CSCI 334: Principles of Programming Languages

Lecture 12: ML and F#

Instructor: Dan Barowy Williams Announcements

Lab machines: see email for dotnet fix

Also, clarified zip3 example in HW6 PDF

Compound Types:

Records, Lists, Tuples, ADTs

Records

### Records

• Like tuple, but with labeled elements:

```
> type Point = { X: float; Y: float; Z:
float; }
> let mypoint = { X = 1.0; Y = 1.0; Z =
-1.0 };
```

• Selector operator:

```
> mypoint.X;;
val it : float = 1.0
- mypoint.Z;
val it : float = -1.0
```



### Lists

### • Examples

- [1; 2; 3; 4], ["wombat"; "numbat"]
- [] is empty list
- all elements of list must be same type
- Operations
  - -length length  $[1;2;3] \Rightarrow 3$
  - append  $[1;2]@[3;4] \Rightarrow [1; 2; 3; 4]$
  - cons  $1::[2;3] \Rightarrow [1; 2; 3]$
  - map List.map succ  $[1;2;3] \Rightarrow [2;3;4]$

# Many Types Of Lists

```
•1::2::[] : int list
"wombat"::"numbat"::[] : string list
```

• What type of list is []? - [];

- Polymorphic type
  - 'a is a type variable that represents any type
  - -1::[] : int list
  - "a"::[] : string list

### Lists

```
• Functions on Lists (usually recursive)
```

```
> let rec product nums =
    if (nums = [])
        then 1
        else
        (List.head nums)
        * product(List.tail nums);;
val product : int list -> int
- product [5; 2; 3];;
val it : int = 30
```



# pattern matching

A pattern is built from

•values,

- constructors,
- and variables
- Tests whether value(s) have shape defined by pattern
- If matches, binds variable(s) in pattern to value(s)

## Pattern Matching on Integers

```
• Patterns on integers
let rec listInts n =
   match n with
   | 0 -> [0]
   | n -> n :: listInts (n-1);;
   > listInts 3;;
```

> listInts 3;; val it : int list = [3; 2; 1; 0]

• Let's flip this list around

```
Provide the second state of the second st
```

## Pattern Matching on Lists

```
•List is one of two things:
- []
- "first elem" :: "rest of elems"
- E.g., [1; 2; 3] = 1::[2,3] = 1::2::[3]
= 1::2::3::[]
```

• Can define function by cases

```
let rec product xs =
  match xs with
    [] -> 1
    | x::xs -> x * product (xs);;
```

# The Length Function

```
• Another Example
```

```
let rec length xs =
  match xs with
    [] -> 0
    | x::xs -> 1 + length xs;;
```

```
• What is the type of length?
```

### Pattern Matching on Tuples

```
let rec cartesianProduct xs ys =
  match xs,ys with
        [],______> []
        [_,[]] __> []
        [ x::xs',____>
        let zs = List.map (fun y -> (x,y)) ys
        zs @ cartesianProduct xs' ys
```

# Patterns and Other Declarations • Patterns can be used in place of variables • Most basic pattern form - let <pattern> = <exp>; • Examples - let x = 3;; - let tuple = ("moo", "cow");; - let (x,y) = tuple;; - let myList = [1; 2; 3];; - let w::rest = myList;; - let v::\_ = myList;;

# Activity

Write a function is\_older that takes two dates (where a date is int\*int\*int) and returns true or false. It evaluates to true if and only if the first argument is a date that comes before the second argument. If the two dates are the same, return false.

E.g.,

is\_older (2018,2,21) (2018,2,22) returns true

# Activity

Write a function number\_in\_month that takes a list of dates
 (where a date is int\*int\*int) and an int month and
 returns how many dates are in month