Lógica en la Informática / Logic in Computer Science

Jan 16th, 2013.

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

Note: The answers will also be considered for evaluation of the transversal competences of English and Reasoning (but both with 0 % impact on the global evaluation of LI).
Questions 1-4 correspond to the propositional logic part.

1) For each one of the following statements, indicate whether it is true or false, without giving any explanations why.

1. Let F, G, H be formulas. If $F \land G \not\models H$ then $F \land G \land H$ is unsatisfiable. **False**
2. If F is unsatisfiable then $\neg F$ is a tautology. **True**
3. The formula $\forall x\ p(x) \lor \forall x\ q(x)$ is a logical consequence of the formula $\forall x\ (p(x) \lor q(x))$. **False**
4. The formula $\forall x\ (p(x) \lor q(x))$ is a logical consequence of the formula $\forall x\ p(x) \lor \forall x\ q(x)$. **True**
5. The formula $\forall x\ p(x) \lor \forall y\ \neg p(y)$ is a tautology. **False**
6. If F is unsatisfiable, then for every G we have $F \models G$. **True**

2A) Let F be a formula. Is it true that if F is unsatisfiable then $\neg F \models F$? Prove it.

Answer: False. Counter example: let F be $p \land \neg p$, which is unsatisfiable. Then $\neg F$ is $\neg(p \land \neg p)$ which is $\neg p \lor \neg\neg p$ which is $\neg p \lor p$. Then $\neg F \not\models F$ because, for instance, if $I(p) = 1$, we have $I \models \neg F$ but $I \not\models F$.

2B) If $F \to G$ is a tautology and F is a tautology, is it true that then G is a tautology? Prove it.

Answer: Yes, it is true. If $F \to G$ is a tautology and F is a tautology then, by definition of tautology, for all interpretations I we have $I \models F \to G$ and $I \models F$. Then (by definition of $\to$) for all I we have $I \models \neg F \lor G$ and $I \models F$. Then, by definition of $\models$, for all I we have $eval_I(\neg F \lor G) = 1$ and $eval_I(F) = 1$. Then, by definition of $\lor$, for all I we have $max(eval_I(\neg F), eval_I(G)) = 1$ and $eval_I(F) = 1$. Then, by definition of $\neg$, for all I we have $max(1 - eval_I(F), eval_I(G)) = 1$ and $eval_I(F) = 1$. Then for all I we have $max(0, eval_I(G)) = 1$. Then for all I we have $eval_I(G) = 1$. Then, by definition of $\models$, for all I we have $I \models G$, and, by definition of tautology, G is a tautology.

3) Consider the particular restriction of propositional resolution where at least one of the two premises must be a unit clause (a clause with just one literal L):

$$\frac{L \qquad L' \lor C}{C} \qquad \text{where } L \text{ is some predicate symbol } p \text{ and } L' \text{ is } \neg p\text{, or vice versa}$$

Is this deduction rule refutationally complete? Prove it.

Answer: No. It is not refutationally complete, because there exist unsatisfiable sets of clauses S from which by this rule we cannot obtain the empty clause. For example, if S is $\{\ p \lor q,\ \neg p \lor q,\ p \lor \neg q,\ \neg p \lor \neg q\ \}$, it is unsatisfiable, but this deduction rule infers nothing from S (because there is no unit clause).

Identify a class of clauses, the more general the better, for which this rule is refutationally complete. Explain why.

Answer: It is complete for Horn clauses. To know why, see the corresponding exercises 22-25 of lecture notes p3sols.pdf, where a more restrictive deduction rule is treated, namely *positive unit resolution*, where L has to be a positive literal p.

Note for advanced students: It is not difficult to prove that if L can be *any* unit clause (i.e., any literal), as here, then this rule is refutationally complete for a wider class of clauses than just Horn clauses, which is called "Renamable Horn" (those clause sets that can be turned into Horn sets by taking a subset of the symbols $p_1 \ldots p_n$ and flipping their signs everywhere, i.e., replacing every occurrence of p_i by its negation; note that this preserves satisfiability). Also note that Renamable Horn is exactly the class of claus sets that a SAT solver can decide by only doing unit propagation! So SAT is polynomial for Renamable Horn clause sets.

4) After consulting a dietician, we have a list of N nutrients that one must eat at least once a week. We also know, for each one of the P products of a grocery store, which of those nutrients it contains. Given an integer K, we want to know whether it is possible to buy at most K products so that all N nutrients are included in at least one of the products. Write a propositional formula that, with the help of a SAT solver, will allow you to solve this problem.

Answer: For each product i, with $1 \leq i \leq P$, we introduce a propositional variable x_i meaning that "we buy product i".

We need clauses to express that at most K products are bought, that is, at-most-$K(x_1 \ldots x_P)$, or equivalently, $x_1 + \ldots + x_P \leq K$. This we can do with any at-most-K encoding (ladder, sorting networks...).

We also need N clauses, one for each nutrient, to express that each nutrient is included in at least one of the products we buy. This we can do by adding for each nutrient a clause $x_{i_1} \vee \ldots \vee x_{i_m}$ if this nutrient is contained in products $\{i_1, \ldots, i_m\}$.

5) Consider the following four sentences:

A: Jack is Rob's brother.

B: Mike is Rob's brother.

C: If one person is someone's brother, then this second person also is a brother of the first one.

D: Jack is Mike's brother.

Translate each one of these sentences into first-order logic. Prove (by reasoning about interpretations) that $A \wedge B \wedge C \not\models D$. Write in first-order logic an additional clause E that defines a general property of brothers, such that $A \wedge B \wedge C \wedge E \models D$, and prove $A \wedge B \wedge C \wedge E \models D$ by resolution.

Answer: Let $B(x,y)$ mean "x is y's brother". Then in FOL $A \wedge B \wedge C \not\models D$ becomes

$$B(Jack, Rob) \wedge B(Mike, Rob) \wedge (\forall x \forall y\ B(x,y) \rightarrow B(y,x)) \not\models B(Jack, Mike)$$

To prove that $A \wedge B \wedge C \not\models D$ (as always, in any logic, when proving that a certain formula is *not* a logical consequence of some other formula), we need to define an interpretation I such that $I \models A \wedge B \wedge C$ but $I \not\models D$. Let I be the interpretation where
$D_I = \{a, b\}$, $\quad Rob_I = a$, $\ Jack_I = Mike_I = b \quad$ and $\quad B_I(x,y) = \ (x \neq y)$.
Then $I \models A \wedge B \wedge C$ but $I \not\models D$.

Clearly, the property that is missing is transitivity: $\forall x\ \forall y\ \forall z\ B(x,y) \wedge B(y,z) \rightarrow B(x,z)$. Then we can prove $A \wedge B \wedge C \wedge E \models D$ by resolution, as follows:

A	$B(Jack, Rob)$	
B	$B(Mike, Rob)$	
C	$\neg B(x,y) \vee B(y,x))$	
E	$\neg B(x,y) \vee \neg B(y,z) \vee B(x,z)$	
$\neg D$	$\neg B(Jack, Mike)$	
1	$B(Rob, Mike)$	$(B+C)$
2	$\neg B(Rob, z) \vee B(Jack, z)$	$(A+E)$
3	$B(Jack, Mike)$	$(1+2)$
4	$[]$	$(\neg D+3)$

6) John, Paul and Ringo are rich. Here we will count their money in natural numbers that represent millions of Euros. They all have more than zero and at most 10. John has at least twice the amount that Ringo has. Paul has at least 3 more than John. Ringo has at least 3.

A: Write all solutions.

B: Write a Gnu Prolog program with finite domain constraint propagation to find all solutions.

C: What are the domains of each variable after fully propagating the constraints (before the labeling)?

Answer: There are 3 solutions:
John has 6, Paul has 9, Ringo has 3;
John has 6, Paul has 10, Ringo has 3;
John has 7, Paul has 10, Ringo has 3.

```
p:- L = [J,P,R], fd_domain(L,1,10),
    J #>= 2*R,
    P #>= J+3,
    R #>= 3,      % writing L here would give: [_#3(6..7),_#24(9..10),3]
    fd_labeling(L), write(L), nl, fail.
% this program wites:
% [6,9,3]
% [6,10,3]
% [7,10,3]
```

Before labeling, after propagation, the domains of `J,P,R` are respectively [6,7], [9,10] and [3].

7) In order to have a nice studying place at home, we decide to build a table. After some investigation, we know that we need the following tools:

`E = [screws,hammer,saw,screwdriver,wood,paint,brush].`

Once we go to the appropriate shop, we realize that in order to save money we must buy packs of tools. The following packs, all with the same price, are available:

```
P = [[screws,screwdriver],
     [wood,brush,paint],
     [hammer,wood],
     [saw,screws],
     [paint,brush],
     [hammer,saw,screwdriver] ].
```

Construct a Prolog predicate `buy(E,P)` that, given a set of tools `E` and a pack list `P`, writes the minimum number of packs that one needs to buy in order to have all necessary tools.

In this case, a possible output would be:

```
Number of packs: 3
[[screws,screwdriver],[wood,brush,paint],[hammer,saw,screwdriver]]
```

Answer:

```
buy(E,P):- nat(N), subset(P,S), length(S,N), coveredBy(E,S),
           write('Number of packs: '), write(N), nl, write(S), nl.

%           coveredBy(E,S): "each tool of E is in some pack of S"
coveredBy( [],    _ ).
coveredBy( [X|L], S ):- member(P,S), member(X,P), coveredBy(L,S).

nat(0).
nat(N):- nat(N1), N is N1+1.

subset( [], [] ).
subset( [X|L], [X|S] ):- subset(L,S).
subset( [_|L], S     ):- subset(L,S).
```