Neuro-Symbolic AI – IA206

Nils Holzenberger

Evaluation - April 2024

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

Q1. We have the following knowledge base:

```
directTrain(forbach,saarbruecken).

directTrain(freyming,forbach).

directTrain(fahlquemont,stAvold).

directTrain(stAvold,forbach).

directTrain(saarbruecken,dudweiler).

directTrain(metz,fahlquemont).

directTrain(nancy,metz).
```

That is, this knowledge base holds facts about cities it is possible to travel between by taking a direct train. But of course, we can travel further by chaining together direct train journeys. In the following two questions, you get bonus points if you avoid infinite loops.

a) Write a recursive predicate travelBetween/2 that tells us whether we can travel by train between two cities. For example, when given the query

```
travelBetween(nancy,saarbruecken).
```

it should succeed.

b) It is, furthermore, plausible to assume that whenever it is possible to take a direct train from A to B, it is also possible to take a direct train from B to A. Can you encode this in Prolog? Your program should allow the following query to succeed:

```
travelBetween(saarbruecken,nancy).
```

Q2. Let *F* be the following formula in first-order logic: $(((\forall x)\, A) \supset B)$.

a) Provide a model in which *F* is false.

b) Provide a model in which *F* is true.

Q3. Use the resolution procedure to show that

$$((a \supset b) \supset (\neg(b \vee (c \wedge d)) \supset \neg(a \vee (c \wedge d))))$$

is a valid formula.

FYI, $(a \supset b) \equiv ((\neg a) \vee b)$

Q4. Consider the following Prolog program :

```
a/2. % a(X, Y) means that X is an ancestor of Y.

a(X, X). % A1: base case for ancestor clause

a(X, Y) :- p(X, Z), a(Z, Y). % A2: recursive call for ancestor

p/2. % p(X, Z) means that X is the parent of Z.

p(h, l). % B1: homer is the parent of lisa

p(h, b). % B2: homer is the parent of bart

p(m, b). % B3: marge is the parent of bart

p(m, l). % B4: marge is the parent of lisa
```

Write out the SLD resolution tree of the query

```
?- a(X, b).
```

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 (A1, A2, B1-4).

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

```
.1::burglary.

.01::earthquake.

.5::alarm :- burglary.

.9::alarm :- earthquake.

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

.3::is_home(alice) ; .7::is_home(bob).
```

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).
```

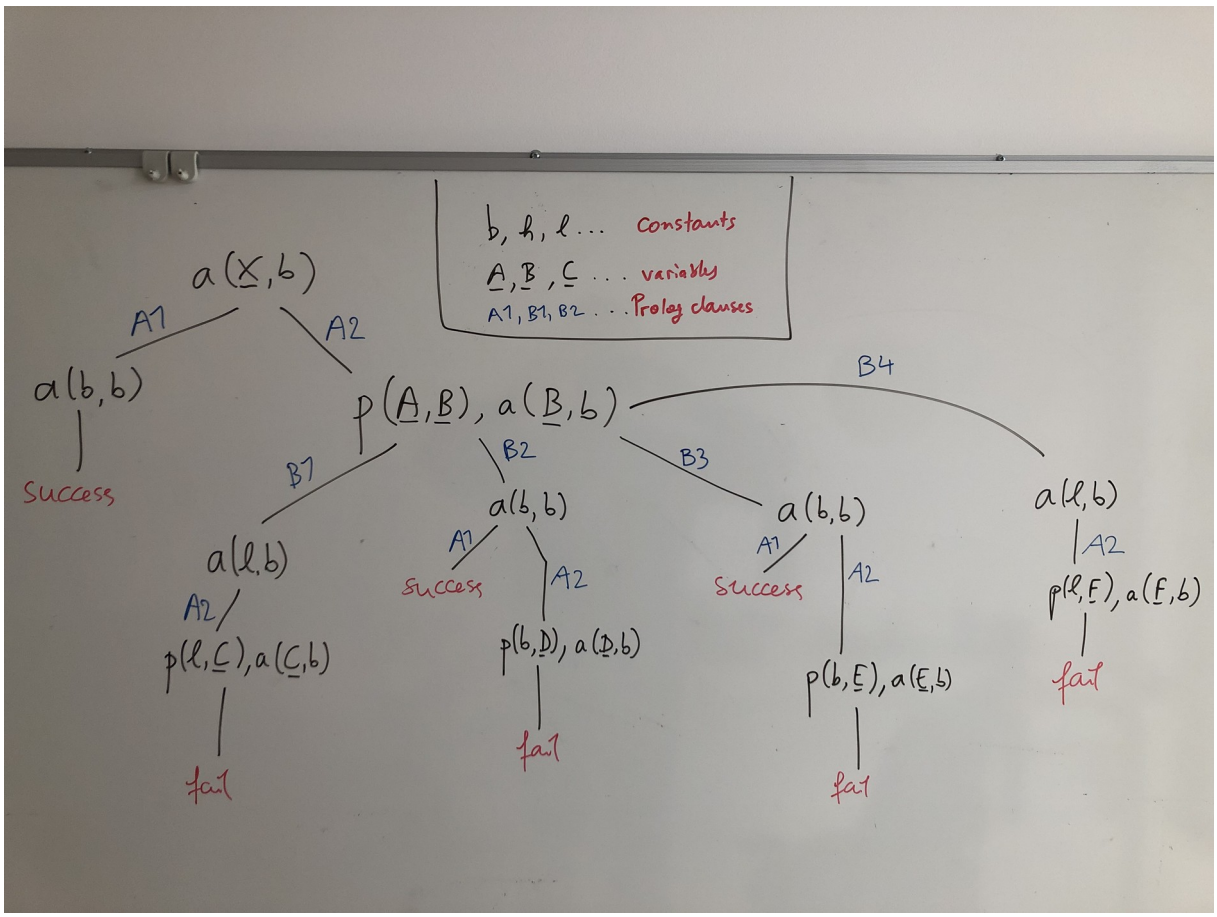
Q2. a) Domain is {0}. Interpretation of A is "x is 0". Interpretation of B is "False".

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

Q3. Proof by resolution.

1. **[¬((*a* ⊃ *b*) ⊃ (¬(*b* ˅ (*c* ˄*d*)) ⊃ ¬(*a* ˅ (*c* ˄*d*))))]**
2. **[(*a* ⊃ *b*)]**
3. **[¬(¬(*b* ˅ (*c* ˄*d*)) ⊃ ¬(*a* ˅ (*c* ˄*d*)))]**
4. **[¬(*b* ˅ (*c* ˄*d*))] from 3.**
5. **[¬¬(*a* ˅ (*c* ˄*d*))] from 3.**
6. **[(*a* ˅ (*c* ˄*d*))] from 5.**
7. **[*a*, (*c* ˄*d*)] from 6.**
8. **[¬*a*, *b*] from 2.**
9. **[¬*b*] from 4.**
10. **[¬ (*c* ˄*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