LKR – SD206
(Logic and Knowledge Representation)

Jean-Louis Dessalles

Evaluation - April 2023

Duration: 85 minutes. No documents - No turned-on devices. Questions are independent.

Q1. Write the Prolog predicate `flatten` which processes a list by (recursively) replacing any element that is itself a list by enumerating its elements. For instance:

```
?- flatten([a, [b, [c, d], e]], X).
    X = [a, b, c, d, e]
```

You will use the standard predicate `is_list/1` which checks whether an element is a list.

Q2. 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.

Q3. ChatGPT proposed the following exercise (where P means premise and C conclusion):

> Prove the validity of the following argument:
> P1: All dogs are mammals.
> P2: Some mammals are furry.
> C: Therefore, some dogs are furry.

(a) Translate the elements of the question into formal logic.

(b) Explore the validity of the argument.

Q4. Show, using resolution and skolemization, that the following formula is valid.
$(\exists w)\ (\forall x)\ R(x, w, f(x, w)) \supset (\exists w)\ (\forall x)\ (\exists y)\ R(x, w, y)$

Q5. 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: `','`).

Q6. Find the least general generalization (lgg) of these rules:

1. `c(alice) :- a(alice, X), b(X).`
2. `c(robert) :- a(robert, X), d(X), e(X).`

based on the background knowledge:

```
f(X) :- b(X).

f(X) :- e(X).

d(X) :- not(g(X)).

g(X) :- b(X).

h(X,Y) :- a(X,Y).

h(X,Y) :- i(X,Y).
```

LKR – SD206
(Logic and Knowledge Representation)

Jean-Louis Dessalles

Evaluation - April 2023

Examples of answers

Q1. Write the Prolog predicat `flatten` which processes a list by (recursively) replacing any element that is itself a list by enumerating its elements. For instance:

```
?- flatten([a, [b, [c, d], e]], X).
       X = [a, b, c, d, e]
```

You will use the standard predicate `is_list/1` which checks whether an element is a list.

```
flatten([X|L], L1) :-
    is_list(X),
    !,
    flatten(X, X1),
    flatten(L, L2),
    append(X1, L2, L1).

flatten([X|L], [X|L1]) :-
    flatten(L, L1).

flatten([ ], [ ]).
```

Q2. 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]).
```

Q3. ChatGPT proposed the following exercise (where P means premise and C conclusion):

> Prove the validity of the following argument:
> P1: All dogs are mammals.
> P2: Some mammals are furry.
> C: Therefore, some dogs are furry.

(a) Translate the elements of the question into formal logic.

(b) Explore the validity of the argument.

(a)

P1:	$(\forall x)\ (dog(x) \supset mammal(x))$
P2:	$(\exists x)\ (mammal(x) \wedge furry(x))$
C:	$(\exists x)\ (dog(x) \wedge furry(x))$

(b)

The conclusion is *not* valid. Consider as domain the set of integers larger than 3.

***dog* is interpreted as 'even', *mammal* as 'integer' and *furry* as 'prime'. P1 reads "even integers (larger than 3) are integers"; P2 reads "some integers (larger than 3) are prime. The conclusion "some even numbers (larger than 3) are prime" and it is false.**

Q4. 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 **;** $y=f(x,a) \rightarrow R(sk(a), a, f(sk(a), a))$

Q5. 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], []).
```

Q6. Find the least general generalization (lgg) of these rules:

3. `c(alice) :- a(alice, X), b(X).`
4. `c(robert) :- a(robert, X), d(X), e(X).`

based on the background knowledge:

```
f(X) :- b(X).
f(X) :- e(X).
d(X) :- not(g(X)).
g(X) :- b(X).
h(X,Y) :- a(X,Y).
h(X,Y) :- i(X,Y).
```

The lgg of 1. and 2. is:

`c(Y) :- a(Y,X), f(X).`