Erasmus Mundus master course in **Data Mining and Knowledge Management** a european master

UPMC SORBONNE UNIVERSITÉS

Logic and Knowledge Representation

Assignment n° 1

Year 2015-2016
J.-G. Ganascia
M.-J. Lesot

Remark

- You should not need more than 4 hours to solve the exercises.
- The solutions must be submitted as two files on the moodle website
  - a file whose name is of the form yourLastName.pdf containing the solution to the logic part (exercices 1 to 4)
  - a file whose name is of the form yourLastName.pl containing the solution to the prolog part (exercice 5).
- The submission deadline is Monday November 9th, 6:00pm.

1 Propositional calculus

Exercise 1 – Interpretation and semantics

Consider the formulas given below, where p, q and r are propositional variables:

$F_1 = (p \supset q) \supset q$
$F_2 = (p \supset q) \supset p$
$F_3 = (p \supset (q \vee r)) \supset ((p \supset q) \vee (p \supset r))$

1. Give the truth table of each formula.
2. Indicate for each of them whether it is valid, contradictory or satisfiable. For each satisfiable formula, give an interpretation that falsifies it and an interpretation that satisfies it.
3. Write F_3 in the normal conjunctive form and the normal disjunctive form.
4. Indicate whether the following semantic consequence holds, justifying your answer.

   $\{(p \wedge q) \vee r, \neg p \supset \neg q\} \models r$

Exercise 2 – Semantic trees

A nutritional advisor proposes the following rules to make up a meal:

(a) meals with meat do not include fish
(b) meals without dessert include cheese
(c) no cheese without salad
(d) meals with salad include meat and dessert
(e) meals with dessert include salad
(f) meals include fish if and only if they include dessert

1. Formalise this knowledge in **propositional calculus**, using propositional variables such as meat, salad, dessert, etc.

   Remark: meals should not appear explicitly in the formulas as all assertions apply to them.
2. Using a semantic tree, show that this dietician forces every one to a drastic diet: show that it is not possible to make up a meal respecting the rules.

3. If a truth table was used for the same proof, how many rows would it have?

   Comparing with the number of leaves of your semantic tree, give a measure of the efficiency of the semantic tree method for this example.

Exercise 3 – Formal systems

1. Justifying each step and using the **Hilbert system** and the deduction theorem, prove that

$$\vdash (\neg p \supset (q \supset \neg r)) \supset ((\neg p \supset q) \supset (r \supset p))$$

2. Prove the same result using **sequent calculus**, again justifying each step.

2 Predicate calculus

Exercise 4 – Formalisation

Consider the following predicates and functions

Predicates		Function et constants	
$sunny(d)$	on day d, the sun shines	$today$	names today
$goesToSea(x, d)$	x goes to sea on day d	e	names Emile
$neighbour(x, y)$	x is y neighbour	$nextDay(d)$	names the day after d

Model the following assertions in predicate calculus

1. Tomorrow will be a sunny day
2. On sunny days, Emile goes to sea.
3. On non sunny days, none of Emile's neighbours goes to sea.
4. The sun never shines two days in a row
5. One of Emile's neighbours never goes to sea when Emile does
6. There is a day when all neighbours of Emile's neighbours go to sea.
7. All neighbours of Emile's neighbours go to sea at least once.
8. None of Emile's neighbours goes to sea two days in a row.

3 Prolog: list manipulation

Tail recursion solutions are required.

If you introduce auxiliary predicates, **comment** their roles, their expected behaviours and their parameters. Generally, comment your code.

Exercise 5

1. Define the predicate `second/2` such that `second(L, X)` is satisfied if `X` is the second element of the list `L`. The predicate must fail if `L` has less than 2 elements. For instance

```
?- second([a,b,c,d],X).
X = b.
?- second([a],X).
false.
?- second(L,a).
L = [_G268, a|_G272].
?-
```

2. Give 2 definitions of the predicate `removeLast/2` such that `removeLast(L, R)` is satisfied if `R` is the list obtained when removing the last element of `L`.

   `removeLast1` must use the `append` predicate, `removeLast2` must not use it.

   For instance, in both cases

```
?- removeLast([1,5,8,2], R).
R = [1, 5, 8] ;
false.
?-
```

3. Define the predicate `replace/4` such that `replace(L, X, Y, R)` is satisfied if `R` is the list obtained when replacing in `L` all occurrences of `X` with `Y`. For instance

```
?- replace([1, 5, 8, 1, 4, 3], 1, 12, L1).
L1 = [12, 5, 8, 12, 4, 3].
?- replace([1, 5, 8, 1, 4, 3], X, Y, [12, 5, 8, 12, 4, 3]).
X = 1,
Y = 12.
?-
```

4. Define the predicate `correspondence/3` such that `correspondence(+L1, +L2, ?R)` is satisfied if, given a list `L1` and a list of couples `L2`, `R` is the list obtained when replacing each element of `L1` by the element it is associated to in list `L2`. If the element does not occur in the list `L2`, it must be kept as such.

   For instance the query `correspondence([3, 8, 1, 1], [[1, one],[2, two],[3, three]], R).` must return as unique unification `R = [three, 8, one, one]`.

5. Define the predicate `decompose/3` such that `decompose(L, LPos, LNeg)` is satisfied for lists of integers if `LPos` is the list of all positive integers occurring in `L` and `LNeg` is the list of all negative integers occurring in `L`. For instance

```
?- decompose([1, -6, 4, 3, 0, -8, -5], L1, L2).
L1 = [1, 4, 3],
L2 = [-6, 0, -8, -5].
?- decompose(L, [1, 4, 3],[-6, 0, -8, -5]).
L = [1, 4, 3, -6, 0, -8, -5] ;
false
?-
```

6. Consider repetitive lists of the form `[a,a,a,b,b,c,a,a,a,a]` and their compact form, defined as lists of couples, `[[a,3],[b, 2],[c,1],[a,4]]`.

   Define the predicate `compress/2` such that `compress(+L1, ?L2)` is satisfied if, given a list `L1`, `L2` is its compact form.

7. Define the opposite predicate `decompress/2` such that `decompress(+L1, ?L2)` is satisfied if, given a list of couples `L1`, `L2` is its expanded form.