Lógica en la Informática / Logic in Computer Science

Tuesday April 30th, 2019

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

1) (4 points)
1a) Let F, G, H be formulas. Is it true that if $F \lor G \models H$ then $F \land \neg H$ is unsatisfiable? Prove it using only the definition of propositional logic.

Answer: This is true. $F \lor G \models H$ implies (by def. of logical consequence) that
for all I, if $I \models F \lor G$ then $I \models H$, which implies (by def. of $\models$) that
for all I, if $eval_I(F \lor G) = 1$ then $eval_I(H) = 1$, which implies (by def of $eval_I(\lor)$) that
for all I, if $max(eval_I(F), eval_I(G)) = 1$ then $eval_I(H) = 1$, which implies (by def of max) that
for all I, if $eval_I(F) = 1$ then $eval_I(H) = 1$, which implies (by arithmetic) that
for all I, if $eval_I(F) = 1$ then $1 - eval_I(H) = 0$, which implies (by def $eval_I(\neg)$) that
for all I, if $eval_I(F) = 1$ then $eval_I(\neg H) = 0$, which implies (by def. of min) that
for all I, $min(eval_I(F), eval_I(\neg H)) = 0$, which implies (by def $eval_I(\land)$) that
for all I, $eval_I(F \land \neg H) = 0$, which implies (by def of $\models$) that
for all I, $I \not\models F \land \neg H$, which implies (by def of unsatisfiable) that $F \land \neg H$ is unsatisfiable.

1B) 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.

Answer: This is false. Counterexample: $F = p$ and $G = p \land \neg p$. Then $F \to G$, which is $\neg F \lor G$, which is $\neg p \lor (p \land \neg p)$ is satisfiable: if we define I such that $I(p) = 0$, then $I \models \neg p$ and hence $I \models \neg p \lor (p \land \neg p)$. 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 $I \models p \land \neg p$.

2) (2 points) Let $\mathcal{P}$ be the set of four predicate symbols $\{p, q, r, s\}$.

2a) How many propositional formulas F built over $\mathcal{P}$ exist?

Answer: infinitely many (including p, $p \lor p$, $p \lor p \lor p$...).

2b) My friend John has a list $L = \{F_1, F_2, \ldots, F_{100000}\}$ of one hundred thousand formulas over $\{p, q, r, s\}$. He says that they are all logically non-equivalent, that is, $F_i \not\equiv F_j$ for all i, j with $1 \leq i < j \leq 100000$. What is the most efficient way to check whether John is right for a given L? Why? Your answer cannot be longer than 20 words.

Answer: Constant time. Just output: "no, John is not right". $F_i \not\equiv F_j$ means that F_i and F_j represent two different Boolean functions with 4 inputs, of which only $2^{(2^4)} = 2^{16} \approx 64000$ exist.

3) (4 points) Let C be the atleast-1 constraint $l_1 + l_2 + l_3 \geq 1$, where l_1, l_2, l_3 are literals, and let S be the set of five exactly-1 constraints

$$\{ \quad l_1 + a_1 + a_4 = 1, \quad l_2 + a_2 + a_4 = 1, \quad l_3 + a_3 = 1, \quad a_1 + a_2 + a_5 = 1, \quad a_3 + a_4 + a_6 = 1 \quad \}$$

where $a_1 \ldots a_6$ are distinct propositional symbols not occurring in C.

3A) Is it true that $S \models C$? Why? (answer in at most two lines).

Answer: Yes. Assume $S \not\models C$. Then $\exists\, I$ with $I \models S$ and $I \not\models C$, i.e., $\neg l_1, \neg l_2, \neg l_3$ true in I, implying a_3 true in I by constraint 3, $\neg a_4$ by 5, a_1 and a_2 by 1,2, contradicting constraint 4, i.e., that $I \models S$.

3B) Is it true that any model I of C can be extended to a model I' of S?
Here, by "extending" I to I' we mean that $eval_I(l_i) = eval_{I'}(l_i)$ and adequately defining the $I'(a_j)$.
Answer by just listing I' for the 7 cases of I, completing the table:

l_1	l_2	l_3	a_1	a_2	a_3	a_4	a_5	a_6
0	0	1	.	.	0	1	.	.
0	1	0	.	.				
.	.	.						

Answer: Yes:

l_1	l_2	l_3	a_1	a_2	a_3	a_4	a_5	a_6
0	0	1	0	0	0	1	1	0
0	1	0	1	0	1	0	0	0
0	1	1	1	0	0	0	0	1
1	0	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0	1
1	1	0	0	0	1	0	1	0
1	1	1	0	0	0	0	1	1

3C) *Exactly-1-SAT* is the problem of deciding the satisfiability of a given set S of exactly-1 constraints. What do you think is the computational complexity of exactly-1-SAT? (polynomial?, NP-complete?, harder?). Why?

Answer: NP-complete.
It is NP-hard since 2A and 2B show how to reduce 3-SAT to Exactly-1-SAT (note that $l_1 + l_2 + l_3 \geq 1$ is in fact a clause $l_1 \vee l_2 \vee l_3$).
It is in NP since we can reduce Exactly-1-SAT to SAT: each exactly-1 constraint generates one clause for atleast-1 and we can use any well-known encoding for the atmost-1 (quadratic, Heule, ladder,...).

3D) Same question if all exactly-1 constraints in S have the form $l + l' = 1$ for literals l and l'.

Answer: Polynomial. We can reduce it to 2-SAT, expressing each constraint $l + l' = 1$ by two clauses: $l \vee l'$ and $\neg l \vee \neg l'$.