# *LKR – SD206*
# *(Logic and Knowledge Representation)*

Jean-Louis Dessalles

## Evaluation - April 2019

## Solutions

Q1. A directed graph is represented by a predicate `edge`. For instance, the fact `edge(a,b).` indicates the existence of an edge linking node `a` to node `b`. Write a Prolog predicate `path(X,Y)` which is true if there is an *acyclic* path from node `X` to node `Y`. The program should avoid being trapped in cycles if any.

```
path(X, Y) :-
    path1(X, Y, [X]).   % The last argument stores the list of visited nodes

path1(X, Y, _) :-
    edge(X, Y).

path1(X, Y, L) :-
    edge(X, Z),
    not(member(Z, L)),
    path1(Z, Y, [Z | L]).
```

Q2. What does the following Prolog expression mean: `[X|X].` ?
Provide two different possible values for X.

**[X|X] matches a list X in which the first element is a list that that can be unified with the rest of the list. Examples: X = [ ] or X = [a], corresoinding to [X|X] = [[ ]] and [[a],a] respectively. To the question ?- [X|X]., the interpreter yields only the empty list for X, as unification does not involve any mechanism to grow X's instantiations.**

Q3. German grammar is SVO (subject-verb-object) in the main clause (as in English or in French), but SOV in the subordinate clause. We consider German subordinate clauses that are complement of a noun, as in:

> "Das Buch, das Hans liest"
> (The book that Hans is reading)

Subordinate complement clauses are introduced by a comma, followed by a relative pronoun (here, 'das'), according to the pattern: NP, RP S O V. Suppose you start from a grammar that includes the DCG clauses for NP:

```
np --> pn.        % proper noun
np --> det, n.
```

Add new DCG clauses to allow for German subordinates as noun complements (the comma should be within quotes: `','`).

```
% one possible solution (among many) to represent subordinate clauses as noun complement
np --> pn.                  % proper noun
np --> det, n.
np --> det, n, [','], sc.   % to avoid left recursive rule
sc --> rp, np, ivp.         % subordinate clause
ivp --> np, v.              % inverted verb phrase
ivp --> v.

rp --> [das]; [die]; [der]. % ...
v --> [liest].
n --> ['Buch'].
pn --> ['Hans'].
det --> [das].

test :-
    np([das, 'Buch', ',', das, 'Hans', liest], []).
```

## Q4. Show that $\vdash (P \supset (Q \supset P))$.

**$\vdash$ indicates that the formula can be proven. Here is a proof by resolution.**

1. **$[\neg(P \supset (Q \supset P))]$**
2. **$[P]$**
3. **$[\neg (Q \supset P)]$**
4. **$[Q]$**
5. **$[\neg P]$**
6. **$[\,]$ resolving clause between 2 and 5.**

## Q5. Show using resolution and skolemization that the following formula is valid.

$(\exists w)\ (\forall x)\ R(x, w, f(x, w)) \supset (\exists w)\ (\forall x)\ (\exists y)\ R(x, w, y)$

**Proof by resolution.**

1. **$[\neg(\exists w)\ (\forall x)\ R(x, w, f(x, w)) \supset (\exists u)\ (\forall v)\ (\exists y)\ R(v, u, y)]$**
2. **$[\ (\exists w)\ (\forall x)\ R(x, w, f(x, w))]$**
3. **$[\neg\ (\exists u)\ (\forall v)\ (\exists y)\ R(v, u, y)]$**
4. **$[R(x, a, f(x, a))]$ skolemization of 2.**
5. **$[\neg R(sk(u), u, y)]$ skolemization of 3.**
6. **$[\,]$ unification of 4 and 5 with $x$=$sk(u)$ ; $u$=a ; $y$=$f(x,a)$ → $R(sk(a), a, f(sk(a), a))$**

Q6. Consider two examples described as Prolog facts:

```
daughter(mary,ann).
daughter(eve,tom).
```

and a background knowledge:

```
mother(ann,mary).
mother(ann,tom).
father(tom,eve).
father(tom,ian).
female(ann).
female(mary).
female(eve).
male(pat).
male(tom).
parent(X,Y) :- mother(X,Y).
parent(X,Y) :- father(X,Y).
```

Show that the following clause is more general than the first example:

```
daughter(X,Y) :- female(X), female(Y), mother(Y,X).
```

Induce a similar clause that covers the second example. Then induce a new clause that generalizes these two clauses.

**The clause**

**daughter(X,Y) :- female(X), female(Y), mother(Y,X).**

**once unified with female(ann), female(mary) and mother(ann,mary), gives the example daughter(mary,ann).**

**A similar clause obtained from the second example is:**

**daughter(X,Y) :- female(X), male(Y), father(Y,X).**

**A LGG between the two clauses is:**

**daughter(X,Y) :- female(X), parent(Y,X).**