A First Look at ML

Adrien Protzel

Exercise 1: Summarize

1. Meta Language (ML) is a functional programming language with a strong type system and straightforward syntax and semantics, making it a preferred choice for both programmers and mathematicians.

Exercise 2: Read & Code

1. SOSML code attached in zip file.

1+2\*3; (\* Evaluates to 7 due to operator precedence \*)

(\* not valid

1+2\*3

=;

\*)

1234; (\* Integer literal \*)

123.4; (\* Real number literal \*)

true; (\* Boolean true \*)

false; (\* Boolean false \*)

"fred"; (\* String literal \*)

"H"; (\* String literal \*)

#"H"; (\* Character literal \*)

~ 1 + 2 - 3 \* 4 div 5 mod 6; (\* Integer arithmetic with various operators \*)

~ 1.0 + 2.0 - 3.0 \* 4.0 / 5.0; (\* Real number arithmetic \*)

"bibity" ^ "bobity" ^ "boo"; (\* String concatenation \*)

2 < 3; (\* Less than comparison \*)

1.0 <= 1.0; (\* Less than or equal to comparison for reals \*)

#"d" > #"c"; (\* Greater than comparison for characters \*)

"abce" >= "abd"; (\* Greater than or equal to comparison for strings \*)

1 = 2; (\* Equality comparison for integers \*)

true <> false; (\* Inequality comparison for booleans \*)

(\*1.3 = 1.3;\*) (\* Equality comparison for reals is not allowed \*)

1 < 2 orelse 3 > 4; (\* Logical OR \*)

1 < 2 andalso not (3 < 4); (\* Logical AND with negation \*)

true orelse 1 div 0 = 0; (\* Short-circuit OR to avoid division by zero \*)

if 1 < 2 then #"x" else #"y"; (\* Conditional expression \*)

if 1 > 2 then 34 else 56; (\* Conditional expression \*)

(if 1 < 2 then 34 else 56) + 1; (\* Conditional expression with arithmetic \*)

1 \* 2 + 3 \* 4; (\* Arithmetic with operator precedence \*)

"abc" ^ "def"; (\* String concatenation \*)

if (1 < 2) then 3.0 else 4.0; (\* Conditional expression with reals \*)

1 < 2 orelse (1 div 0) = 0; (\* Short-circuit OR to avoid division by zero \*)

(\*10/5; cannot merge int and real \*)

(\*#"a" = #"b" or 1 = 2; or is not legal here\*)

(\*1.0 = 1.0; cannot compare real and real\*)

(\*if (1 < 2) then 3; expected else\*)

1 \* 2; (\* Integer multiplication \*)

1.0 \* 2.0; (\* Real number multiplication \*)

(\*1.0 \* 2; cannot merge int and real\*)

real(123); (\* Convert integer to real \*)

floor(3.6); (\* Floor function for reals \*)

floor 3.6; (\* Floor function for reals \*)

str #"a"; (\* Convert character to string \*)

(\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*)

(\*the following functions were considered unbound value identifiers in this sosml\*)

(\*functions taken off internet\*)

fun square(x: int): int = x \* x; (\* Function to square an integer \*)

fun trunc(x: real): int = floor x; (\* Function to truncate a real number \*)

fun charToAscii(c: char): int = Char.ord c; (\* Function to get ASCII value of a character \*)

(\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*)

square 2 + 1; (\* Incorrect: evaluates to (square 2) + 1 \*)

square (2 + 1); (\* Correct: evaluates to square of 3 \*)

(\*trunc 5; trunc cannot use int in place of real\*)

(\*ord "a"; needs type char not string\*)

(\*if 0 then 1 else 2; cannot use int as bool\*)

(\*if true then 1 else 2.0; cannot merge int and real\*)

chr(trunc(97.0)); (\* Convert ASCII value to character \*)

chr(trunc 97.0); (\* Convert ASCII value to character \*)

(\*chr trunc 97.0 ; needs binding\*)

val x = 1 + 2 \* 3; (\* Define x as 7 \*)

x; (\* Evaluate x \*)

val y = if x = 7 then 1.0 else 2.0; (\* Define y based on condition \*)

val fred = 23; (\* Define fred as 23 \*)

fred; (\* Evaluate fred \*)

val fred = true; (\* Redefine fred as true \*)

fred; (\* Evaluate fred \*)

val a = "123"; (\* Define a as string "123" \*)

a; (\* Evaluate a \*)

val b = "456"; (\* Define b as string "456" \*)

val c = a ^ b ^ "789"; (\* Concatenate strings \*)

val a = 3 + 4; (\* Redefine a as 7 \*)

a; (\* Evaluate a \*)

b; (\* Evaluate b \*)

c; (\* Evaluate c \*)

val barney = (1 + 2, 3.0 \* 4.0, "brown"); (\* Define tuple barney \*)

val point1 = ("red", (300, 200)); (\* Define nested tuple point1 \*)

#2 barney; (\* Access second element of barney \*)

#1 (#2 point1); (\* Access first element of second element of point1 \*)

(1, 2); (\* Tuple with two elements \*)

(1); (\* Single element tuple is not valid \*)

#1 (1, 2); (\* Access first element of tuple \*)

(\*#1 (1); no tuple of one\*)

[1, 2, 3]; (\* List of integers \*)

[1.0, 2.0]; (\* List of reals \*)

[true]; (\* List of booleans \*)

[(1, 2), (1, 3)]; (\* List of tuples \*)

[[1, 2, 3], [1, 2]]; (\* List of lists \*)

[]; (\* Empty list \*)

nil; (\* Empty list \*)

null []; (\* Check if list is empty \*)

null [1, 2, 3]; (\* Check if list is not empty \*)

[1, 2, 3] @ [4, 5, 6]; (\* List concatenation \*)

val x = #"c" :: []; (\* Create list with single character \*)

val y = #"b" :: x; (\* Prepend character to list \*)

val z = #"a" :: y; (\* Prepend character to list \*)

val z = 1 :: 2 :: 3 :: []; (\* Create list of integers \*)

hd z; (\* Head of list \*)

tl z; (\* Tail of list \*)

tl (tl z); (\* Tail of tail of list \*)

tl (tl (tl z)); (\* Tail of tail of tail of list \*)

explode "hello"; (\* Convert string to list of characters \*)

implode [#"h", #"i"]; (\* Convert list of characters to string \*)

#2 (3, 4, 5); (\* Access second element of tuple \*)

hd (1 :: 2 :: nil); (\* Head of list \*)

hd (tl (#2 ([1, 2], [3, 4]))); (\* Access head of tail of second element of tuple \*)

(\*1 @ 2; cannot concatenate non-lists\*)

(\*hd (tl (tl [1, 2])); cannot merge int into lists\*)

(\* :: [2, 3]; :: is to merge int into a list\*)

fun firstChar s = hd (explode s); (\* Function to get first character of string \*)

firstChar "abc"; (\* Use firstChar function \*)

fun quot(a, b) = a div b; (\* Function for integer division \*)

quot (6, 2); (\* Use quot function \*)

val pair = (6, 2); (\* Define pair \*)

quot pair; (\* Use quot function with pair \*)

(\*the new line = does not work nicely here\*)

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

fact 5; (\* Use fact function \*)

fun listsum x = if null x then 0 else hd x + listsum (tl x); (\* Recursive list sum function \*)

listsum [1, 2, 3, 4, 5]; (\* Use listsum function \*)

fun length x = if null x then 0 else 1 + length (tl x); (\* Recursive list length function \*)

length [true, false, true]; (\* Use length function with boolean list \*)

length [4.0, 3.0, 2.0, 1.0]; (\* Use length function with real list \*)

fun badlength x = if x = [] then 0 else 1 + badlength (tl x); (\* Another list length function \*)

badlength [true, false, true]; (\* Use badlength function \*)

(\*badlength [4.0, 3.0, 2.0, 1.0]; operator and operand don't agree\*)

fun reverse L = if null L then nil else reverse (tl L) @ [hd L]; (\* Recursive list reverse function \*)

reverse [1, 2, 3]; (\* Use reverse function \*)

fun prod(a, b) = a \* b; (\* Function for integer multiplication \*)

fun prod(a: real, b: real): real = a \* b; (\* Function for real number multiplication \*)

Exercise 3: Inquire

1. Clarification Question: (more of a technical question) some of the typical functions like square and trunc didn't seems to work on SOSML, is there a reason for this or did i miss doing something in the reading?

Interview Question: Describe how ML handles recursive functions and polymorphism. Provide an example of a recursive function in ML and explain how ML’s type system supports polymorphic functions. How does the use of null in list operations help avoid unnecessary type restrictions? -- interview question written by ChatGPT as is standard for modern technology interviewers.