will have not(I |= G), i.e., I |= -G. So in algorithm MY-SAT(F), the call CNF-NEG-SAT(G) always returns YES, independently of F. In particular, even if F is unsatisfiable (for example, q & -q), MY-SAT(F) will return YES.

Problem 3. (2.5 points). @n@nota3:

3) Given S a set of clauses (CNF) over n propositional symbols, and Resolution the deductive rule:

```
                p v C          -p v D
                ---------------------      for some symbol p
                        C v D
```

3a) Given n propositional symbols, how many different clauses are there (seen as sets of literals)?

Answer:
    2^(2^n) = 4^n different clauses

3c) Is Resolution a correct deductive rule: (p v C) & (-p v D) |= C v D for any p, C, D?
    Prove it.

Answer:
    Let I some model of (p v C) & (-p v D). There are two cases:
      - I(p)=1 Then: I |= (p v C) & (-p v D) ==> I |= -p v D ==> I |= D ==> I |= C v D
      - I(p)=0 Then: I |= (p v C) & (-p v D) ==> I |= p v C ==> I |= C ==> I |= C v D.

3c) Can Resolution be used to decide SAT? Briefly explain why o why not?

Answer:
    Res(S), the closure under Resolution of an initial set of clauses S,
    can be computed in a finite time because there are a finite number
    of different clauses.
    Resolution is correct and refutationally complete, so if the empty clause
    belongs to Res(S) we can say that S is unsatisfiable. Otherwise S is
    satisfiable.

Problem 4. (2 points).   @n@nota4:

4) Consider the cardinality constraint x1 + x2 + x3 + x4 + x5 >= 2 (expressing that at
   least 2 of the propositional symbols {x1, x2, x3, x4, x5, x6} are true).

4a) Write the clauses needed to encode this constraint using no auxiliary variables.

Answer:
   These 5 clauses:
```
   x1 v x2 v x3 v x4
   x1 v x2 v x3      v x5
   x1 v x2      v x4 v x5
   x1      v x3 v x4 v x5
        x2 v x3 v x4 v x5
```

4b) In general, in terms of n and k, how many clauses are needed to encode a cardinality
    constraint x1 + . . . + xn >= k using no auxiliary variables? (give no explanations
here).

Answer:
   One clause for each way to choose (n-k+1) literals from a set of n literals:
   Binomial n choose (n-k+1)   [n! / ((n-k+1)!*(k-1)!) clauses]

4c) Write at least two names of any other encoding you know for cardinality constraints,
    encodings that do use auxiliary variables.

Answer:
   Ladder encoding, Sorting networks, Cardinality networks