Lógica en la Informática / Logic in Computer Science

Wednesday April 27th, 2016

Time: 1h45min. No books, lecture notes or formula sheets allowed.

1) Given two propositional formulas F and G, is it true that $F \to G$ is a tautology iff $F \models G$? Prove it using only the formal definitions of propositional logic.

Answer: This is true.

$F \to G$ is a tautology iff	(by def. of $\to$)
$\neg F \vee G$ is a tautology iff	(by def. of tautology)
for all I, $I \models \neg F \vee G$ iff	(by def. of $\models$)
for all I, $eval_I(\neg F \vee G) = 1$ iff	(by def of $eval_I(\vee)$)
for all I, $max(eval_I(\neg F), eval_I(G)) = 1$ iff	(by def of $eval_I(\neg)$)
for all I, $max(1 - eval_I(F), eval_I(G)) = 1$ iff	(by def of max)
for all I, $1 - eval_I(F) = 1$ or $eval_I(G) = 1$ iff	(by def of $-$)
for all I, $eval_I(F) = 0$ or $eval_I(G) = 1$ iff	(by def of $\models$)
for all I, $I \not\models F$ or $I \models G$ iff	(by def of logical consequence)
$F \models G$.	

2) 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 using only the formal definitions of propositional logic.

Answer: Yes, it is true.

$\Longrightarrow$) Let G be any logical consequence of F, that is, $F \models G$. Now F satisfiable means F has some model I and by definition of $F \models G$, every model of F is a model of G, which implies $I \models G$, so G is satisfiable.

$\Longleftarrow$) By definition of logical consequence we have $F \models F$ (obviously, every model of F is a model of F), so if all logical consequences of F are satisfiable formulas then F itself is also satisfiable.

3) What is Horn-SAT? What is its computational complexity? Explain very briefly why.

Answer: Horn-SAT is the problem of deciding the satisfiability of a set of clauses S such that all clauses in S are Horn: they have at most one positive literal.

It is polynomial, more precisely linear, because we can decide it simply by unit propagation of positive unit literals.

For example, given the set of four Horn clauses $S = \{ \quad \neg r, \quad p, \quad \neg p \vee q, \quad \neg p \vee \neg q \vee r \quad \}$, by propagation of the positive unit clause p, we get q, and then r, and then a conflict (the empty clause). Then S is unsat: since unit propagation is correct, all new units are logical consequences of S.

Another example: $S = \{ \quad \neg r \vee \neg r', \quad p, \quad \neg p \vee q, \quad \neg p \vee \neg q \vee r \quad \}$. We also propagate positive units p, q, r, but no empty clause appears. In that case S is sat, since we get a model I by setting all propagated units to 1 and the rest to 0: here $I(p) = I(q) = I(r) = 1$ and $I(r') = 0$. Indeed then always I is a model of S, i.e., $I \models C$, for every non-empty clause C of S that is not a positive unit clause: if C has propagated, then $I \models C$; otherwise, since C is Horn, it must have at least one negative literal $\neg p$ where p has not been propagated, so $I \models C$ too.

4) Consider the following decision problem, called "MaxSAT":
Input: A natural number k and a set S of n propositional clauses over propositional symbols $\mathcal{P}$.
Question: Is there any interpretation $I : \mathcal{P} \to \{0, 1\}$ that satisfies at least k clauses of S?

4a) Do you think that MaxSAT is polynomial? NP-complete? Exponential? Why?
Answer: It is NP-complete.
–MaxSAT is not easier than general SAT, since SAT is the particular case of MaxSAT where $k = n$.
–MaxSat is also not harder than SAT since one can reduce MaxSat to SAT: decide MaxSat with one call to a SAT solver (see 4b). Another independent proof (without using 4b): MaxSat is still in NP because one can verify a given solution, an interpretation I, in linear time (check whether indeed I satisfies at least k clauses).

4b) How would you use a SAT solver to decide it?
Answer: A single call to a SAT solver is sufficient.
Let S be $\{C_1, \ldots, C_n\}$. Use new auxiliary variables $x_1, \ldots, x_n$. Let S' be $\{C_1 \vee \neg x_1, \ldots, C_n \vee \neg x_n\}$. Then we need to find a model I for S' such that at least k of the auxiliary variables are true. Let $Card$ be the set of clauses obtained by encoding the cardinality constraint $x_1 + \ldots + x_n \geq k$. Then, running the solver with input clauses $S' \cup Card$ will find the desired I if it exists, and return "unsat" otherwise.

4c) How would you use a SAT solver to solve the optimization version of MaxSAT, that is, how to find the I that satisfies as many of the clauses of S as possible?
Answer: For this we need more than one call to the SAT solver.
A) Run the solver on the input S' defined as in 4b). Note that S' is satisfiable (just set all x_i's to 0).
B) If it finds a model where m x_i's are true, run again with input $S' \cup Card$, where $Card$ is the set of clauses obtained by encoding the cardinality constraint $x_1 + \ldots + x_n > m$.
Repeat step B (finding each time models with more true $x_i's$), until the solver returns unsat. The last solution found is the optimal one.

Another algorithm is to make calls to the solver with S' and a constraint $x_1 + \ldots + x_n \geq k$, where the first call k is n; if it is unsatisfiable, then try $n - 1$, etc., until a model is found, which is then optimal. Yet another algorithm is to do a binary search. But the first algorithm given here works very well, because it is usually easier to find a model than to prove unsatisfiability and because the improvement in solution quality at each iteration frequently is by more than 1.