Andrew Giardina 4/30/2021

Language Study

Homework 8

Problem 1

1. When creating and testing the *Adjacent* program in Prolog, the results for *Adjacent(X, Y)?* was that it did in fact execute, however it provided an infinite loop of answers! Returning alternating X and Y values over and over for a,b & e,f. This is because there is nothing stopping *adjacent* from continually calling itself.

2. My solution, as demonstrated below, is to remove the problematic recursive calling of *adjacent* to itself, and instead offer an additional function to determine whether X & Y are facts in either order. This provides us with the desired output, showcasing all scenarios and then terminating.

adj.pl

```
adj(a, b).
adj(e, f).

/*
adj(X, Y) :-
    adj(Y, X).
*/

/* Having this seperate rule avoids infinite loops    */

isAdj(X, Y) :-  /* Either X & Y will show up in order */
    adj(X, Y).

isAdj(X, Y) :-  /* Or X & Y will be reversed          */
    adj(Y, X).
```

SWI Terminal

```
?- isAdj(X, Y).
X = a,
Y = b ;
X = e,
Y = f ;
X = b,
Y = a ;
X = f,
Y = e.

?-
```

Problem 2

By reversing the order of those append goals within *SubList*, the resulting computation would be an infinite loop. This is because the first goal must be working towards some reachable solution (one that will terminate). In other words, when *SubList(Xs, AsXsBs)* were to run, the first goal:

append(AsXs, Bs, AsXsBs)

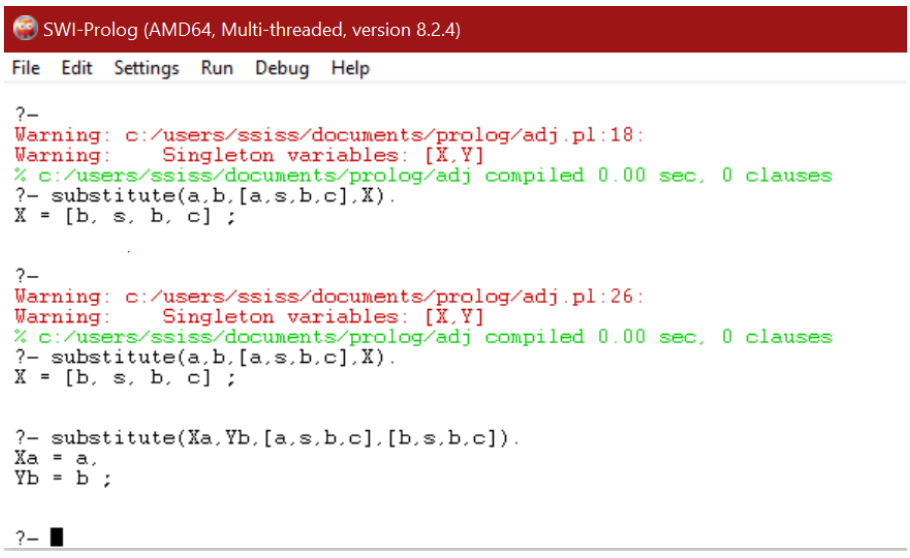
would be actively looking for a solution equal to AsXsBs (which is explicitly passed in through the Sublist parameters), which could be further determined utilizing the second rule (find out what AsXs is). Whereas if we were to reverse the statement order, the first goal:

append(As, Xs, AsXs)

would be trying to compute some AsXs given only Xs, which will just run indefinitely.

Problem 3

1. Reversing the rules, I did not find any varying results. First query is the original order, second and third query when they are reversed. This should be expected, as for this specific predicate, the order of the rules should not matter (both rules imply whether X & Y is equal to what is at the front of the list, so order really doesn't matter).



2. Reversing the goals of the second rule DOES matter.

   On the left shows if the only the goals are reversed.

   On the right shows the goals & rules reversed.

   Significance: We can see the additional computations/effects of the 'later' X \= Z check

SWI-Prolog (AMD64, Multi-threaded, version 8.2.4)

File Edit Settings Run Debug Help

```
?- substitute(A,B,[a,s,b,c],Z).
A = a,
Z = [B, s, b, c] ;
A = s,
Z = [a, B, b, c] ;
A = b,
Z = [a, s, B, c] ;
A = c,
Z = [a, s, b, B] ;

?-
Warning: c:/users/ssiss/documents/prolog/adj.pl:18:
Warning:    Singleton variables: [X,Y]
% c:/users/ssiss/documents/prolog/adj compiled 0.00 sec, 0 clauses
?- substitute(A,B,[a,s,b,c],Z).
A = a,
Z = [B, s, b, c] ;

?-
```

Reversed

Original

SWI-Prolog (AMD64, Multi-threaded, version 8.2.4)

File Edit Settings Run Debug Help

```
?- substitute(A,b,[a,b,a,s,b],Z).
A = a,
Z = [b, b, b, s, b] ;

?- rsubstitute(A,b,[a,b,a,s,b],Z).
A = s,
Z = [a, b, a, b, b] ;
A = b,
Z = [a, b, a, s, b] ;
A = a,
Z = [b, b, b, s, b] ;
```

Original

Reversed

SWI-Prolog (AMD64, Multi-threaded, version 8.2.4)

File Edit Settings Run Debug Help

```
?- substitute(a,b,X,[b]).
X = [a] ;
X = [b] .

?- rsubstitute(a,b,X,[b]).
X = [b] ;
X = [a].

?-
```

Original

Reversed

3. Given for the particular query: *substitute(a,b,X,Y)?* which is asking: Replace all "a" in list X, with "b", resulting in Y. There is an infinite loop for both the reversed and original Substitute, however notice the difference. I believe this is because for the original query, it is computing all resulting list possibilities, which it can achieve and display (although it will last forever). However, for the reversed version, it gets caught up on recursively calling itself before it can compare X to Z (the X & Z variables in the rules, not in the query statement). For both cases, this specific query would result in nontermination, for different reasons!

The problem is that this particular query has infinite solutions

SWI-Prolog (AMD64, Multi-threaded, version 8.2.4)

File Edit Settings Run Debug Help

```
?- substitute(a,b,X,Z).
X = Z, Z = [] ;
X = [a],
Z = [b] ;
X = [a, a],
Z = [b, b] ;
X = [a, a, a],
Z = [b, b, b] ;
X = [a, a, a, a],
Z = [b, b, b, b] ;
X = [a, a, a, a, a],
Z = [b, b, b, b, b] ;
X = [a, a, a, a, a, a],
Z = [b, b, b, b, b, b] .

?- rsubstitute(a,b,X,Z).
X = Z, Z = [] ;
ERROR: Stack limit (1.0Gb) exceeded
ERROR:   Stack sizes: local: 0.7Gb, global: 0.2Gb, trail: 62.0Mb
ERROR:   Stack depth: 4,066,475, last-call: 0%, Choice points: 4,066,468
ERROR:   Possible non-terminating recursion:
ERROR:     [4,066,475] user:rsubstitute(a, b, _48797690, _48797692)
ERROR:     [4,066,474] user:rsubstitute(a, b, [length:1|_48797722], [length:
1])
   Exception: (4,066,474) rsubstitute(a, b, _48797602, _48797608) ? creep
   Exception: (4,066,473) rsubstitute(a, b, _48797590, _48797596) ?
```

Original

Reversed (goals)