

COMP302: Programming Languages and Paradigms

Assignment Project Exam Help

Prof. Brigitte Pientka (Sec 01)

bpientka@cs.mcgill.ca

Francisco Ferreira (Sec 02)

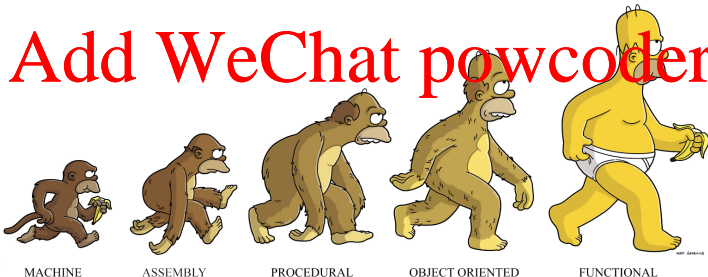
fferre8@cs.mcgill.ca

<https://powcoder.com>

School of Computer Science
McGill University

Week 3-1, Fall 2017

Add WeChat powcoder



Assignment Project Exam Help

How do I prove that all slices of cake are tasty using structural induction?

Step 1. Define a set of cake slices recursively.

<https://powcoder.com>



is cake.

Add WeChat powcoder



are cake, then both of them put together is still cake :



Functional Tidbit: Cake is tasty!

Step 2. Prove that a single piece of cake is tasty.



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Functional Tidbit: Cake is tasty!

Step 2. Prove that a single piece of cake is tasty.



Step 3. Use the recursive definition of the set to prove that all slices are tasty.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Functional Tidbit: Cake is tasty!

Step 2. Prove that a single piece of cake is tasty.



<https://powcoder.com>

Step 3. Use the recursive definition of the set to prove that all slices are tasty.

Add WeChat powcoder

Step 4. Conclude all slices of cake are tasty.



SUPERCOLORINE.COM

Assignment Project Exam Help

<https://powcoder.com>
More on Structural Induction

Add WeChat powcoder

Two programs: Do they compute the same value?

Program A (naive)

```
1 (* rev : 'a list -> 'a list *)  
2 let rec rev l = match l with  
3 | [] -> []  
4 | x::l -> (rev l) @ [x]
```

<https://powcoder.com>

Program B (tail-recursive)

```
1 (* rev : 'a list -> 'a list *)  
2 let rec rev l =  
3   (* rev_tr : 'a list -> 'a list -> 'a list *)  
4   let rec rev_tr l acc = match l with  
5     | [] -> acc  
6     | h::t -> rev_tr t (h::acc)  
7   in  
8   rev_tr l []
```

Two programs: Do they compute the same value?

Program A (naive)

```
1 (* rev : 'a list -> 'a list *)
2 let rec rev l = match l with
3   | [] -> []
4   | x::l -> (rev l) @ [x]
```

Theorem: For all lists l . $\text{rev } l = \text{rev}' l$

Program B (tail-recursive)

```
1 (* rev' : 'a list -> 'a list *)
2 let rev l =
3   (* rev_tr : 'a list -> 'a list -> 'a list *)
4   let rec rev_tr l acc = match l with
5     | [] -> acc
6     | h::t -> rev_tr t (h::acc)
7   in
8   rev_tr l []
```


What to prove? – Finding the invariant!

Program A (naive)

```
1 (* rev : 'a list -> 'a list *)
2 let rec rev l = match l with
3   | [] -> []
4   | x::l -> (rev l) @ [x]
```

What is the relationship between `l acc` and `rev_tr l acc`?

Program B (tail-recursive)

```
1 (* rev' : 'a list -> 'a list *)
2 let rev' l =
3   (* rev_tr : 'a list -> 'a list -> 'a list *)
4   let rec rev_tr l acc = match l with
5     | [] -> acc
6     | h::t -> rev_tr t (h::acc)
7   in
8   rev_tr l []
```

What to prove? – Finding the invariant!

Program A (naive)

```
1 (* rev : 'a list -> 'a list *)
2 let rec rev l = match l with
3   | [] -> []
4   | x::l -> (rev l) @ [x]
```

For all l , acc , $(rev\ l) @ acc \Downarrow v$ and $rev_tr\ l\ acc \Downarrow v$

Program B (tail-recursive)

```
1 (* rev' : 'a list -> 'a list *)
2 let rev' l =
3   (* rev_tr : 'a list -> 'a list -> 'a list *)
4   let rec rev_tr l acc = match l with
5     | [] -> acc
6     | h::t -> rev_tr t (h::acc)
7   in
8   rev_tr l []
```

Theorem: For all l, acc ,
 $length (rev_tr\ l\ acc) \leq v$ and $length\ l + length\ acc \leq v$

We often simply write instead:

For all l, acc ,
 $length (rev_tr\ l\ acc) = length\ l + length\ acc$

Add WeChat powcoder

Can't see the forest for the trees

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Really ?

Inductive definition of a binary tree

- The empty binary tree `Empty` is a binary tree
- If `l` and `r` are binary trees and `v` is a value of type `T`, then `Node(v, l, r)` is a binary tree.
- Nothing else is a binary tree.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Inductive definition of a binary tree

- The empty binary tree `Empty` is a binary tree
- If `l` and `r` are binary trees and `v` is a value of type `T`, then `Node(v, l, r)` is a binary tree.
- Nothing else is a binary tree.

<https://powcoder.com>
How can we define a data type that describes binary trees?

Add WeChat powcoder

Forest and trees

Example of a mutual recursive data type definition:

```
1 type 'a forest = Forest of ('a tree) list  
2 and 'a tree = Empty | Node of 'a * 'a forest
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Remember the slice of cake?

Step 1. Define a set of cake slices
recursively.



is cake.



are cake, then both of them
put together is still cake:



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Give an OCaml data type definition for cake!