Lógica en la Informática / Logic in Computer Science

Tuesday November 18th, 2014

Time: 1h345min. No books, lecture notes or formula sheets allowed.

1A) (1 pt) Let F and G be propositional formulas. Is it true that if $F \to G$ is a tautology and F is satisfiable, then G is a tautology? Prove it using only the definition of propositional logic.

Solution: This is false. Counterexample: If $F = G = p$, then $p \to p$, that is, $\neg p \vee p$, is a tautology. Morevover, p is satisfiable. But p is not a tautology.

1B) (1 pt) Let F and G be propositional formulas. Is it true that if $F \to G$ is satisfiable and F is satisfiable, then G is satisfiable? Prove it using only the definition of propositional logic.

Solution: This is false. Counterexample: $F = p$ and $G = p \wedge \neg p$. Then $F \to G$, which is $\neg F \vee G$, which is $\neg p \vee (p \wedge \neg p)$ is satisfiable: if we define I such that $I(p) = 0$, then $I \models \neg p$ and hence $I \models \neg p \vee (p \wedge \neg p)$. Also F is satisfiable: if we define I such that $I(p) = 1$, then $I \models p$. But G is not satisfiable: there is no I such that $I \models p \wedge \neg p$.

2) (2 pts) Let S be a set of propositional clauses and let C be a propositional clause $l_1 \vee \ldots \vee l_n$. We want to use a SAT solver in order to know whether $S \models C$. One Erasmus student says: "That's easy! It is true if the SAT solver finds a model for $S \cup \{\neg l_1, \ldots, \neg l_n\}$". Is he right? Explain in detail why.

Solution: He is not right. We have $S \models C$ iff $S \cup \{\neg C\}$ is **un**satisfiable. Since $\neg C$ is $\neg l_1 \wedge \ldots \wedge \neg l_n$, we have $S \models C$ iff $S \cup \{\neg l_1, \ldots, \neg l_n\}$ is **un**satisfiable, i.e., if the SAT solver finds no model for $S \cup \{\neg l_1, \ldots, \neg l_n\}$, i.e., if the SAT solver returns "unsatisfiable".

3) (2.5 pts) The engineers at Intel have two very large digital circuits C_1 and C_2, built with binary **and** and **or** gates and unary **not** gates (inverters). Each one of them has n input wires `w1...wn` and 1 output wire `w-out`, and they want to know whether the two circuits are equivalent, that is, whether they give the same output bit for every combination of n input bits. Each circuit is given in a simple format with one line per gate and inner wires represented by numbers, for example:

```
w-out=and(w1,1)
    1=or(2,3)
    2=and(w2,3)
    3=and(4,w4)
    4=not(w3)
```

Note that circuits are not always trees (as formulas are). They can be directed acyclic graphs because subcircuits can be shared, as it happens with wire `3` in the example. Explain in the simplest possible way what you would do for using the picosat SAT solver to determine whether C_1 and C_2 are equivalent.

Solution: Assume F_1 and F_2 are propositional formulas. Then we know that $F_1 \equiv F_2$ iff $(F_1 \wedge \neg F_2) \vee (\neg F_1 \wedge F_2)$ is unsatisfiable. Let F be the formula $(F_1 \wedge \neg F_2) \vee (\neg F_1 \wedge F_2)$. Then we can use the Tseitin transformation to transform F into another formula in CNF (a set of clauses) S, such that F and S are equisatisfiable (that is, F is satisfiable iff S is satisfiable). So $F_1 \equiv F_2$ iff picosat returns "unsatisfiable" when given S as input. [Explain a bit further the Tseitin transformation: it introduces a new auxiliary variable a for every logical connective. If the connective is an $\wedge$, and the children have variables b and c, then the three clauses $\{\neg a \vee b,\ \neg a \vee c,\ a \vee \neg b \vee \neg c\}$ are added. A unit clause, stating that the variable at the root of the formula is true, is added as well. Etc.]

In this case, instead of having formulas F_1 and F_2, we have circuits C_1 and C_2 where the Tseitin transformation would introduce a new variable a_i for every internal wire i. But this is exactly the same, since re-using subformulas such as the one for wire `3` does not matter: equisatisfiability holds with a single copy of the subformula headed with a variable a_3, as well as with two copies of the subformula with two copies of the new variables a_3, a'_3, etc.

4) (3.5 pts) The Seat factory in Martorell has a single production line, on which every day N cars are produced in a certain sequence (a linear order: one car after the other). Each car needs zero or more special *features* (demanded by the buyer who has ordered the car: leather seats, sunroof, special music equipment, special paint, etc.). In total there are 40 possible features, that are installed in the car in 40 workshops located along the production line, and there are constraints saying things like: "at most 3 out of every 7 *consecutive* cars can get a sunroof". This is because these workshops only have a limited work capacity: more sunroofs close together would slow down the production line. So the information for a specific day, written in Prolog style, would look like this:

```
car(1,[3,5,8,10]).  % car 1 needs features 3,5,8,10
...
car(N,[2,6,40]).    % car N needs features 2,6,40

feature(1,5,10).    % at most 5 cars with feature 1 in each consecutive 10 cars
...
feature(40,3,7).    % at most 3 cars with feature 40 in each consecutive 7 cars
```

4A) Explain in detail how to use a SAT solver for deciding in which order we can produce the N cars. Clearly indicate which variables you use and their precise meaning, and which properties you impose using which clauses or which constraints (for cardinality or pseudo-Boolean constraints, it is not necessary to write their encodings into clauses).

Solution:
Variables:

1. $x_{i,j}$ meaning "at position i in the sequence is car j", for $1 \leq i \leq N$ and $1 \leq j \leq N$.
2. $f_{i,k}$ meaning "at position i in the sequence is a car with feature k", for $1 \leq i \leq N$ and $1 \leq k \leq 40$.

Clauses/constraints:

1. At each position i is exactly one car: N constraints of the form $x_{i,1} + \ldots + x_{i,N} = 1$ (one clause for the ALO constraint; the AMO constraint can be encoded into clauses in different ways).
2. Each car j is at exactly one position: N constraints of the form $x_{1,j} + \ldots + x_{N,j} = 1$ (ALO+AMO as before).
3. For each position i, and for each car j having feature k in its list of features, one binary clause indicating that if at position i in the sequence is car j, then at position i is a car with feature k, that is, $\neg x_{i,j} \vee f_{i,k}$.
4. For each feature k allowing at most m cars out of each consecutive c cars, cardinality constraints of the form $f_{i,k} + \ldots + f_{i+c-1,k} \leq m$ for every subsequence of positions of length c, that is, between i and $i + c - 1$ for some i with $1 \leq i < N - c + 1$.

4B) Imagine the problem becomes too difficult for the SAT solver. Explain how you would exploit the fact that there are groups of identical cars, i.e., cars needing the same subset of features.

Solution: In this case, $x_{i,j}$ can mean "at position i in the sequence is a car **of group** j". The clauses of type 1, 2 and 3 are adapted accordingly:

1. At each position i is exactly one car **group**: N constraints of the form $x_{i,1} + \ldots + x_{i,N} = 1$.
2. For each **car group** j **of** n_j **cars**, we express that it is **at exactly** n_j **positions**, with a cardinality constraint $x_{1,j} + \ldots + x_{N,j} = n_j$.
3. For each position i, and for each car **group** j having feature k in its list of features, one binary clause indicating that if at position i in the sequence is a car of group j, then at position i is a car with feature k, that is, $\neg x_{i,j} \vee f_{i,k}$.

If groups are large, this considerably reduces the number of $x_{i,j}$ variables, as well as the number of clauses of type 1, 2 and 3. Note that, given a model, we can obtain a correct car sequence by arbitrarily assigning the cars of each group j into the n_j positions available for that group.