Logic, Knowledge Representation and Probabilities – 0EL07

Nils Holzenberger

Evaluation - April 2025

Duration: 90 minutes. No documents - No turned-on devices. Questions are independent.

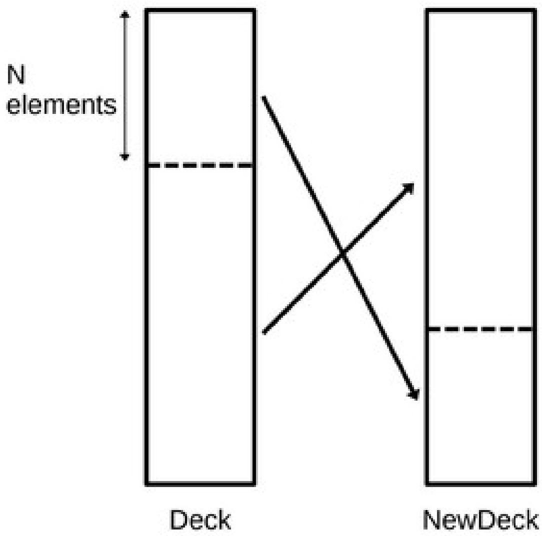
Q1.

a) Write a predicate `reverse/2` such that `reverse(A, B)` holds if list `B` is the reverse of list `A`.

```
?- reverse([a,b,c], X).

X = [c, b, a].
```

b) Write a predicate `cut_deck(Deck, N, NewDeck)` cutting the list `Deck` at position `N` to produce `NewDeck`. You may use `append(L1, L2, L3)` where `L3` is the concatenation of `L1` and `L2`, and you can use the previously defined `reverse/2`.




```
?- cut_deck([a,b,c,d,e], 2, X).

X = [c,d,e,a,b]

?- cut_deck([a,b,c], 1, X).

X = [b,c,a]
```

Q2.

a) Find a model in which the following formula is true:
$(\forall x)\ (\exists y)\ (A(y) \supset \neg A(x))$

b) Find a model in which the following formula is true:
$(\forall x)\ (\exists y)\ \neg(A(x) \supset B(y))$

Q3. Use the resolution procedure to show that

$$\{(\forall x)(P(x) \vee Q(x)), (\exists x)\neg P(x)\} \vdash (\exists x)Q(x)$$

Q4. Consider the following ProbLog program :

```
y/1. % y(X) means that X yawns.

.8::y(X) :- s(X,Y), y(Y). % R: if X sees Y and Y yawns, then X yawns.

s/2. % s(X,Y) means that X can see Y.

s(b, a). % S1: b (Bob) can see a (Alice).

s(c, b). % S2: c (Charlie) can see b (Bob).

.1::y(a). % Y1: a yawns, w.p. 0.1

.3::y(b). % Y2: b yawns, w.p. 0.3

.2::y(c). % Y3: c yawns, w.p. 0.2
```

Write out the SLD resolution tree of the query

```
?- y(c).
```

Be sure to specify which leaves are successful and which ones fail, and to label the edges between nodes with the label of the rule that was used (R, S1-2, Y1-3).

Q5. Let *T* be the following ProbLog program:

```
f(s3).
.8::t(s1,a,s1).
.7::t(s1,a,s2).
.5::t(s1,b,s1).
.9::t(s2,b,s3).
.2::t(s3,b,s4).
.1::s(s2,s4).
.1::s(s3,s1).
a(S,[]) :- f(S).
.3::a(S,[X|R]) :- t(S,X,T), a(T,R).
a(S, L) :- s(S,T), a(T,L).
```

Write down 2 samples from *T*, together with their probability (or at least the formula to compute it). Only write down samples with non-zero probability.

1. Eléments de corrigé

Q1.

a)

```
travelBetween(A, A). % base case 1

travelBetween(A, B) :- directTrain(A, B). % base case 2

travelBetween(A, B) :- travelBetween(A, B, []).

% keep track of cities visited to avoid loops

travelBetween(A, B, AlreadyVisited) :-

  directTrain(A, Z),

  \+ member(Z, AlreadyVisited),

  travelBetween(Z, B, [Z|AlreadyVisited]).

travelBetween(A, A, _). % base case
```

b) Add this line to the above program:

```
directTrain(A, B) :- A@>B, directTrain(B, A).
```

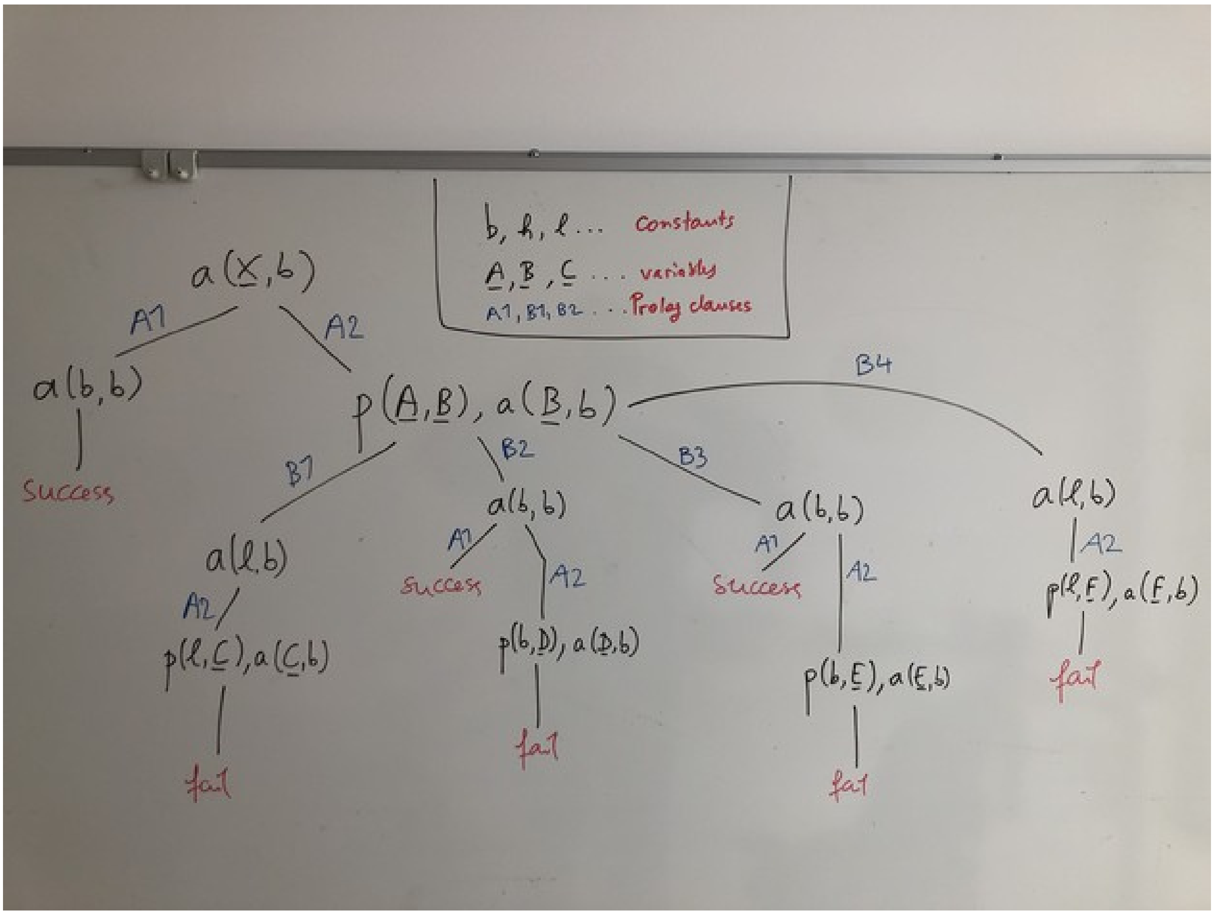
Q2. a) Domain is {0}. Interpretation of A is "x is 0". Interpretation of B is "False".

b) Domain is {0,1}. Same interpretations for A and B.

Q3. Proof by resolution.

1. $[\neg((a \supset b) \supset (\neg(b \vee (c \wedge d)) \supset \neg(a \vee (c \wedge d))))]$
2. $[(a \supset b)]$
3. $[\neg(\neg(b \vee (c \wedge d)) \supset \neg(a \vee (c \wedge d)))]$
4. $[\neg(b \vee (c \wedge d))]$ **from 3.**
5. $[\neg\neg(a \vee (c \wedge d))]$ **from 3.**
6. $[(a \vee (c \wedge d))]$ **from 5.**
7. $[a, (c \wedge d)]$ **from 6.**
8. $[\neg a, b]$ **from 2.**
9. $[\neg b]$ **from 4.**
10. $[\neg (c \wedge d)]$ **from 4.**
11. **[a]** **unification of 7 and 10.**
12. **[b]** **unification of 8 and 11.**
13. [ ] unification of 9 and 12

Q4.



Q5.

```
alarm :- earthquake.

calls(X) :- alarm, is_home(X).

is_home(alice).
```

p=.9x.99x.5x.9x.3

```
burglary.
alarm :- burglary.
alarm :- earthquake.
calls(X) :- alarm, is_home(X).
is_home(bob).
```

p=.1x.99x.5x.9x.7