Lógica en la Informática / Logic in Computer Science

Monday November 9th, 2020

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

1) (3 points)
Prove, using only the definitions of propositional logic, that the deduction rule of resolution in propositional logic is correct, that is, if from the two clauses C_1 and C_2 by resolution we can obtain a clause D, then $C_1 \wedge C_2 \models D$.

Answer: [Note: This question already appeared before; e.g., in the partial exam of Fall 2017.]
If from C_1 and C_2 by resolution we can obtain a clause D, then C_1 must be of the form $p \vee C_1'$, and C_2 must be of the form $\neg p \vee C_2'$ for some propositional symbol p, and D is $C_1' \vee C_2'$.
[Note: if you "prove" $C_1 \wedge C_2 \models D$ without using the previous fact, then you probably do not know what resolution is and you also "prove" something that is not true, so of course you can get no points.]
We now prove that indeed it is true that $(p \vee C_1') \wedge (\neg p \vee C_2') \models C_1' \vee C_2'$. By definition of logical consequence, we have to prove that for all I, if $I \models (p \vee C_1') \wedge (\neg p \vee C_2')$ then $I \models C_1' \vee C_2'$.
We prove it by case analysis. Take an arbitary I. Assume $I \models (p \vee C_1') \wedge (\neg p \vee C_2')$.
Case A): $I(p) = 1$.

$I \models (p \vee C_1') \wedge (\neg p \vee C_2')$ implies,	by definition of satisfaction, that
$eval_I((p \vee C_1') \wedge (\neg p \vee C_2')) = 1$ which implies,	by definition of evaluation of $\wedge$, that
$min(eval_I(p \vee C_1'), eval_I(\neg p \vee C_2')) = 1$ which implies,	by definition of min, that
$eval_I(\neg p \vee C_2') = 1$ which implies,	by definition of evaluation of $\vee$, 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,	by definition of max, that
$eval_I(C_2') = 1$ which implies,	by definition of $eval_I$ and max, that
$max(eval_I(C_1'), eval_I(C_2')) = 1$ 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 $I \models C_1' \vee C_2'$.

2) (3 points)
Assuming you can use a SAT solver or any other algorithm, explain very briefly what you would do and what the computational cost would be and why, to decide the following two problems:
2a) Given a formula F in disjunctive normal form (DNF), decide whether F is a tautology.
2b) Given a formula F in disjunctive normal form (DNF), decide whether F is a satisfiable.

Answer: [Note: this problem is like problem 2 of the final exam of Fall 2014, and others.]

2a: No polynomial algorithm is known. If it exists, then SAT for CNF is polynomial too!, because a CNF F is unsatisfiable iff $\neg F$ is a tautology, and by moving the negations of $\neg F$ inward (using de Morgan and doble negation), in linear time we obtain a DNF G with $G \equiv \neg F$.
[Note: This problem is Co-NP-complete (its negation is NP-complete): NOT being a tautology is in NP since it has a short certificate, an interpretation I that is NOT a model.]

2b: A DNF has the form $C_1 \vee \ldots \vee C_n$ where each C_i is a "cube", a conjunction of literals. It is satisfiable if some C_i is satisfiable, which is the case if it does not contain a literal and its negation. This can be checked in linear time.

3) (4 points)
Let $\mathcal{P}$ be a set of propositional predicate symbols. Let S be a set of clauses over $\mathcal{P}$ and let N be a subset of $\mathcal{P}$. We define $\mathit{flip}(N,S)$ to be the set of clauses obtained from S by *flipping* (changing the sign) of all literals with symbols in N.

For example, $\mathit{flip}(\ \{p,q\},\ \{\ p \vee \neg q \vee \neg r,\ \ q \vee r\ \}\ )$ is $\{\ \neg p \vee q \vee \neg r,\ \ \neg q \vee r\ \}$.
A clause is called *Horn* if it has at most one positive literal. A set of clauses S is called *renamable Horn* if there is some $N \subseteq \mathcal{P}$ such that $\mathit{flip}(N,S)$ is a set of Horn clauses.

3a) Explain in three lines: given S and N such that $\mathit{flip}(N,S)$ is a set of Horn clauses, what would you do to efficiently decide whether S is satisfiable, and why?

Answer: [Note: this problem is like, e.g., problem 2 of the final exam of Spring 2018.]
Linear: computing $\mathit{flip}(N,S)$ is linear, and deciding if it is satisifiable is linear too because it is Horn. S and $\mathit{flip}(N,S)$ are equi-satisfiable because if I is a model of one of them, then I' is a model of the other one, where $I'(p) = I(p)$ iff $p \notin N$.

3b) Given an arbitrary set of clauses S, we want to decide whether it is renamable Horn and, if so, find the corresponding N. We will do this using an algorithm based on... SAT! For each $p \in \mathcal{P}$, we introduce a SAT variable $\mathit{flipped}(p)$ meaning that symbol p is in N. Then we add clauses for every clause C of S and every pair of literals l and l' in C, forbidding that *after doing all flips, l and l' both become positive.*

Explain in three lines: which clauses do you need, what is the cost of the resulting SAT-based algorithm and why?

Answer: For every pair of literals appearing in the same clause of S:
a) if both are positive symbols p and q then we add the clause $\mathit{flipped}(p) \vee \mathit{flipped}(q)$
b) if both are negative, of the form $\neg p$ and $\neg q$ then we add $\neg\mathit{flipped}(p) \vee \neg\mathit{flipped}(q)$
c) otherwise they are of the form $\neg p$ and q, and we add $\neg\mathit{flipped}(p) \vee \mathit{flipped}(q)$
This gives a quadratic number of 2-SAT clauses, so using the linear 2-SAT algoritm we get a quadratic algorithm for deciding whether S is renamable Horn and, if so, finding the corresponding N.