SML (3): More on SML

Comp3031 Lab 03
Fall 2013

JIA Xiaoying, Su Zhiyang

Higher order functions

- Function that output a function or take one or more functions as an input
- Example:
  - fun double (x:int): int = 2 * x;
  - fun square (x:int): int = x * x;
- quadruple/fourth a number
  - fun quad (x:int) :int = double(double (x));
  - fun fourth(x:int) :int = square(square(x));

Higher order functions

- fun applyTwice (f:int->int,x:int) :int = f(f(x));

- fun quad(x:int):int = applyTwice(double,x);
- fun fourth(x:int): int = applyTwice(square,x);

Higher Order Functions on Lists

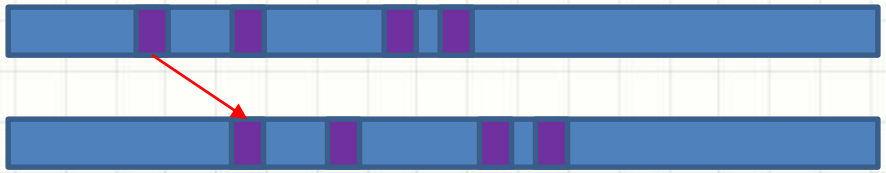
- map
  - fun map f [ ] = [ ]
  - | map f (head::tail) = (f head)::(map f tail);
- filter
  - fun filter f [ ] = [ ]
  - | filter f (head::tail) = if (f head)
  - then head::(filter f tail)
  - else (filter f tail);

Higher Order Functions on Lists

- reduce
  - fun reduce f [ ] v = v
  - | reduce f (head::tail) v = f (head, reduce f tail v);

Higher Order Functions on Lists

- map example:
  - fun add2 x = x+2;
  - map add2 [2,4,7,8];
  - val it = [4,6,9,10] : int list

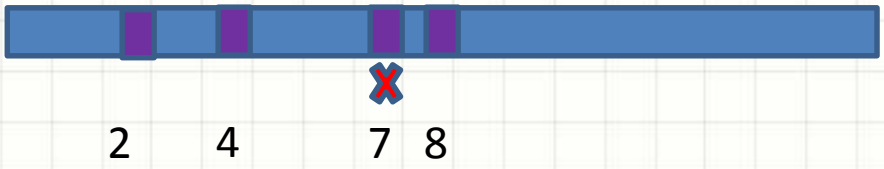


Higher Order Functions on Lists

- map example:
  - fun valueConvert “D” = 10
    - | valueConvert “S” = 0;
  - map valueConvert [“D”,“S”,”D”,”S”];
  - val it = [10,0,10,0] : int list

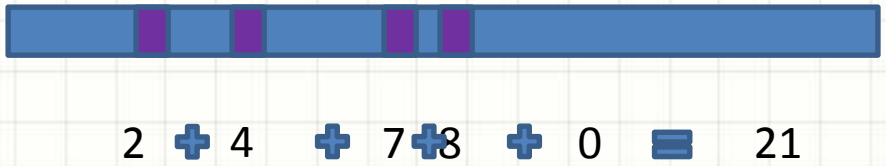
Higher Order Functions on Lists

- filter example
  - fun even x = (x mod 2) = 0;
  - filter even [2,4,7,8];



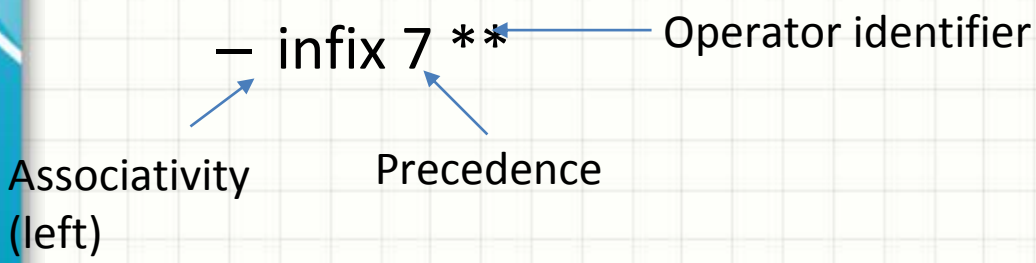
Higher Order Functions on Lists

- reduce example
  - fun add (x,y) = x+y;
  - reduce add [2,4,7,8] 0;
  - val it = 21 : int



User defined operators

- First define a function, then declare it to be an infix operator
  - fun **(a,0) = 1 | **(a,b) = a * **(a,b-1);
  - val ** = fn : int * int -> int
  - infix 7 **;
  - infix 7 **



User defined operators

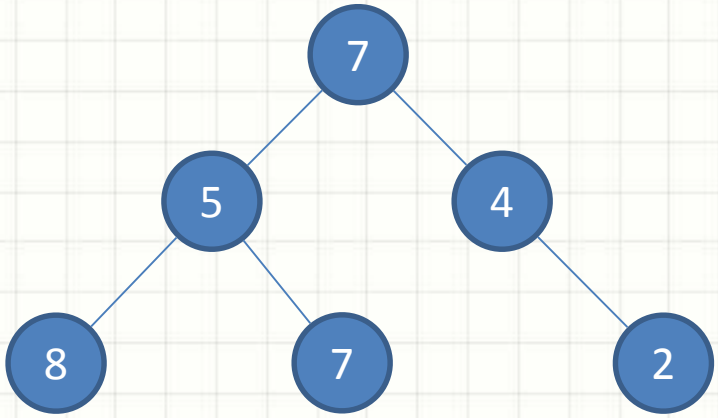
- Example:
  - 2**2**3;
  - val it = 64 : int
- If we change infix into infixr, what will be the result? Try it.
  - infixr 7 **;
  - 2**2**3;

User-defined Data Types

- Define new structure of data, like struct in C
  - datatype time = day | night; ← Constant constructor
- Functions can be defined on the new datatype
  - fun love day = true | love night = true;
- Notice that every constructor should be dealt with in a function
  - fun love day = true;
  - Warning: match nonexhaustive

New datatypes

- datatype tree = nil | node of int * tree * tree;
- This datatype can represent any binary tree of int. How to represent the following tree?



Exercise

- Write a function poTrav = fn : Tree -> int list to return a list of integers representing the tree nodes traversed in the post-order. (See datatype tree on page 12)

- Write a function dCountTree = fn : Tree -> int to return the total number of unique integers representing the tree nodes. Try to use reduce.