Brandon Cao

CIS 425

Spring 16’

Assignment 4

1)

a)

Function maptree f calls the parameter function on any leaf ‘a. It then recursively calls the parameter function on any nodes. Function maptree will map a tree to a tree by mapping the values to new values.

Fun maptree f = nil = nil

maptree f( LEAF ‘a) = LEAF( f ‘a)

maptree f(NODE tl tr) = NODE (maptree f tl, maptree f tr);

b)

ML gives my function type tree t -> tree t. It does not give the expected type “(’a → ’a) → ’a tree → ’a” because the function maps to two nodes.

2)

fun reduce oper (LEAF ‘a) = LEAF (oper ‘a)

reduce oper(NODE tl tr) = NODE(reduce oper tl, reduce oper tr);

3)

a)

To compute the sum of the first 5 squared numbers, you use the reduce function and pass in two parameters. The first parameter will be the operation you will be performing on the second parameter which is a list. The first parameter will be an anonymous function that takes in two parameters and adds them together. The second parameter of the reduce function will be calling the map function, which squares the number in each item in the list that is passed into the map function. The reduce function will then perform it’s anonymous function defined in the first parameter on the list that is returned by the map function, which is a list of the squared numbers.

reduce (fn(x,y) => x+y) (map (fn x => x\*x) [1,2,3,4,5]);

b)

To find the number of positive integers in the list, we assign a variable to the list that is returned from the map function which creates a list that contains a value of 1 or 0. This means that the integer in the list that is passed into the map function is a positive value or negative value at that index, respectively. We use the reduce function passing an anonoymous function as the first parameter that will add the values in the list. The second parameter will be the list that was returned from the map function. The integer returned will be the number of 1’s in the list, which represents if the values from the original list was a positive or negative number.

val list = map (fn(x) => if x <=0 then 0 else 1) [1,~2,3,~4,5];

reduce (fn(x,y) => x+y) list;

c)

We use the reduce function and flatten x and y in the array.

val flatten = reduce ( fn(x,y) => x @ y) [[1,2],[3,4],[5,6]];

4)

a)

Function Curry takes in the function f : (‘a \* ‘b) -> c and returns a curried function g: ‘a -> ‘b -> ‘c. g xy = f(x,y)

fun Curry f = fn x => fn y => f(x,y);

UnCurry does the opposite of the curry function by taking in a function like f and returning a function like g.

fun UnCurry g = fn (x,y) => g x y;

5)

a)

The addition may not work because the union type is neither an int or a char but variables i is an int and \*s is of type char. The addition may also not work because when x is evaluated, the variable s is a pointer to a string value that would be interpreted as an integer. Because of these two reasons, the errors wouldn’t be caught during the program.

b)

The compiler will show a warning message saying “Warning: match non-exhaustive” because the tag\_int does not provide a pattern in its declation to match the input of tag\_str. This error should inform you to account for all possible data types in the domain.

6)

a)

fun merge( [ ] , x) = x

|  merge(y, [ ]) = y

|  merge(a::y, b::x) = if a < b then a::merge(y, b::x) else b::merge(a::y, x);

* val merge = fn : int list \* int list -> int list

b)

* val compose = fn (f, g) => fn h => f(g h);