

CSE 341 — Prolog Discussion Questions

Derivation Trees; Difference Lists; Controlling Search — Answer Key

These questions use the Prolog rules in the lecture notes (both the basics and the ones on controlling search).

1. Draw the derivation tree for the following goals:

```
?- reverse([1],R).
```

Please see the separate scan of the hand-drawn tree. Also try running the goal with the Prolog trace facility.

2. Consider `mymember` and also the `member_cut` rule defined in the notes on controlling search. What are all the answers that Prolog returns for the following goals?

```
?- mymember(1, [A,B,C]).
```

```
A = 1 ;
```

```
B = 1 ;
```

```
C = 1 ;
```

```
false.
```

```
?- member_cut(1, [A,B,C]).
```

```
A = 1.
```

3. What are all the answers that Prolog returns for the following goals?

```
?- mymember(X, [1,2]), mymember(X, [0,2,2]).
```

```
X = 2 ;
```

```
X = 2 ;
```

```
false.
```

(Note that you get the same answer twice!)

```
?- member_cut(X, [1,2]), mymember(X, [0,2,2]).
```

```
false.
```

```
?- mymember(X, [1,2]), member_cut(X, [0,2,2]).
```

```
X = 2 ;
```

```
false.
```

```
?- member_cut(X, [1,2]), member_cut(X, [0,2,2]).
```

```
false.
```

4. What are all the answers that Prolog returns for the following goals?

```
?- not(mymember(1, [1,2,3])).
```

```
false.
```

```
?- not(mymember(5, [1, 2, 3])).
true.
```

```
?- not(mymember(X, [1, 2, 3])).
false.
```

```
?- mymember(X, [1, 2, 3]), not(mymember(X, [1, 2, 4])).
X = 3 ;
false.
```

```
?- not(mymember(X, [1, 2, 4])), mymember(X, [1, 2, 3]).
false.
```

5. Consider the standard version of `append`:

```
append([], Ys, Ys) .
append([X|Xs], Ys, [X|Zs]) :- append(Xs, Ys, Zs) .
```

If you know that the first argument is ground (that is, fully instantiated, containing no variables), there is a more efficient version that you can write by including a cut.

(a) Define such a version.

```
append([], Ys, Ys) :- !.
append([X|Xs], Ys, [X|Zs]) :- append(Xs, Ys, Zs) .
```

(b) Give an example of a query that has exactly the same behavior for both the standard version and the version with a cut.

```
append([1, 2], [3, 4, 5], X) .
```

(c) Give an example of a query that behaves differently for for the standard version and the version with a cut.

```
append(A, B, [1, 2, 3]) .
```

(d) What restrictions do we need on the inputs for the two versions to behave exactly the same? (Is it that the first argument is ground?)

No, it's a little more general: just that the first argument not be a variable.

6. Which of the following lists represent valid difference lists? For valid difference lists, what list do they represent?

```
[1, 2|T]\T          -- valid, represented [1, 2]
[1, 2, 3]\[]        -- valid, represents [1, 2, 3]
[1, 2, 3]\[1, 2]    -- not valid
[1, 2, 3|T]\[3|T]   -- valid, represents [1, 2]
[1, 2, 3]\[1, 2, 3] -- valid, represents []
```

7. Write the list `[squid, clam]` as a difference list (in the most general possible way). Also draw a box-and-arrow diagram of the difference list.

```
[squid, clam|T]\T
```