CS 560: Homework 3

Eric Stevens

October 21, 2018

Question 1

Let $I = \{D, \phi, \pi\}$ be any model of KB.
Let $\phi(a) = A$
Let $\phi(b) = B$
Let ρ be the variable assignment that maps a to A and b to B respectivly.

Since $p(X, Y) \in KB, I_\rho(P(X, Y))$ is true.
So, $\pi(p)(\rho(X), \rho(Y))$ is true.
So, $\pi(p)(A, B)$ is true.
Therefore, $I(p(a, b))$ is true.

Since I is a modle of KB it follows that $KB \models p(a, b)$

Question 1 Critique

Since there is no provided solution for this problem I cannot offer a critique. I based this solution on the solution from question 2.2.

Question 2

animal(X) <-- dog(X)
gets_pleasure(X) <-- animal(X) ^ feeding(X)
likes(X,Y) <-- person(X) ^ animal(Y) ^ owns(X,Y)
does_for(X,Y) <-- person(X) ^ animal(Y) ^ gets_pleasure(Y)

Question 2 Critique

Several mistakes were made in this solution. I missunderstood that feeding was a constant and entered it as a predicate. The provided solution took the form "X gets pleasure from feeding if X is an animal". My solution read "X gets pleasure if X is an animal and X is feeding. This mistake cascaded through the solution and caused. I was not able to the "does()" predicate 3-ary because my feeding predicate can not be an argument of another predicate. I added "person()" predicate which wasn't needed. Finally, I forgot to add the gound clauses.

Question 3

```
? does_for(mary,Y)
```

Question 3 Critique

Here my mistake from the previous problem led to a similar problem. I did not think of the action of feeding as a constant and instead thought of it as a predicate. Therefore I could not have a variable for action in the query.

Question 4

Did not complete

Question 5

```
?- append(A,[a|B],[b,a,d,a,c,a]).
A = [b],
B = [d, a, c, a] ;
A = [b, a, d],
B = [c, a] ;
A = [b, a, d, a, c],
B = [] ;
```

```
?- append(X,[A,B|Y],[a,b,c,d,e]).
X = [],
A = a,
B = b,
Y = [c, d, e] ;
X = [a],
A = b,
B = c,
Y = [d, e] ;
X = [a, b],
A = c,
B = d,
Y = [e] ;
X = [a, b, c],
A = d,
B = e,
Y = [] ;
```

Question 5 Critique

This solution appears to be correct

Question 6

```
?- subset([a,b],[b,a,d]).
true.
```

```
?- subset([X,Y],[b,a,d]).
X = a,
Y = a;
X = b,
Y = b;
X = d,
Y = d;
X = a,
Y = b;
X = b,
Y = a;
X = a,
Y = c;
X = c,
Y = a;
X = a,
Y = d;
X = d,
Y = a;
X = b,
Y = d;
X = d,
Y = b;
```

```
?- subset(S,[b,a,d]).
S = [];
S = [a];
S = [b];
S = [d];
S = [a,a];
S = [a,b];
S = [a,c];
...
```

This usage of set does not fit the mathematical difinition of a set because in this situation the same value can occour multiple times in a set. A dictionary would be a better structure for a set.

Question 6 Critique

This solution also appears to be mostly correct. Looking back at what I did in my haste, I probably should hav added several more examples to the final query befoe the ... I knew what I was doing here, every combination of three or less length list with a, b, and d as elements. This would have been $3^0 + 3^1 + 3^2 + 3^3$ lines so I did not write all of them.

I think my explanation of why the list is not the proper representation of a set is valid, but incomplete. I forgot to add that multiple lists can represent the same set.

Question 7

```
shorter([_],_).
shorter([_|B],[_|J]) <- shorter(B,J).


?- shorter([a,g],[b,a,d,a]).
true.


?- shorter([a,a,d,a],[a,g]).
false.


?- shorter([a,a,a,a],[a,b]).
false.


?- shorter(L,[b,a,d,a]).
L = [] ;
L = [W] ;
L = [W, X] ;
L = [W, X, Y] ;
L = [W, X, Y, Z] ;

?- shorter([b,a,d,a],L).
L = [W, X, Y, Z|K].
```

Question 7 Critique

My solutions to these problems is correct but I made some variations to the output that may have not been needed. Also my formatting of the prolog scripts was different because this is my first time writing Prolog and it was less confusing to write them this way. When I wrote them the way the solution has them I got "singleton variable" warnings and thought that I was doing something wrong. The way I wrote the code did not cause this warning. Also, when I got the ouput of the form _GXXX I replaced it with variables because I did not know what the _GXXX represented.

Question 8

```
remove(E,[E|Y], Y).
remove(E,[A|B],[A|C]) <- remove(E,B,C).
```

```
?- remove(a,[b,a,d,a],R).
R = [b, d, a] ;
R = [b, a, d] ;
```

```
?- remove(E,[b,a,d,a],R).
E = b,
R = [a, d, a] ;
E = a,
R = [b, d, a] ;
E = d,
R = [b, a, a] ;
E = a,
R = [b, a, d] ;
```

```
?- remove(E,L,[b,a,d]).
L = [E, b, a, d] ;
L = [b, E, a, d] ;
L = [b, a, E, d] ;
L = [b, a, d, E] ;
```

```
?- remove(p(X),[a,p(a),p(p(a)),p(p(p(a)))], R).
X = a,
R = [a, p(p(a)), p(p(p(a)))] ;
X = p(a),
R = [a, p(a), p(p(p(a)))] ;
X = p(p(a)),
R = [a, p(a), p(p(a))] ;
```

Question 8 Critique

This solution is correct. I did use different variable names.

Question 9

Question 10

```
plus(X,0,X).
plus(X, s(Y), Z) <-- plus(s(X), Y, Z).
```

Question 10 Critique

This solution is not the same as the solution provided. I see how the provided solution works, as Y counts down s()s are wrapped around Z and cout up on the way out. That being said I think my solution may work two. It increments X while decrementing Y and hits the base case when Y is 0 and X is the first X + the first Y.

Question 11

```
times(X,0,0).
times(X,1,X).
times(X,s(Y),Z) <-- times(X,Y,Z) ^ plus(Z,X,Z)


neq(X,Y) <-- lt(X,Y)
neq(X,Y) <-- lt(Y,X)
```

Question 11 Critique

It looks like I was very close to this solution. I must be making a mistake thinking that recursive calls can take the form Z = Z + X. In the solution Z = XY + X. I can see where this is a mistake since prolog is trying to find a solution the predicate by itself can be true. Therefore my plus(Z,X,Z) statement could only be possible if X was 0. The introduction of the variable XY allows for a storage variable.

Question 12

Did not complete

Question 13

Did not complete

Question 14

Did not complete

Question 15

Did not complete