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. ☐ **False**
2. If F es a tautology, then for every G we have $G \models F$. ☐ **True**
3. If F is unsatisfiable then $\neg F$ is a tautology. ☐ **True**
4. If $F \land G \models \neg H$ then $F \land G \land H$ is unsatisfiable. ☐ **True**
5. If $F \lor G \models H$ then $F \land \neg H$ is unsatisfiable. ☐ **True**
6. The formula $p \lor p$ is a logical consequence of $(p \lor q \lor r) \land (\neg q \lor r) \land (\neg r)$. ☐ **True**
7. If F is unsatisfiable, then for every G we have $G \models F$. ☐ **False**
8. It can happen that $F \models G$ and $F \models \neg G$. ☐ **True**
9. The formula $(p \lor q) \land (\neg p \lor q) \land (\neg p \lor \neg q) \land (\neg q \lor p)$ is unsatisfiable. ☐ **True**
10. If F is a tautology, then for every G we have $F \models G$. ☐ **False**
11. If F is unsatisfiable then $\neg F \models F$. ☐ **False**
12. F is satisfiable if, and only if, all logical consequences of F are satisfiable formulas. ☐ **True**

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.

Answer:
[remember: *Resolution* is a deduction rule where from two clauses of the form $p \lor C$ and $\neg p \lor D$ (the *premises*), the new clause $C \lor D$ (the *conclusion*) is obtained. Here p is a predicate symbol, and C and D are (possibly empty) clauses. The *closure under resolution* $Res(S)$ contains all clauses that can be obtained from S by zero or more resolution steps; formally, it is the union, for i in $0..\infty$, of all S_i where $S_0 = S$ and $S_{i+1} = S_i \cup Res_1(S_i)$, where $Res_1(S_i)$ is the set of clauses that can be obtained by one step of resolution with premises in S_i.]

It is true that $(p \lor C_1') \land (\neg p \lor C_2') \models C_1' \lor C_2'$. By definition of logical consequence, we have to prove that for all I, if $I \models (p \lor C_1') \land (\neg p \lor C_2')$ then $I \models C_1' \lor C_2'$.

We prove it by case analysis. Take an arbitary I. Assume $I \models (p \lor C_1') \land (\neg p \lor C_2')$.

Case A): $I(p) = 1$.

$I \models (p \lor C_1') \land (\neg p \lor C_2')$ implies,	by definition of satisfaction, that
$eval_I((p \lor C_1') \land (\neg p \lor C_2')) = 1$ which implies,	by definition of evaluation of $\land$, that
$min(eval_I(p \lor C_1'), eval_I(\neg p \lor C_2')) = 1$ which implies,	by definition of min, that
$eval_I(\neg p \lor C_2') = 1$ which implies,	by definition of evaluation of $\lor$, that
$max(eval_I(\neg p), eval_I(C_2')) = 1$ which implies,	by definition of evaluation of $\neg$, that
$max(1 - eval_I(p), eval_I(C_2')) = 1$ which implies,	by definition of $eval_I(p)$, that
$max(1 - I(p), eval_I(C_2')) = 1$ which implies,	since $I(p) = 1$, that
$max(0, eval_I(C_2')) = 1$ which implies	
$eval_I(C_2') = 1$ which implies	

$max(eval_I(C'_1), eval_I(C'_2)) = 1$ which implies, by definition of evaluation of $\vee$, that
$eval_I(C'_1 \vee C'_2) = 1$ which implies, by definition of satisfiction, that
$I \models C'_1 \vee C'_2$.

Case B): $I(p) = 0$.

The proof is analogous to Case A, with the difference that now from $min(eval_I(p \vee C'_1), eval_I(\neg p \vee C'_2)) = 1$ we obtain $eval_I(p \vee C'_1) = 1$ and hence (since $I(p) = 0$) $eval_I(C'_1) = 1$ which implies $eval_I(C'_1 \vee C'_2) = 1$ and hence $I \models C'_1 \vee C'_2$.

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.

Answer: Yes. We have $Res(S) \models S$ (all models of $Res(S)$ are models of S) because $Res(S) \supseteq S$. We also have $S \models Res(S)$. Let I be a model of S. $Res(S)$ is obtained from S by finitely many times adding to the set a new clause that (as we have seen in 2a) is a logical consequence of two clauses we already have. So each time we add a new clause $C \vee D$ to a set of the form $Set \cup \{\ p \vee C \quad \neg p \vee D\}$, we will have $I \models Set \cup \{\ p \vee C \quad \neg p \vee D\}$ and then also $I \models Set \cup \{\ p \vee C \quad \neg p \vee D \quad C \vee D\}$.

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

$$x_1 \wedge (x_3 \wedge x_4 \ \vee \ x_3 \wedge x_4) \quad \vee \quad x_2 \wedge (x_3 \wedge x_4 \ \vee \ x_3 \wedge x_4),$$

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.

Answer: We can apply the Tseitin transformation directly to each sub-circuit: each gate already has its auxiliary variable. Each gate $a_i = and(x, y)$, generates three clauses: $\neg a_i \vee x$, $\neg a_i \vee y$, and $a_i \vee \neg x \vee \neg y$, and each gate $a_i = or(x, y)$ another three: $a_i \vee \neg x$, $a_i \vee \neg y$, and $\neg a_i \vee x \vee y$. Negations can also be handled as usual.

Let S_1 and S_2 be the resulting sets of clauses for the gates of C_1 and C_2, respectively. Then we have:

$C_1 \equiv C_2$ (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 (CNF) input $S_1 \cup S_2 \cup \{\ \neg a_0 \vee \neg b_0, \quad a_0 \vee b_0\ \}$, 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.