Assignment 6

Jon Sporring

11. juli 2019

1 Preface

This document is a response to Assignment 6 on the course Introduction to Functional Programming at UESTC, 2019/7/8-12.

2 Introduction

Assignment 6 includes the code `recursiveMapFoldFilter.fsx`, which uses `List.fold` and `List.filter`. My task is to make my own implementations of these two functions and to verify that they work.

`List.fold` is defined as

```
List.fold : f:('a->'b->'a) -> acc:'a -> lst:'b list -> 'a
```

and for $acc = a$ and $lst = [e_1; e_2; \ldots; e_n]$ computes $f \ldots (f(fae_1)e_2)\dot{e}_n$. For example, a call could be `List.fold (fun acc elm -> acc+elm) 0 [1;2]` which sum 0 with all the elements in the list and give 3.

`List.filter` is defined as

```
List.filter : p:('a->bool) -> lst:'a list -> 'a list
```

and returns the list of elements in `lst` for which `p` returns `true`. For example, a call could be `List.filter (fun elm -> elm%2=0) [0..5]` , which returns `[0; 2; 4]`, i.e., all the even numbers from the list.

3 Problem analysis and design

The course requires us to use the functional programming paradigm and F#, so I must use recursion and no mutable values, for-loops, nor while-loops.

`List.fold` takes a function, an initial accumulator, and a list. I have considered the following case:

The list is empty: I have tested `List.fold`, and it returns the initial accumulator.

The list is non-empty: Here, `List.fold` acts as explained in the introduction.

Error cases: If arguments are missing, then F# gives a syntax error.

`List.filter` takes a function and a list. I have considered the following case:

The list is empty: I have tested `List.filter`. If the type of the empty filter can be determined by the function, then it returns an empty list. Otherwise, it gives an error.

The list is non-empty: Here, `List.filter` acts as explained in the introduction.

Error cases: If arguments are missing, then F# gives a syntax error.

```
// fold f [a1; ...; an] = f ... (f (f acc a1) a2) ...) an
let rec myFold (f: 'b -> 'a -> 'b) (acc: 'b) (lst: 'a list) : 'b =
  // List.fold f acc lst
  match lst with
    | []     -> acc
    | a::rst -> myFold f (f acc a) rst
```

Figur 1: The implemented code for fold.

```
// filter f [a1; ...; an] = [ai; aj; ak ...], where f ai = true etc.,
let rec myFilter (p: 'a -> bool) (lst: 'a list) : 'a list =
  // List.filter p lst
  match lst with
    | a::rst -> if p a then a :: (myFilter p rst) else myFilter p rst
    | []     -> []
```

Figur 2: The implemented code for filter.

4 Program description

The resulting code pieces are so short that we have chosen to include them here in their entirety.

I have used recursion and match-with to implement fold. There are 2 cases: empty and non-empty. For the empty, list we return `acc`, and for the non-empty, we assume that `myFold` works, so I split the list into its first element and the rest and call `myFold` with a new accumulator and the rest. This is shown in Figure 1.

I have also used recursion and match-with to implement filter. It is implemented similarly to fold. For an empty list, an empty list is returned. This will give similar errors as `List.filter`. For a non-empty list, I assume that `myFilter` works, and hence, if `p` is true for the first element, I prepend it to the result of `myFilter` on the rest. Otherwise, I just return the result of `myFilter` on the rest. I have found no way to avoid having two calls to `myFilter p rst`. The code is shown in Figure 2.

5 Testing and experiements

The code `recursiveMapFoldFilter.fsx` includes testing of my code, so I have run the modified program using a long list, which statistically should cover many although not all possible versions. The result is shown in Figure 3 In the figure, the results of running my code is identical to the `List` implementations, so I conclude that my code is working correctly.

6 Discussion and/or Conclusion

I have worked with Assignment 6 and implemented my own versions of fold and filter. I have used recursion and match-statements, and I have tested the code using the supplied test function. My conclusion is that my code is running correctly.

```
$ mono recursiveMapFoldFilter.exe fold 10
Random list is: [9; 5; 9; 6; 7; 6; 1; 7; 2; 9]
Result is CORRECT : [18; 4; 14; 2; 12; 14; 12; 18; 10; 18]
$ mono recursiveMapFoldFilter.exe filter 10
Random list is: [9; 4; 6; 3; 1; 4; 4; 4; 5; 6]
Result is CORRECT : [9; 6; 5; 6]
```

Figur 3: Output from the console of test runs.