Languages for AI, Module 1
January 18, 2021

Time: 2 hours.

In the following $A, B, \ldots$ are propositional variables, $a, b, \ldots$ constant symbols, $f, g, \ldots$ function symbols, $X, Y \ldots$ variables, $p, q, \ldots$ predicate symbols and $F, G, \phi, \psi, \ldots$ formulas (unless differently specified).

1. (4 points) Consider the language of propositional logic. Use natural deduction to prove that the following holds, or find a counter-example to show that it does not hold

   - $((F \to G) \wedge (G \to \neg F)) \to \neg F$
   - $(F \to G) \to (G \to F)$

   Solution:

   - For the first deduction:

$$\dfrac{\dfrac{\dfrac{[(F \to G) \wedge (G \to \neg F)]_2}{(G \to \neg F)} \wedge E \qquad \dfrac{\dfrac{[(F \to G) \wedge (G \to \neg F)]_2}{(F \to G)} \wedge E \qquad [F]_1}{G} \to E}{\dfrac{\dfrac{\neg F}{\bot} \to E}{\neg F} RAA_1} \to I_2}{((F \to G) \wedge (G \to \neg F)) \to \neg F}$$

   - For the second part, $(F \to G) \to (G \to F)$ cannot be proved because this formula is false, as one can verify by considering F = false and G = true.

2. (3 points) Transform the following propositional logic formula into an equivalent formula in Disjunctive Normal Form

   - $\neg(A \wedge B) \vee (\neg B \wedge ((C \wedge D) \to A))$

   Solution:

$$\begin{array}{l} \neg(A \wedge B) \vee (\neg B \wedge ((C \wedge D) \to A)) \equiv \\ \neg A \vee \neg B \vee (\neg B \wedge ((C \wedge D) \to A)) \equiv \\ \neg A \vee \neg B \end{array}$$

   The last step can be justified by observing that $(B \wedge C) \to B$ for any C, and that if $C \to B$ then B is equivalent to $B \vee C$ .The exercise of course could have been done also in the "normal" way, transforming into DNF the $(\neg B \wedge ((C \wedge D) \to A))$ formula.

3. (4 points) Prove that that $\phi \models \psi$ (ψ is a logical consequence of ϕ) or that $\phi \not\models \psi$ for the following formulas:

   - $\phi : (A \to B) \wedge (A \to C)$ and $\psi : \neg C \to (\neg A \wedge \neg B)$
   - $\phi : A \vee \neg C \vee B$ and $\psi : ((A \to C) \to (B \to A)) \to B$

   Solution:

   In both cases $\phi \not\models \psi$, which can be easily verified by writing down the truth tables for the formulas.

4. (5 points) Anna, Max and Bob carpool to work. On any day one, and only one, of them is the driver and the remaining two are the passengers. Tre driver cannot drive for more than two consecutive days and each person must drive at least once in seven days. Formalize the problem using propositional logic.

   Solution:

   Consider the following propositional variables:

   A_k, meaning that Anna is the driver on day k, M_k, meaning that Max is the driver on day k, B_k, meaning that Bob is the driver on day k.

We have the following axioms:

$$i)(A_k \wedge \neg M_k \wedge \neg B_k) \vee (\neg A_k \wedge M_k \wedge \neg B_k) \vee (\neg A_k \wedge \neg M_k \wedge B_k)$$
$$ii)((A_{k-2} \wedge A_{k-1}) \rightarrow \neg A_k)) \wedge (M_{k-2} \wedge M_{k-1}) \rightarrow \neg M_k)) \wedge (B_{k-2} \wedge B_{k-1}) \rightarrow \neg B_k))$$
$$iii) \bigvee_{i=1}^{7} A_i \wedge \bigvee_{i=1}^{7} M_i \wedge \bigvee_{i=1}^{7} B_i$$

5. (5 points) Formalize in first order logic the train connections in Italy. Provide a language that allows to express the fact that a town is directly connected (no intermediate train stops) with another town, by a type of train (e.g., intercity, regional, interregional). Formalize the following facts by means of axioms:

   (a) There is no direct connection from Rome to Trento

   (b) There is an intercity from Rome to Trento that stops in Firenze, Bologna and Verona.

   Solution:

   The axioms that formalizes the situation described in the exercise are the following:

   (a)

   $$\neg \exists DirectConn(X, rm, tn)$$

   (b)

   $$\exists X.(DirectConn(X, rm, fi) \wedge DirectConn(x, fi, bo) \wedge DirectConn(x, bo, vr) \wedge$$
   $$\wedge DirectConn(X, vr, tn) \wedge TrainType(X, interCity))$$

   where rm, fi, vr, tn are constants with the obvious mneaning, DirectConn(X,Y,Z) is a predicate which means that there exists a train X which goes directly from Y to Z, and TrainType(X, Y) is a predicate which means that Y is the type of train X.

6. (5 points) Morgan is a traveling salesperson that must visit three clients: Anton, Beth, and Carlos. The house of Morgan is distant 3 hours from each client. The house of Anton is distant 2 hours from Beth, and 3 from Carlos. The houses of Beth and Carlos are only 1 hour distant from each other. Beth must be visited before 14, because she is very busy in the afternoon. Knowing that Morgan won't leave the house before 9 and want to be back at home at most at 18, write a CLP program or a MiniZinc to compute the possible timetables for Morgan, including leaving and returning home.

   Solution:

```
salesperson(VARIABLES) :-
                length(VARIABLES, 5),
                VARIABLES = [H1,A,B,C,H2],
                VARIABLES ins 9..18,
                all_distinct(VARIABLES),
                B in 9..14,
                abs(A-B) #>= 2,
                abs(A-C) #>= 3,
                abs(B-C) #>= 1,
                abs(A-H1) #>= 3,
                abs(B-H1) #>= 3,
                abs(C-H1) #>= 3,
                abs(A-H2) #>= 3,
                abs(B-H2) #>= 3,
                abs(C-H2) #>= 3,
                H2 #> A, H2 #> B, H2 #> C, H1 #< A, H1 #< B, H1 #< C.
```

7. (5 points) Provide a Prolog program that defines a predicate `couples(L,C,N)` that, given a list of integers L, returns a list of couples C that contains all the pairs of adjacent numbers in L , where the second element of the couple is greater or equal than the first one. The counter N must return the number of pairs which have been discarded.

   For example, given the query
   `? -couples([0, 0, 12, 3, 11, 5, 7],C, N).`
   the program return the answer
   `C = [[0, 0], [0, 12], [3, 11], [5, 7]], N = 2`
   (the two discarded couples are [12,3] and [11,5]).

   Solution:

```
couples([],[], 0):- !.
couples([_|[]],[], 0):- !.
couples([E,F|L],[[E,F]|C],N):- couples([F|L], C, N), F>=E, !.
couples([E,F|L],C,N):- couples([F|L], C, M), F<E, N is M+1.
```

8. (3 points) The leftmost selection rule (used by Prolog) is the rule which selects the leftmost atom in a goal as the next one to be evaluated. The rightmost selection rule instead selects the rightmost atom (as the next one to be evaluated). Is it possible to provide a Prolog program P and a goal G such that:

   (a) the evaluation of G in P by using the leftmost selection rule terminates and

   (b) the evaluation of G in P by using the rightmost selection rule does not terminates ?

   Provide a short explanation for your answer.

   Yes, it is possible, just consider the goal p(a) in the following Prolog program:

   ```
   p(X):- q(X), p(X)
   q(b).
   ```

   With the leftmost selection rule the goal p(a) finitely fails (because a and b cannot be unified), while with the rightmost selection rule it does not terminate (since it is always used the 1st clause only).