Lógica en la Informática / Logic in Computer Science

Friday November 24, 2017

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

1) Below F, G, H denote arbitrary propositional formulas. Mark with an `X` the boxes of the true statements (give no explanations).

1. If $F \land G \not\models H$ then $F \land G \land H$ is unsatisfiable. ☐
2. If F es a tautology, then for every G we have $G \models F$. ☐
3. If F is unsatisfiable then $\neg F$ is a tautology. ☐
4. If $F \land G \models \neg H$ then $F \land G \land H$ is unsatisfiable. ☐
5. If $F \lor G \models H$ then $F \land \neg H$ is unsatisfiable. ☐
6. The formula $p \lor p$ is a logical consequence of $(p \lor q \lor r) \land (\neg q \lor r) \land (\neg r)$. ☐
7. If F is unsatisfiable, then for every G we have $G \models F$. ☐
8. It can happen that $F \models G$ and $F \models \neg G$. ☐
9. The formula $(p \lor q) \land (\neg p \lor q) \land (\neg p \lor \neg q) \land (\neg q \lor p)$ is unsatisfiable. ☐
10. If F is a tautology, then for every G we have $F \models G$. ☐
11. If F is unsatisfiable then $\neg F \models F$. ☐
12. F is satisfiable if, and only if, all logical consequences of F are satisfiable formulas. ☐

2) Let C_1 and C_2 be propositional clauses, and let D be the conclusion by resolution of C_1 and C_2.

2a) Is D a logical consequence of $C_1 \land C_2$? Prove it formally, using only the definitions of propositional logic.

2b) Let S be a set of propositional clauses and let $Res(S)$ be its closure under resolution. Is it true that $S \equiv Res(S)$? Very briefly explain why.

3) 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. Representing the output of each logical gate as a new auxiliary variable a_i 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** how you would use a standard SAT solver for CNFs to **efficiently** 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 auxiliary variables of C_2.