ML – Functions

Read Chap 5

Syntax:

<fdef> ::= **fun** <function-name> <parameter> **=** <expression>
<function-name> ::= an appropriate function name
<parameter> ::= a variable name
| a tuple of variable names
<expression> ::= any expression discussed so far including function calls

Example (default type int):

```
- fun add2 (a,b) = a + b;
val add2 = fn : int * int -> int
```

tuple

function type

Example (real):

function return type

```
- fun add2r (a:real,b:real):real = a + b;
val add2r = fn : real * real -> real
```

ML – Functions

Using Functions:

```
- fun add2 (a,b) = a + b;
val add2 = fn : int * int -> int
- add2(1,2);
val it = 3 : int
```

ML – Functions

Example: write a function that computes the negative of a real value.

```
- fun neg(v:real):real = v * ~1.0;
val neg = fn : real -> real
```

ML – Functions

Example: write a function that computes the sum of the elements of a list of integers. (Hint: use recursion)

Cases:
[1,2,3,4,5] → 15
[6,12] → 18
[7] → 7
[ ] → 0

```
- fun listsum (L:int list) = if null(L) then 0
                              else hd(L) + listsum(tl(L));
```

recursion!

ML – Functions

Example: write a function that computes the length of a list.

Cases:
[1,2,3] $\rightarrow$ 3
[1,2] $\rightarrow$ 2
[1] $\rightarrow$ 1
[ ] $\rightarrow$ 0

```
- fun listlength(l) = if null(l) then 0
                      else 1 + listlength(tl(l));
```

recursion!

ML – Functions

Example: write a function that will reverse the order of the elements in a list.

Cases:
[1,2,3] → [3,2,1]
[1] → [1]
[ ] → [ ]

```
- fun reverse (L) = if null(L) then [ ]
                    else reverse(tl(L)) @ [hd(L)];
```

ML – Functions

- Write a function *red3* of type *'a * 'b * 'c -> 'a * 'c* that converts a tuple with three elements into one with two by eliminating the second element.
- Write a function *thirds* of type *string -> char* that returns the third character of a string (your function does not need to be defined on strings with length less than 3), Hint: use the *explode function*.
- Write a function *del3* of type *'a list -> 'a list* whose output list is the same as the input list, but with the third element deleted (your function does not need to be defined on lists with length less than 3).
- Write a function *sqsum* of type *int -> int* that takes a non-negative integer *n* and returns the sum of the squares of all the integers *0* through *n* (your function does not need to be defined for inputs < 0).
- Write a function *sort3* of type *real * real * real -> real list* that returns a sorted list of three numbers.
- Write a function *pow* of type *real * int -> real* that raises a real number to an integer power (your function does not need to be defined for integer values less than 0)

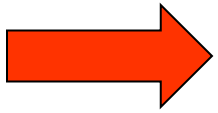
ML – Functions

- Some pragmatics for ML:
  - You can use your favorite editor to write a program and then load it into ML with
    - use “<filename>”;
  - Comments in ML: (* … *)

sq.sml

```
(*
 * this program computes the
 * square of a real value
 *)

fun sq (x:real):real = x * x;
```



```
iBook:~ lutz$ sml
Standard ML of New Jersey v110.59
- use "sq.sml";
[opening sq.sml]
val sq = fn : real -> real
val it = () : unit
- sq(3.0);
val it = 9.0 : real
-
```

ML – Functions

- Assignment #4 – see website