Functional Programming – asignment 1

Hamza Bhatti (21223241)

Contents

[Introduction 2](#_Toc443932389)

[Chapter 1 2](#_Toc443932390)

[atom? 2](#_Toc443932391)

[How It Works 2](#_Toc443932392)

[Tracing Execution 3](#_Toc443932393)

[Chapter 2 4](#_Toc443932394)

[lat? 4](#_Toc443932395)

[How It Works 4](#_Toc443932396)

[Tracing Execution 5](#_Toc443932397)

[member? 6](#_Toc443932398)

[How It Works 6](#_Toc443932399)

[Tracing Execution 6](#_Toc443932400)

[Chapter 3 8](#_Toc443932401)

# Introduction

For this assignment the functional programming used was Scheme. This assignment uses the first three chapters of the book “The Little Schemer”. The chapters contain user defined functions. These user defined functions will be accompanied with tracing executions and explanations.

The user define functions will also contain primitive functions, these will be identified after the user defined function is provided in each section.

# Chapter 1

This chapter contains one user defined function. This function is called atom?

## atom?

The function take an argument and then checks to see if the argument is an atom. An atom can be defined as a string of characters. Below is the user defined function:

**(define** atom?

**(lambda** **(**x**)**

**(and** **(not** **(pair?** x**))**

**(not** **(null?** x**)))))**

The function contains some primitive functions. These are $and, $not and $pair? The $and function will evaluate a number of arguments. If both arguments when evaluated are true, the outcome will also be true. If at least one of the arguments is false, the whole outcome will be false. The $not function will invert the result. If an evaluation is false, $not will convert this to true. The primitive pair? Will take an argument and see if it is the same as a pair. For example, a list ‘(and not) will be true as there are two elements within the list.

### How It Works

As stated, the atom? Function will identify if the given argument is an atom. First an argument will be given. Next the evaluation will begin.

The $and primitive is used to compare two expressions. With **(not** **(pair?** x**))**, the outcome should be expected to be false as we would expect just one argument provided. The $not will convert this to true. The next expression is **(not** **(null?** x**))** . We will expect this to be false, as there needs to be something provided to us. This is also converted to true with the $not. As both parts are true, evaluating the results will give us the result true.

### Tracing Execution

Below are some tracing executions of the atom? Function.

First case

; (atom? 'Adam)

; (and (not (pair? 'Adam)) (not (null? 'Adam)))

; (and (not #f)(not #f))

; (and #t #t))

; #t

In the first case, the argument provided is ‘Adam. The second will see that the program will move to evaluating a statement with the $and function. Part of the statement will check if ‘Adam is a pair and it will also check if it is null. It is found that both results are false. Next these will be evaluated with not. The result will give true results. With $and the final result will be true. This shows that ‘Adam is an atom.

Second case

; (atom? '())

; (and (not (pair? '())) (not (null? '())))

; (and (not #f) (not #t))

; (and #t #f)

; #f

In this second case, an empty list is provided. When the list has been evaluated, it is found that the list is not a pair and is null, giving false and true respectively. When converted with not, $and will evaluate true and false. The final result gives false, showing the empty list not an atom.

# Chapter 2

This chapter contains two user defined functions.

## lat?

The lat? Function is used to determine if all the elements in a list are all atoms. In this function, we need to use our user defined atom? Function too. Below is the lat? Function.

**(define** lat?

**(lambda** **(**l**)**

**(cond**

**[(null?** l**)** #t**]**

**[(**atom? **(car** l**))** **(**lat? **(cdr** l**))]**

**(else** #f**))))**

Like the atom? From the last chapter, lat? Also has some primitive functions. The new functions seen here are $car and $cdr. With the $car primitive, it will return the first element from a list. The $cdr primitive will return the all the elements after the first as a list.

### How It Works

lat? Will check to see if a given list contains only atoms, if this is true, the result will be true. A feature of this function is the conditional statement. There are different statements that will be used to determine if something is true or false. If at a point the statement is found to be false, the next statement will be used, and so on.

The first condition will check if the list is null. If the list is null, the function will terminate and give the result of true. If this is not true, the next statement will be investigated.

In the second statement, it will start by checking if the first element of the list is an atom with the atom? Function along with the $car of the list. If the car is true, it will continue and call upon the function. When it calls itself, it will take the cdr (remainder of the list) as a new argument and start over. If when the atom? Function returns a false, the function will move to the else statement and return a false. This could happen if the car of l is a nested list.

The result will keep checking the car and cdr of the list. It will end when the list is empty and return a true value when the list s null.

### Tracing Execution

Below are some tracing executions for the lat? Function.

First case

; (lat? '(Andrew Ben Charlie))

; (#t (lat? '(Ben Charlie)))

; (#t (#t (lat? '(Charlie))))

; (#t (#t (#t (lat? '()))))

; #t

In this case, a list is provided. It is populated with three elements. As the list is not null, it will carry onto the next statement. First it will check if the $car of the list is an atom. The first time this occurs, the element ‘Andrew is the car of the list and is an atom. In the same condition, it will then call the function itself and provide the $cdr of the list, which is the remaining elements.

As can be seen from the trace, all elements are atoms. When lat continues and has an empty list as the argument, it will trigger the first conditional statement and return true. This is because at this point, all the elements of the list will have been identified as atoms.

Second case

; (lat? '(Andrew (Ben) Charlie))

; (#t (lat? '((Ben) Charlie)))

; #f

In this case a list with a nested list is provided as an argument. The list is not null, so it will move on to the next conditional statement. The $car of the list (Andrew) is an atom, so the function will continue and call upon the function with the $cdr of the list as the argument. The list is still not null. The car of the list is not an atom. This is because at this stage, it will be ‘(Ben) which in fact is a list and not an atom. The function will move to the else statement and return false as the final answer.

## member?

The member? User defined functions aims to check if an argument is present in a list. Below is the function.

**(define** member?

**(lambda** **(**a lat**)**

**(cond**

**[(null?** lat**)** #f**]**

**(else** **(or** **(eq?** **(car** lat**)** a**)**

**(**member? a **(cdr** lat**)))))))**

The member? Function contains two new primitive functions. One is the $or. This will evaluate two elements. If at least of the elements is true, the result will be true. Another primitive is the $eq. This will return true if values are equal.

### How It Works

The functions takes two arguments one will be an element and the second will be a list of elements. The function will start off with a conditional statement. The first condition will check to see if the provided list is null. If the list is null, the function will return false. If the list is not null, the next statement will be initiated.

The next statement will check to see if the $car of the list and the element provided are equal. Even if this is false, the statement still be carried out. It will then take the $cdr of the list and the element provided.

The function will carry on until either the car of the list and the provided element are equal, or the list is found to be null.

### Tracing Execution

Below are some examples of how the member? Function works with some traces.

First case

; (member? 'Joe '(Hamza Amrit Joe Tashan))

; (member? 'Joe '(Amrit Joe Tashan))

; (member? 'Joe '(Joe Tashan))

; #t

The arguments provided are ‘Joe and the list ‘(Hamza Amrit Joe and Tashan). First the element ‘Joe will be compared with the $car of the list. As these are not equal, the function will move on and call itself. It continues on calling itself along with the $cdr of the list. When the else statement is reached again, the car of the list and the element are equal. Because this is true, the function will return true and terminate.

Second case

; (member? 'Joe '(Hamza Amrit Tashan))

; (member? 'Joe '(Amrit Tashan))

; (member? 'Joe '(Tashan))

; (member? 'Joe '())

; #f

In the second case, the two arguments are ‘Joe and a list ‘(Hamza Amrit Tashan). When the function is active, it eventually calls itself with the argument ‘Joe and the $cdr of the list. At the late stage the $cdr of the list is empty. Because of this, the function will terminate because the list is null. The function will return the result false.

# Chapter 3