

## An Introduction (I) to SML, Java & Prolog

CS216

1

## SML

CS216

2

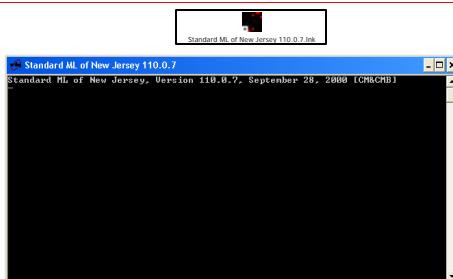
### Function-Oriented Style

- Functions
- Function applications
- Expressions
- Solving problems using functions!
- There are a number of dialects
  - Standard ML-NJ

CS216

3

### SML-NJ Environment



CS216

4

### SML-NJ Environment

A screenshot of the SML-NJ environment showing a terminal window. The window title is "Standard ML of New Jersey 110.0.7". The text in the window shows a computation: "1+2\*3; val it = 7 : int - 1+2 \*3 = ; val it = ? : int - ". Below the window, several lines of text are highlighted in red, explaining the meaning of the symbols used in the prompt.

CS216

5

### Primitive Data Types: Integer, Real, Boolean

```
-1234;  
val it = 1234 : int  
- 123.4;  
val it = 123.4 : real  
- true;  
val it = true : bool  
- false;  
val it = false : bool
```

CS216

6

## Primitive Data Types: String, Character

```
- "fred";
val it = "fred" : string
- "H";
val it = "H" : string
- #"H";
val it = #"H" : char
```

CS216

7

## Operators

```
- ~ 1 + 2 - 3 * 4 div 5 mod 6;
val it = ~1 : int
- ~ 1.0 + 2.0 - 3.0 * 4.0 / 5.0;
val it = ~1.4 : real
```

Left associative, precedence is  $\{+,-\} < \{\ast,\text{/},\text{div},\text{mod}\} < \{\sim\}$ .

CS216

8

## Operators

```
- "bibity" ^ "bobity" ^ "boo";
val it = "bibitybobityboo" : string
- 2 < 3;
val it = true : bool
- 1.0 <= 1.0;
val it = true : bool
- 1 = 2;
val it = false : bool
- true <> false;
val it = true : bool
```

CS216

9

## Conditional Expression

```
- if 1 < 2 then #'x" else #'y";
val it = #'x" : char
- if 1 > 2 then 34 else 56;
val it = 56 : int
- (if 1 < 2 then 34 else 56) + 1;
val it = 35 : int
```

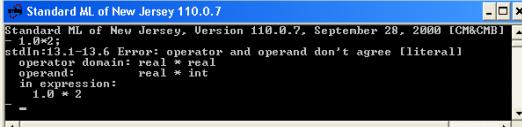
CS216

10

## Types

```
- 1 * 2;
val it = 2 : int
- 1.0 * 2.0;
val it = 2.0 : real
- 1.0 * 2;
???

```



The screenshot shows a window titled 'Standard ML of New Jersey 110.0.7'. It displays the command '- 1.0 \* 2;' followed by an error message: 'operator and operand don't agree (literal)'. Below the message, it says 'expected domain: real \* real' and 'operand: real \* int'. At the bottom, it shows 'in expression: 1.0 \* 2'.

ML does not perform implicit type conversion  
CS216

11

## Type Conversion Functions

```
- real(123);
val it = 123.0 : real
- floor(3.6);
val it = 3 : int
- floor 3.6;
val it = 3 : int
- str #'a';
val it = "a" : string
```

CS216

12

## Variables

```
- val x = 1+2*3;
val x = 7 : int
-x;
val it = 7 : int
-val y = if x = 7 then 1.0 else 2.0;
val y = 1.0 : real
```

CS216

13

## Local Variables

```
-let val x = 1 in x+2*3 end;
val it = 7 : int
-x;
???
- let val x=1
=val y=2
=in
=x+y
=end;
val it = 3 : int
```

CS216

14

## Functions

```
-fun square x = x*x;
val square = fn : int -> int
```

```
<fun-def> ::= 
  fun <function-name>
    <parameter> =
      <expression> ;
```

CS216

15

## Function Application

```
- square 2;
val it = 4 : int
-square 2+1;
val it = 5 : int
-square (2+1);
val it = 9 : int
```

Function application has higher precedence than any operator.

CS216

16

## Functions

```
-fun square x = x*x;
val square = fn : int -> int
-square 2;
val it = 4 : int
-val s = fn x => x*x;
val square = fn : int -> int
-s 2;
val it = 4 : int
```

CS216

17

## Functions

```
-fun quot(a,b) = a div b;
val quot = fn : int * int -> int
- quot (6,2);
val it = 3 : int
- val pair = (6,2);
val pair = (6,2) : int * int
- quot pair;
val it = 3 : int
```

All ML functions take exactly one parameter.

To pass more than one thing, you can pass a tuple.

CS216

18

## Example – The Factorial Function

```
-fun fact n =
=  if n = 0 then 1
=  else n * fact(n-1);
val fact = fn : int -> int
- fact 5;
val it = 120 : int

- fun fact (n: int): int =
=  if n = 0 then 1
=  else n * fact(n-1);
val fact = fn : int -> int
```

CS216

19

## Tuples

```
- (1, 2);
val it = (1,2) : int * int
- #1 (1, 2);
val it = 1 : int
- val point1 = ("red", (300,200));
val point1 = ("red",(300,200)) : string *
(int * int)
- #1 (#2 point1);
val it = 300 : int
```

A tuple is like a record with no field names.

CS216

20

## Lists

```
- [1,2,3];
val it = [1,2,3] : int list
- [1.0,2.0];
val it = [1.0,2.0] : real list
- [true];
val it = [true] : bool list
- [(1,2),(1,3)];
val it = [(1,2),(1,3)] : (int * int) list
- [[1,2,3],[1,2]];
val it = [[1,2,3],[1,2]] : int list list
```

Unlike tuples, all elements of a list must be the same type.

CS216

21

## Lists

```
- [];
val it = [] : 'a list
- nil;
val it = [] : 'a list
-null [];
val it = true : bool
- null [1,2,3];
val it = false : bool
-nil = [];
val it = true : bool
```

Empty list is [] or nil.

CS216

22

## List Operations - ::

```
- 1 :: [2,3];
val it = [1,2,3] : int list
- op ::;
val it = fn: 'a * 'a list -> 'a list
-val z = 1::2::3::[];
val z = [1,2,3] : int list
- [1,2] :: [3,4];
-???
```

List-builder (*cons*) operator is ::

The :: operator is right-associative.

CS216

23

## List Operations

```
- hd z;
val it = 1 : int
- tl z;
val it = [2,3] : int list
- tl(tl z);
val it = [3] : int list
- tl(tl(tl z));
val it = [] : int list
```

The hd function gets the head of a list: the first element.

The tl function gets the tail of a list: the whole list after the first element.

CS216

24

## List Operations

```
[* Standard ML of New Jersey 110.0.7
Standard ML of New Jersey, Version 110.0.7, September 28, 2000 (CM&CMB)
- val it = [] : 'a list
- nil;
val it = [] : 'a list
- null [];
val it = true : bool
- nil [];
val it = true : bool
- nil [];
val it = [1];
val it = [1,2];
val it = [1,2,3];
val it = [1,2,3] : int list
- op ::;
val it = fn : 'a * 'a list -> 'a list
- val l = [1,2,3];
val l = [1,2,3] : int list
- GC #0.0.0.0.1.16: <0 ms>
- hd l
= i;
val it = 1 : int
- cl l;
val it = [2,3] : int list
[1]
CS216
25]
```

## List Operations

```
-[1,2,3]@[4,5,6];
val it = [1,2,3,4,5,6] : int list
- op @;
val it = fn: 'a list * 'a list -> 'a
list
- 1 @ [2,3];
???
```

The @ operator concatenates lists.

Operands are two lists of the same type.

Note: 1@[2,3,4] is wrong: either use [1]@[2,3,4] or 1:::[2,3,4]

26

## List Operations

```
[* Standard ML of New Jersey 110.0.7
Standard ML of New Jersey, Version 110.0.7, September 28, 2000 (CM&CMB)
- 1@[2,3];
val it = [1,2,3] : int list
- null [];
stdIn:14.1-14.15 Error: operator and operand don't agree [literal]
operator domain: int list * int list
operand: int list * int list
in expression:
  1@[2,3];
stdIn:14.1-14.15 Error: operator and operand don't agree [literal]
operator domain: 'Z list * 'Z list
operand: int * int list
in expression:
  1 @ 2 :: 3 :: nil
[1]
CS216
27]
```

Note: 1@[2,3,4] is wrong: either use [1]@[2,3,4] or 1:::[2,3,4]

CS216

27

## List Functions

```
-fun length x =
=  if null x then 0
=  else 1 + length (tl x);
val length = fn : 'a list -> int

-length [true,false,true];
val it = 3 : int

-length [4.0,3.0,2.0,1.0];
val it = 4 : int
```

28

## List Functions

```
-fun listsum x =
=  if null x then 0
=  else hd x + listsum(tl x);
val listsum = fn : int list -> int

-listsum [1,2,3,4,5];
val it = 15 : int
```

CS216

29

## List Functions

```
-fun reverse L =
=  if null L then nil
=  else reverse(tl L) @ [hd L];
val reverse = fn : 'a list -> 'a list

-reverse [1,2,3];
val it = [3,2,1] : int list
```

30

## Garbage Collection

- GC #0.0.0.0.1.3: (0 ms)

CS216

31

## Use

```
Standard ML of New Jersey 110.0.7
Standard ML of New Jersey, Version 110.0.7, September 28, 2000 (CM&CMB)
- op use;
val it = fn : string -> unit
-

```

CS216

32

## Use

```
-op use;
val it = fn: string -> unit

-use ("filename.sml");
```

factorial.sml:

```
(* COMMENTS factorial function *)
fun fact n = if n = 0 then 1 else n * fact(n-1);

```

CS216

33

## Use

```
es Command Prompt - sml
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\Administrator>cd C:\Documents and Settings\Administrat
or\My Documents\YOUNGTEACHING\Spring2004\BU-CS216_SP04\SML
C:\Documents and Settings\Administrator>cd C:\Documents and Settings\Administrat
or\My Documents\YOUNGTEACHING\Spring2004\BU-CS216_SP04\SML>sml
Standard ML of New Jersey, Version 110.0.7, September 28, 2000 (CM&CMB)
-use ("factorial.sml");
[Loading "factorial.sml"]
val fact = fn : int -> int
val it = () : unit
val fact = fn : int -> int
val it = 120 : int
val fact = fn : int -> int
val it = 6 : int
-
```

CS216

34