CS603 Programming Language Organization

Lecture 22
Spring 2004
Department of Computer Science
University of Alabama

Objectives

- Provide you with some familiarity with ML. In particular, you should be able to:
  - Know how to debug type errors
  - Write some programs

Bonus!

- Higher Order Functions!
- ```
  - fun twice f x = f (f x);
  > val twice = fn : ('a -> 'a) -> 'a
  -> 'a
  - fun inc n = n + 1;
  > val inc = fn : int -> int
  - twice inc 20;
  > val it = 22 : int
  -
  ```
  ```

One more thing....

- - use "myprogram.sml";
- Immediate mode is fun, but it can be counter-productive....

Recursion in ML

- In functional languages, repetition is accomplished by recursion.
- ```
  fun gcd m n =
      if m = n then m
      else if m < n then gcd m (n % m)
      else gcd n m;
  ```
  ```

Lists

- Recursive data structure:
  - A □ list (a list whose elements are of type □ is either empty or a □ joined to a □ list.
- If L = x joined to M, then x is the head of L and M is the tail of L

Lists in ML

- Constants:
  - `[1,2,3]: int list`
  - `[true, false]: bool list`
- Operations:
  - `hd [1,2,3] = 1`
  - `tl [1,2,3] = [2,3]`
  - `null [] = true`
  - `null [1,2] = false`
  - `1::[2,3] = [1,2,3]`

Types of ML lists and operations

- Lists can contain other lists, but are homogeneous.
  - `[[1,2], [], [4,5,2]]: (int list) list`
- But `[1, [2,3]]` is not legal.
- List operations have polymorphic types:
  - hd: 'a list -> 'a
  - tl: 'a list -> 'a list
  - :: : 'a * 'a list -> 'a list
  - null: 'a list -> bool

Simple examples

- `fun tltl L =` list with first 2 elements removed
  `fun hdtl L =` 2nd element of list
  `fun incrhd L =` list with first element incremented
  `fun swaphd L =` list with first 2 elements reversed

Pair up:
Write each of the above functions

Simple examples

- ```
  fun tltl L = (tl (tl L));
  fun hdtl L = hd (tl L);
  fun incrhd L = (1+(hd L))::(tl L);
  fun swaphd L =
      (hdtl L) :: (hd L) :: (tltl L);
  fun length L = if null L then 0
             else 1+(length (tl L));
  fun append L M = if null L then M
     else (hd L)::(append (tl L) M);
  fun concat L M = if null L then L
       else (append (hd L) M);
  ```
  ```

Pattern-matching in function definitions

- ```
  fun f [] = ...
    |   f (x::xs) = ...x...(f xs)...
  fun f [] M = ...M...
    |   f L [] = ...L...
    |   f (x::xs) (y::ys) =
                ...x...y...(f xs ys)...
  fun f [] = ...
    |   f [x] = ...x...
    |   f (x::y::xs) =
                ...x...y...(f (y::xs))...
  ```
  ```

More code

```
fun prod L = if null L then 1 else (hd L) * (prod (tl L));

fun prod [] = 1
  | prod (x::xs) = x * prod xs;

fun length [] = 0
  | length (x::xs) = 1 + length xs;

local
    fun addlen (n, []) = n
      | addlen (n, x::xs) = addlen (n+1, xs)
in
    fun length l = addlen (0,l)
end;
```

Example: merge sort

```
fun msort L =
let val halves = split L
in merge (msort (hd halves))
         (msort (hdtl halves))
end

fun split [] = [[], []]
  | split [a] = [[a], []]
  | split (a::b::t) =
      let val splittltl = split t
      in [a::(hd splittltl),
          b::(hdtl splittltl)]
      end;
```

Pair up:
Write merge

merge

- ```
  fun merge ([], ys)     = ys
      merge (xs, [])     = xs
      merge (x::xs, y::ys) =
        if x <= y then x::merge(xs, y::ys)
                  else y::merge(x::xs, ys);
  ```
  ```

Algebraic Data-types

- ```
  datatype 'a tree = Empty
                   | Node of 'a tree *
                             'a *
                             'a tree

  fun height Empty = 0
    | height (Node (lft, _, rht)) =
          1 + max (height lft, height rht)
  ```
  ```

Resources

- "Programming in Standard ML"
  - http://www-2.cs.cmu.edu/~rwh/smlbook/offline.pdf
- Source code from above book
  - http://www-2.cs.cmu.edu/~rwh/smlbook/examples/
- Moscow ML
  - http://www.dina.dk/~setsoft/mosml.html