Lógica en la Informática / Logic in Computer Science

Friday April 16th, 2021

Time: 1h20min. No books, lecture notes or formula sheets allowed.

1) (3 points) Prove your answers using only the formal definitions of propositional logic.
1a) Is it true that if F, G, H are formulas such that $F \land G \not\models H$ then $F \land G \land H$ is unsatisfiable?

Answer: This is false. Counterexample: $F = p$, $G = p$, $H = q$.
Then $p \land p \not\models q$: if $I(p) = 1$ and $I(q) = 0$ then $I \models p \land p$ but $I \not\models q$.
But $p \land p \land q$ is satisfiable: if $I(p) = 1$ and $I(q) = 1$ then $I \models p \land p \land q$.

1b) Let F be a tautology, and let G an unsatisfiable formula. Is it true true that $F \land \neg G$ is a tautology?

Answer: F tautology and G unsatisfiable implies (by def. of tautology and unsatisfiable)
$\forall I$, $I \models F$ and $I \not\models G$ implies (by def. of $\models$)
$\forall I$, $eval_I(F) = 1$ and $eval_I(G) = 0$ implies (by arithmetic)
$\forall I$, $eval_I(F) = 1$ and $1 - eval_I(G) = 1$ implies (by def. evaluation of $\neg$)
$\forall I$, $eval_I(F) = 1$ and $eval_I(\neg G) = 1$ implies (by arithmetic)
$\forall I$, $min(eval_I(F), eval_I(\neg G)) = 1$ implies (by def. evaluation of $\land$)
$\forall I$, $eval_I(F \land \neg G) = 1$ implies (by def. of $\models$)
$\forall I$, $I \models F \land \neg G$ implies (by def. of tautology)
$F \land \neg G$ is a tautology.

2) (2 points) The problem called "minOnes" takes as input a natural number k and a propositional formula F over propositional variables $\{x_1 \ldots, x_n\}$. Its aim is to decide if there is any model I of F with at most k ones, that is, any model I such that $I(x_1) + \ldots + I(x_n) \leq k$.
Answer in a few words: Is minOnes NP-hard? Why?

Answer: Yes. We can polynomially reduce SAT to minOnes (solve SAT using minOnes): to decide SAT for F (over n propositional symbols), call minOnes with input $k = n$ and F. So, since SAT is NP-Hard (that is, any problem in NP can be polynomially reduced to SAT) minOnes is NP-Hard too.

3) (2 points) Every propositional formula F over n variables can also expressed by a Boolean circuit with n inputs and one output. In fact, sometimes the circuit can be much smaller than F because each subformula only needs to be represented once. For example, if F is

$$x_1 \land (x_3 \land x_4 \lor x_3 \land x_4) \quad \lor \quad x_2 \land (x_3 \land x_4 \lor x_3 \land x_4),$$

a circuit C for F with only five gates exists. Giving names a_i to the output wires of each logical gate, and using a_0 as the output of C, we can write C as:

```
a0 = or(a1,a2)    a1 = and(x1,a3)      a3 = or(a4,a4)
                  a2 = and(x2,a3)      a4 = and(x3,x4)
```

Explain **very briefly** what do you think is the best way to use a standard SAT solver for CNFs to determine whether two circuits C_1 and C_2, represented like this, are logically equivalent.
Note: assume different names $b_0, b_1, b_2 \ldots$ are used for the internal wires of C_2.

Answer: Apply Tseitin. Each gate already has its auxiliary variable a_i. Each gate $a_i = and(x, y)$, generates three clauses: $\neg a_i \lor x$, $\neg a_i \lor y$, and $a_i \lor \neg x \lor \neg y$, and each gate $a_i = or(x, y)$ another three: $a_i \lor \neg x$, $a_i \lor \neg y$, and $\neg a_i \lor x \lor y$. Negations can also be handled as usual.
If S_1 and S_2 are the resulting clause sets for the gates of C_1 and C_2, respectively, then:
$C_1 \equiv C_2$ (both circuits have the same models) iff
there is no model of $S_1 \cup S_2$ such that the root variables a_0 and b_0 get different values iff
on input $S_1 \cup S_2 \cup \{ \neg a_0 \lor \neg b_0, \ \ a_0 \lor b_0 \}$, the SAT solver returns unsatisfiable.

(Note: if we first transform the circuits (directed acyclic graphs) into formulas (trees) and then apply Tseitin, the CNF can become much larger, due to multiple copies of sub-circuits.)

4) (3 points) Consider the cardinality constraint $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \leq 4$ (expressing that at most 4 of the propositional symbols $\{x_1, x_2, x_3, x_4, x_5, x_6\}$ are true).
4a) Write the clauses needed to encode this constraint using no auxiliary variables.

Answer:
$\neg x_1 \lor \neg x_2 \lor \neg x_3 \lor \neg x_4 \lor \neg x_5$ $\qquad$ $\neg x_1 \lor \neg x_2 \lor \neg x_3 \lor \neg x_4 \lor \neg x_6$ $\qquad$ $\neg x_1 \lor \neg x_2 \lor \neg x_3 \lor \neg x_5 \lor \neg x_6$
$\neg x_1 \lor \neg x_2 \lor \neg x_4 \lor \neg x_5 \lor \neg x_6$ $\qquad$ $\neg x_1 \lor \neg x_3 \lor \neg x_4 \lor \neg x_5 \lor \neg x_6$ $\qquad$ $\neg x_2 \lor \neg x_3 \lor \neg x_4 \lor \neg x_5 \lor \neg x_6$

4b) In general, in terms of n and k, how many clauses are needed to encode a cardinality constraint $x_1 + \ldots + x_n \leq k$ using no auxiliary variables? (give no explanations here).

Answer: $\binom{n}{k+1}$

4c) Write the names of any other encoding you know for cardinality constraints $x_1 + \ldots + x_n \leq k$, an encoding that do use auxiliary variables. In terms of n and k, how many clauses are needed? (give no explanations here).

Answer: sorting networks, $O(n\ log^2 n)$ or cardinality networks, $O(n\ log^2 k)$