Lógica en la Informática / Logic in Computer Science

Thursday January 9th, 2014

Time: 2h30min. No books, lecture notes or formula sheets allowed.

Questions 1,2,3 are the part of propositional logic, and 4,5,6 the part of first-order logic.

Results published: Monday Jan 20; Review/revisión: Wed Jan 22, 16h, Omega-139.

1a) Is it true that if F and G are propositional formulas, then $F \models G$ or $F \models \neg G$? Prove it using only the formal definitions of propositional logic.
Answer: It is false. Counter example: if there are two symbols, p and q, F is p and G is q, then neither $F \models G$ (e.g., if $I(p) = 1$ and $I(q) = 0$ then $I \models F$ but $I \not\models G$) nor $F \models \neg G$ (e.g., if $I(p) = 1$ and $I(q) = 1$ then $I \models F$ but $I \not\models \neg G$).

1b) Let F and G be propositional formulas such that $F \to G$ is satisfiable and F is satisfiable. Is it true that then G is satisfiable? Prove it using only the formal definitions of propositional logic.
Answer: It is false. Counter example: if F is p and G is $p \wedge \neg p$, then $F \to G$ is satisfiable (if $I(p) = 0$ then $I \models F \to G$) and F is satisfiable (if $I(p) = 1$ then $I \models F$) but $p \wedge \neg p$ is unsatisfiable.

2) Given a natural number n with $n > 1$, let F_n denote the propositional formula
$(p_{11} \wedge \ldots \wedge p_{1n}) \quad \vee \quad (p_{21} \wedge \ldots \wedge p_{2n}) \quad \vee \quad \ldots \quad \vee \quad (p_{n1} \wedge \ldots \wedge p_{nn})$.
2a) Write F_2 and write an equivalent formula in CNF without using any auxiliary variables. Do the same for F_3. Express how many clauses are needed in general for F_n, as a function of n.
Answer: The CNF for F_2 has the $2^2 = 4$ clauses: $p_{11} \vee p_{21}$, $p_{11} \vee p_{22}$, $p_{12} \vee p_{21}$, $p_{12} \vee p_{22}$. In general, the CNF for F_n has all clauses with exactly one literal from each conjunction in F_n, so it has has n^n clauses. Indeed, the CNF for F_3 has the $3^3 = 27$ clauses:
$p_{11} \vee p_{21} \vee p_{31}$, $p_{11} \vee p_{21} \vee p_{32}$, $p_{11} \vee p_{21} \vee p_{33}$, $p_{11} \vee p_{22} \vee p_{31}$, $p_{11} \vee p_{22} \vee p_{32}$
$p_{11} \vee p_{22} \vee p_{33}$, $p_{11} \vee p_{23} \vee p_{31}$, $p_{11} \vee p_{23} \vee p_{32}$, $p_{11} \vee p_{23} \vee p_{33}$
and the same 9 clauses with p_{12} instead of p_{11} and another 9 clauses with p_{13} instead of p_{11}.
2b) Write the Tseitin transformation of F_2. Is it logically equivalent to F_2? How many clauses are there in the Tseitin transformation of F_n, as a function of n?
Answer: F_2 is $(p_{11} \wedge p_{12}) \vee (p_{21} \wedge p_{22})$. A new symbol aux_1 is introduced for the outermost connective $\vee$, and two more aux_2 and aux_3 for the two $\wedge$ connectives. A unit clause aux_1 is generated for the root of the formular. Three more clauses are generated for aux_1:
$aux_1 \vee \neg aux_2$, $\quad aux_1 \vee \neg aux_3$, $\quad \neg aux_1 \vee aux_2 \vee aux_3$
and three clauses for aux_2 and three more for aux_3:
$\neg aux_2 \vee p_{11}$, $\quad \neg aux_2 \vee p_{12}$, $\quad aux_2 \vee \neg p_{11} \vee \neg p_{12}$
$\neg aux_3 \vee p_{21}$, $\quad \neg aux_3 \vee p_{22}$, $\quad aux_3 \vee \neg p_{21} \vee \neg p_{22}$
This Tseitin transformation is not logically equivalent to F_2. It is only equisatisfiable (that is, a formula is satisfiable iff its Tseitin transformation is satisfiable). Since F_n has $n^2 - 1$ connectives, the Tseitin transformation of F_n will have $1 + 3(n^2 - 1) = 3n^2 - 2$ clauses.

3) Recently we got a visit from N students from the Ecole Normale Superieure de Cachan (Paris). First we had a session where each one of 9 research groups of our LSI department gave a short talk. After each that, each student $i \in 1..N$ selected a subset $\{i_1, i_2, i_3\} \subset \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ with 3 of the 9 groups to receive a long talk from those 3 groups, for which there were 3 slots (named A, B and C) for the remainder of the day. Note that if a certain student chooses, for example, talks 3, 5 and 6, then these three long talks must be given in three different slots, but not necessarily in that order.

Obviously, one possibility is to give all the long talks in all 3 the slots (27 talks in total), but here at most K talks, with $K < 27$ are allowed.

3a) Explain how to use a SAT solver for scheduling the talks. If you use any AMO, cardinality or pseudo-Boolean constraints, it is not necessary to convert these into CNF.

Hint: note that if a student chooses, for example, talks 3, 5 and 6, then it suffices to state that talks 3 and 5 are given in (at least) two different slots, and also 3 and 6 in different slots, and also 5 and 6. Furthermore, to state that two talks are given in (at least) two different slots out of three slots, it suffices to force that in any pair of slots at least one of the two talks is given.

Answer: we introduce 27 variables $x_{t,s}$ meaning "talk t is given (at least) in slot s". We need one cardinality constraint: $x_{1A} + x_{1B} + x_{1C} + \cdots + x_{9A} + x_{9B} + x_{9C} \leq K$. In addition, we need nine four-literal clauses per student i with for each pair $\{t, t'\} \subset \{i_1, i_2, i_3\}$, three clauses to express that talks t and t' are given in (at least two) different slots:

$x_{tA} \vee x_{tB} \vee x_{t'A} \vee x_{t'B}, \quad x_{tA} \vee x_{tC} \vee x_{t'A} \vee x_{t'C}, \quad x_{tA} \vee x_{tC} \vee x_{t'B} \vee x_{t'C}.$

For example, to express that talks 3 and 5 are given in different slots we would have the clauses:

$x_{3A} \vee x_{3B} \vee x_{5A} \vee x_{5B}, \quad x_{3A} \vee x_{3C} \vee x_{5A} \vee x_{5C}, \quad x_{3B} \vee x_{3C} \vee x_{5B} \vee x_{5C}.$

3b) Express that no talk is given more than twice.

Answer: For each talk t, we would need a clause $\neg x_{t,A} \vee \neg x_{t,B} \vee \neg x_{t,C}$ (total, nine 3-literal clauses).

3c) How could you use the SAT solver to find the solution with the minimal total number K of talks? If $K = 27$ obviously the problem is satisfiable, so we try first with

$x_{1A} + x_{1B} + x_{1C} + \cdots + x_{9A} + x_{9B} + x_{9C} \leq 26.$

If we get a solution, we make another call to the SAT solver with

$x_{1A} + x_{1B} + x_{1C} + \cdots + x_{9A} + x_{9B} + x_{9C} \leq 25,$

and we keep making calls to the SAT solver with each time smaller Ks until it returns "unsatisfiable". The smallest K for which the SAT solver finds a solution is the optimal one.

4a) Is there any procedure that takes as input a formula F of first-order logic, and that always terminates saying "yes" if F is satisfiable, and that always terminates saying "no" if F is unsatisfiable? If so, briefly explain how it works.

Answer: No. This problem is not decidable.

4b) Is there any procedure that takes as input two formulas F and G of first-order logic, and that always terminates saying "yes" if $F \models G$, and that always terminates saying "no" or does not terminate if $F \not\models G$? If so, briefly explain how it works.

Answer: Yes. Let S be the clausal form of $F \wedge \neg G$. The procedure systematically computes the closure of S under resolution and factoring. It will terminate saying "yes" as soon as the empty clause appears (which always happens iff $F \models G$), and tt will terminate saying "no" if resolution and factoring terminate without empty clause (but termination may never happen).

4c) Is there any procedure that takes as input a formula F of first-order logic, and that always terminates saying "yes" if F is satisfiable, and that always terminates saying "no" or does not terminate if F is unsatisfiable? If so, briefly explain how it works.

Answer: No. This problem is not semi-decidable. It is co-semi-decidable: the procedure alwys terminates saying "no" if the answer is "no", but it may not terminate if the answer is "yes".

5a) Consider the first-order interpretation I where:

$D_I = \{0, 1, 2\}$, $a_I = 2$, and $P_I(n, m) = 1$ if and only if $m = (n + 1)$ *modulo* 3.

Let F be the formula $\forall x\, \exists y\, \exists z\, P(a, y) \wedge (P(y, z) \vee P(z, x))$. Do we have $I \models F$? Prove it.

Answer: Yes. For any element x of D_I, if we choose $y = 0$ and $z = 1$ the formula evaluates to 1.

5b) Let F be the first-order formula $\forall x\, \exists y\, \forall z\, P(x, y, z) \wedge Q(y)$, and let G be $\exists y\, \forall x\, \forall z\, Q(x) \wedge P(x, y, z)$. Are they logically equivalent? Is any one of the two a logical consequence of the other one? Prove it.

Answer: They are not logically equivalent because $F \not\models G$: for the interpretation I where $D_I = \{a, b\}$, $P_I(...) = 1$ always, $Q_I(a) = 1$ and $Q_I(b) = 0$, we have that $I \models F$ but $I \not\models G$. It is the case however that $G \models F$. We show that $G \wedge \neg F$ is unsatisfiable. In clausal form, G gives two clauses: $Q(x)$ and

$P(x', b, z)$. The formula $\neg F$ gives one clause: $\neg P(a, y, f(y)) \vee \neg Q(y)$. In two simple resolution steps we obtain the empty clause.

6) We have three dice (a die in Spanish is "dado", and the plural of die is dice). They are *fair* (each one of their six sides has the same probability of coming up) and their sides have numbers between 1 and 9 (not between 1 and 6!). Now suppose we play a game (many times): I pick a die; after that, you pick another die, we roll both dice, and the player who gets the highest number receives one Euro from the other player. Can you design the dice (putting the numbers on them) in such a way that you can become rich, that is, so that you can always pick a die that is *better* than mine (here *better* means that it wins with probability $p > 0.5$)? Write a Prolog program that checks whether this is possible or not. Include all non-predefined predicates you use.

To make the problem easier, assume that die A has number A1 on two of its sides, A2 on two sides and A3 on two sides, and similarly, die B has B1, B2, B3 and die C has C1, C2 and C3 (each number on two sides), where all nine numbers A1,A2,A3, B1,B2,B3, C1,C2,C3 are different and between 1 and 9. Also note that die A is better than die B if A wins in at least five of the nine possible outcomes (A1,B1),(A1,B2),...,(A3,B3), and that you have to make die A better than die B, die B better than C, and C better than A.

Answer:

```
p:-     permutation( [1,2,3,4,5,6,7,8,9], [A1,A2,A3,  B1,B2,B3,  C1,C2,C3] ),
        wins( [A1,A2,A3], [B1,B2,B3] ),
        wins( [B1,B2,B3], [C1,C2,C3] ),
        wins( [C1,C2,C3], [A1,A2,A3] ),
        nl, write( [A1,A2,A3]-[B1,B2,B3]-[C1,C2,C3] ), nl, halt.

wins(A,B):- findall( X-Y, (member(X,A),member(Y,B),X>Y),  L ),  length(L,K),  K>=5.

% this writes:  [1,5,9]-[3,4,8]-[2,6,7]
% Well-known definitions of member, length and permutation to be added.
```