LKR – SD206
(Logic and Knowledge Representation)

Jean-Louis Dessalles

Evaluation - April 2021

Eléments de corrigé

Q1. Write a Prolog program `second(List1, List2)` that keeps every second element of `List1` in `List2`. For instance, `second([a,b,c,d,e], L).` should instantiate `L` to `[a,c,e]`. Then indicate what one would get by calling
`second(L, [a,c,e]).`

```
second([ ], [ ]).
second([X], [X]).
second([X, _ | R], [X | R1]) :-
    second(R, R1).
```

```
?- second(L, [a,c,e]).
L = [a, _4818, c, _4830, e] .
```

Q2. Consider the following statements:

1. Every child loves every candy.
2. Anyone who loves some candy cannot be a nutrition fanatic.
3. Anyone who eats all pumpkins is a nutrition fanatic.

Translate them into first-order logic language. Then present the formulas in prenex form. Then transform them into skolemized form. Then present the result in conjunctive normal form.

1. $(\forall i)\ (\forall c)\ (\ (\mathrm{child}(i) \wedge \mathrm{candy}(c)) \supset \mathrm{loves}(i, c)\ )$
2. $(\forall i)\ (\ \ (\ (\exists c)\ (\mathrm{candy}(c) \wedge \mathrm{loves}(i, c))\ ) \supset \neg\mathrm{nfan}(i)\ \ )$
3. $(\forall i)\ (\ \ (\ (\forall p)\ (\mathrm{pumpkin}(p) \supset \mathrm{eat}(i, p))\ ) \supset \mathrm{nfan}(i)\ \ )$

1. $(\forall i)\ (\forall c)\ (\ (\mathrm{child}(i) \wedge \mathrm{candy}(c)) \supset \mathrm{loves}(i, c)\ )$
2. $\forall i)\ (\forall c)\ (\ (\mathrm{candy}(c) \wedge \mathrm{loves}(i, c)) \supset \neg\mathrm{nfan}(i)\ )$
3. $(\forall i)\ (\exists p)\ (\ \ (\mathrm{pumpkin}(p) \supset \mathrm{eat}(i, p)) \supset \mathrm{nfan}(i)\ \ )$

1. $(\ (\mathrm{child}(i) \wedge \mathrm{candy}(c)) \supset \mathrm{loves}(i, c)\ )$
2. $(\ (\mathrm{candy}(c) \wedge \mathrm{loves}(i, c)) \supset \neg\mathrm{nfan}(i)\ )$
3. $(\ \ (\mathrm{pumpkin}(p(i)) \supset \mathrm{eat}(i, p(i))) \supset \mathrm{nfan}(i)\ \ )$

1. **< [$\neg$child(i), $\neg$candy(c), loves(i, c)] >**
2. **< [$\neg$candy(c), $\neg$loves(i, c)), $\neg$nfan(i)] >**
3. **< [pumpkin($p(i)$), nfan(i)], [$\neg$eat(i, $p(i)$)), nfan(i)] >**

Q3. Use the resolution method to prove that:
$\{(\forall x)\ (P(x) \vee Q(x)), (\exists x)\ \neg P(x)\} \vdash (\exists x)\ Q(x)$.

Hypotheses :	**$[(\forall x)\ (P(x) \vee Q(x))]$**
	$[(\exists x)\ \neg P(x)]$
Negated conclusion:	**$[\neg(\exists x)\ Q(x)]$**
Skolemization:	**$[\ P(x), Q(x))]$**
	$[\neg P(a)]$
	$[\ \neg Q(a))]$
Two resolutions :	**[ ]**

Q4. Consider the small DCG grammar:

```
aff --> np, vp.
np --> [they]; [she].
np --> det, n.
vp --> v, np.
v --> [like].
det --> [the].
n --> [cake].
```

This grammar recognizes affirmative sentences such as "they like the cake".
Write a DCG program that recognizes interrogative sentences in English.
It should recognize sentences like (we only consider 3rd person):
"do they like the cake", "are they crazy", and even the incorrect sentence "is they crazy", but not "do they crazy".
Then propose a way to discard "is they crazy".

```
% First solution with no number feature
int --> aux1, np, vp.
int --> aux2, np, adj.
aux1 --> [do]; [does].
aux2 --> [are]; [is].
np --> [they]; [she].
np --> det, n.
vp --> v, np.
v --> [like].
det --> [the].
n --> [cake].
adj --> [crazy].
```

```
aux1(sing) --> [does].
aux1(plur) --> [do].
aux2(sing) --> [is].
aux2(plur) --> [are].
np(sing) --> [she].
np(plur) --> [they].
np(_) --> det, n.
vp --> v, np(_).
v --> [like].
det --> [the].
n --> [cake].
adj --> [crazy].
```

```
int --> aux1(Number), np(Number), vp.
int --> aux2(Number), np(Number), adj.
```

Q5. Find the least general generalization (lgg) of these rules:

```
1. c(lisa) :- a(lisa, X), b(X).
2. c(clara) :- a(clara, 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).`

Q6. In a game in which only numbers under 20 are considered, which number sequence is the simplest, and why?

1. 5, 7, 9, 11, 13, 15, 17, 19
2. 1, 3, 5, 7, 9, 13, 15, 17, 19
3. 1, 3, 5, 7, 17, 19

1. can be described as "odd numbers greater than 5", or "odd numbers but 1 and 3".

2. can be described as "odd numbers but 11".

3. can be described as "odd numbers except 9, 11, 13, 15".

The corresponding algorithms only differ by the constants involved.

We may consider that 1 and 3 together are simpler than 11 (1+2 bits instead of 4).

So 1. is simpler than 2., which is obviously simpler than 3 (even if 3. is shorter).

(Note: this hierarchy may vary if one is able to find shorter description than those).