CPSC 312

Functional and Logic Programming

29 September 2015

List Processing: append

```
append([],X,X).
append([H|T1],X,[H|T2]) :- append(T1,X,T2)
```

List Processing

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

Try this: list X is a prefix of list Y if Y can be split into two other lists, where X is the first list of those two other lists.

```
prefix(X,Y) :- append(X,_,Y).
```

The other list? It's just any other list that's not X or Y.

Exercise

```
?- suffix([c,d],[a,b,c,d]).

?- sublist([a,b],[c,d,a,b,e]).
```

Exercise

- rewrite the mathematical functions from the Section 3.1 with Arithmetic operators.

iterative vs recursive

- recursive:

```
factorial1(0,1).
factorial1(N,F) :- N1 is N-1,
factorial1(N1,F1), F is N*F1.
```

iterative vs recursive

- recursive:

```
factorial1(0,1).

factorial1(N,F) :- N1 is N-1,
factorial1(N1,F1), F is N*F1.
```

- iterative:

```
factorial3(N,F) :- factorial3(0,N,1,F).

factorial3(N,N,F,F).

factorial3(I,N,T,F) :- I < N, I1 is I+1,
T1 is T*I1, factorial3(I1,N,T1,F).
```

length: four versions

```
length1([X|Xs], s(N)) :- length1(Xs,N).
length1([],0).

length2([],0).
length2([X|Xs],N) :- length2(Xs,N1), succ(N1,N).
```

```
length3([],0).
length3([X|Xs],N) :- length3(Xs,N1), N is N1+1.

length4([],0).
length4([X|Xs], N) :- N>0, N1 is N-1, length4(Xs, N1).

% length4 cant calculate the length of a list but can say yes
or no and can take an unknown list.
% ERROR: arguments not instantiated.
```

recursive vs iterative

- once again the intended use of a program emerges as a most important consideration in choosing the method of implementation

- therefore, that also explains the difference between iterative and recursive; sometimes iterative implementations can respond to queries that recursive ones can't and vice versa. It all depends on the intended use.

recursive vs iterative

- recursive functions space use is linear, but iterative for the same function can be constant. advantage: efficiency

- using accumulators to turn recursive programs into iterative: storing intermediate results.

accumulators: reverse

An accumulator is an added argument, that's not used as input or output and is treated as hidden, that holds a temporary value that's updated in between recursive calls.

```
reverse1([],[]).

reverse1([X|Xs],Zs) :- reverse1(Xs,Ys),append(Ys,
[X],Zs).


reverse2(Xs,Ys) :- reverse2(Xs, [], Ys).

reverse2([X|Xs],Acc,Ys) :- reverse2(Xs, [X|Acc], Ys).

reverse2([],Ys,Ys).
```

which implementation of reverse is more efficient?

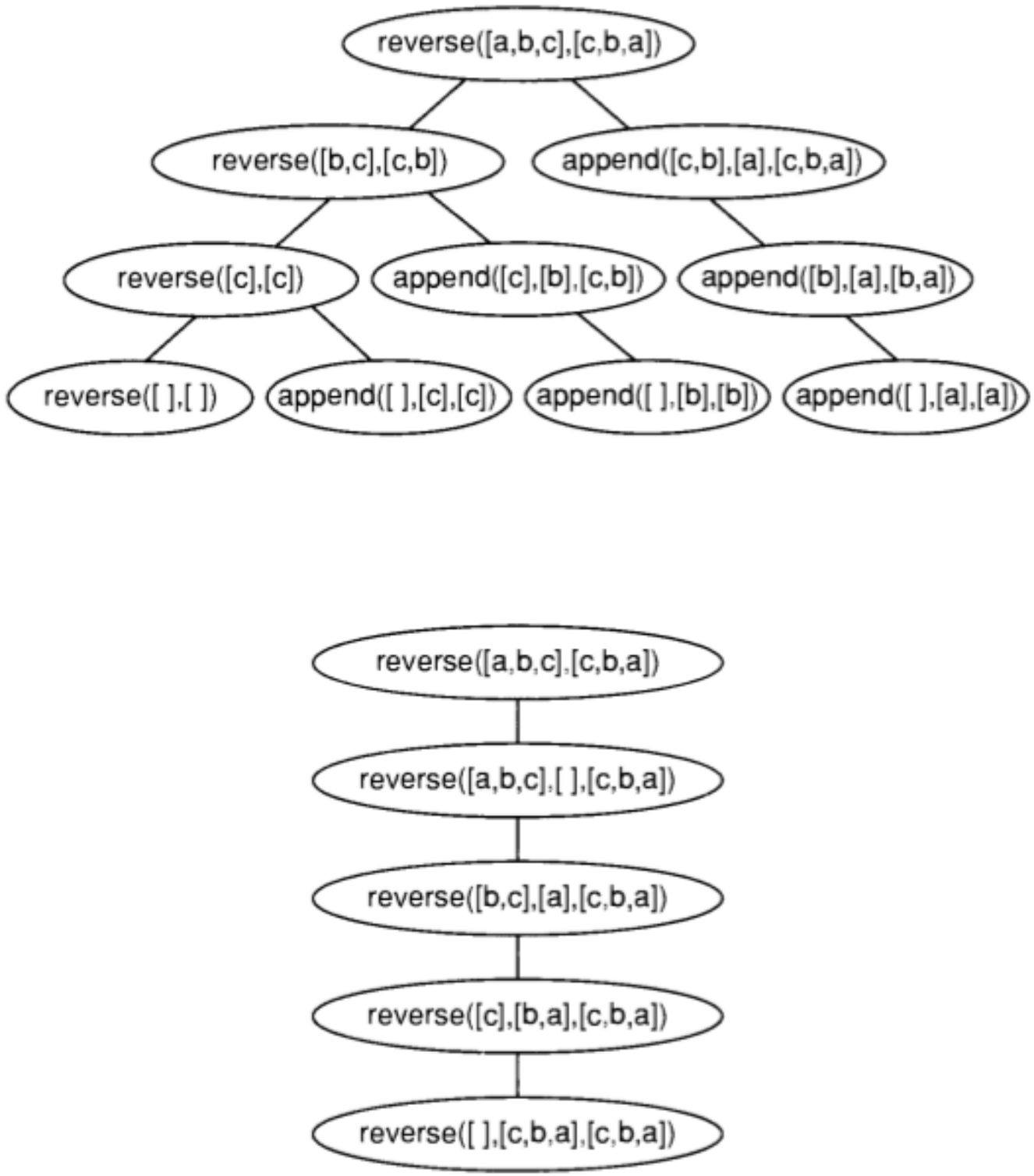

Figure 3.5 Proof trees for reversing a list

Questions