Lógica en la Informática / Logic in Computer Science

Nov 15th, 2012.

Time: 1h30min. No books, lecture notes or formula sheets allowed.

Note: The answers will also be considered for evaluation of the transversal competences of English and Reasoning (but both with 0 % impact on the global evaluation of LI).

1) For each one of the following statements, indicate whether it is true or false, without giving any explanations why.

1. Let F, G, H be formulas. If $F \land G \models \neg H$ then $F \land G \land H$ is unsatisfiable. **True**
2. Let F, G, H be formulas. If $F \lor G \models H$ then $F \land \neg H$ is unsatisfiable. **True**
3. The formula $p \lor p$ is a logical consequence of the formula $(p \lor q \lor r) \land (\neg q \lor r) \land (\neg r)$. **True**
4. The formula $(p \lor q) \land (\neg p \lor q) \land (\neg p \lor \neg q) \land (\neg q \lor p)$ is unsatisfiable. **True**
5. Assume $|\mathcal{P}| = n$. There are 2^n interpretations. Moreover there are exactly $k = 2^{2^n}$ formulas $F_1, \ldots, F_k$ such that for all i, j with $i \neq j$ in $1 \ldots k$, $F_i \not\equiv F_j$. Each one of these F_i represents a different Boolean function. **True**
6. If F is unsatisfiable, then for every G we have $G \models F$. **False**
7. If F is a tautology, then for every G we have $F \models G$. **False**
8. If F es a tautology, then for every G we have $G \models F$. **True**

2A) Let F and G be formulas. Can it happen that $F \models G$ and $F \models \neg G$? Prove it.

Answer: Yes, if F is unsatisfiable. Then F has no models and by definition of $\models$ it happens for all formulas G.

2B) Let F be a formula. Is it true that F is satisfiable if, and only if, all logical consequences of F are satisfiable formulas? Prove it.

Answer: Yes, it is true. If F is satisfiable then F has some model I, and, for every G such that $F \models G$, by definition of $\models$ we have that $I \models G$, so G is satisfiable. For the reverse implication, note that by definition of $\models$ we have $F \models F$, so if all logical consequences of F are satisfiable formulas then F itself is satisfiable.

3) What is the complexity of deciding the satisfiability of an input formula in DNF? Explain why.

Answer: The complexity is linear. A formula F in DNF has the form $C_1 \lor \ldots \lor C_n$, where each C_i is a *cube*, a conjunction of literals. Then, by definition, F is satisfiable if at least one of its cubes C_i is satisfiable (because for every interpretation I we have $eval_I(F) = 1$ iff $eval_I(C_i) = 1$ for some C_i). This can be checked in linear time, since a cube $p_1 \land \ldots \land p_k \land \neg q_1 \land \ldots \land \neg q_{k'}$ is satisfiable iff it does not contain two opposed literals, that is, iff $p_i \neq q_j$, for all $i \in 1..k$ and $j \in 1..k'$.

4) After the general strike yesterday, the n workers of Iberia want to create a national comittee with k members. Each worker proposes a list of 10 names out of the list of n workers. The objective is that, for each worker, at least one of the names of his/her list is in the commitee. We want to decide whether this is possible or not.

4A Do you think that this problem is polynomial? How would you solve it using SAT?

Answer: Unless P=NP, this problem in not polynomial. Even in the particular case where each worker proposes only two names (case 4B below) it is NP-complete, because it is equivalent to the well-known NP-complete problem of vertex cover: given a graph with n vertices (the workers) and with edges $(u_1, v_1) \ldots (u_n, v_n)$, where each (u_i, v_i) is the pair of names proposed by worker i, decide whether there exists a subset of k vertices such that for each edge (u, v) at least one of u or v is in the subset. [Note: the vertex cover problem is also described in lecture notes p3, exercise 29.]

To solve the problem using SAT, we can simply have n propositional variables $x_1 \ldots x_n$, where each x_i means "worker i is in the committee". Clauses are needed to express that at most k of $x_1 \ldots x_n$ can be true, and, for each list of 10 names $k_1 \ldots k_{10}$, one clause $x_{k_1} \vee \ldots \vee x_{k_{10}}$ (to make sure that at least one of them will be in the committee).

The at-most-k-constraint can be expressed using any of the well-known encodings (without auxiliary variables, or using the ladder encoding, or using sorting networks, etc).

4B Answer the same questions if each worker proposes only 2 names.

Answer: the same as before, except that now each list of names generates a two-literal clause, instead of a 10-literal one.