S1.1: Introduction to OCaml

CSci 2041:

Advanced Programming Principles

University of Minnesota,
Prof. Van Wyk,
Spring 2018


```
3                  false          "World"

fun x -> x * x     [false; false; false ]

"Hello"            (4, 'z')

fun x -> x + 1     -10

fun n -> int_of_string n

[ 1; 2; 3 ]        true

[ true; false ]    [ 4; 5; 6 ]

(1, 'c')           45

[ fun x -> x + 1 ; fun x -> x * x ]
```



```
int
     3                  -10           45
bool
     true               false
string
     "Hello"            "World"
int -> int
     fun x -> x + 1     fun x -> x * x
int -> string
     fun n -> int_of_string n
int list
     [ 1; 2; 3 ]        [ 4; 5; 6 ]
bool list
     [ true; false ]    [false; false; false ]
int * char
     (1, 'c')           (4, 'z')
(int -> int) list
     [ fun x -> x + 1 ; fun x -> x * x ]
```

Organizing those values above.

- There are many types of values.
- We will group them by types.
- A type is a name for a set of set of values.
- `int` is a set of values
- `int list` is a set of values
- `'a list` is a set of values
  It contains `int list` and `string list` and `int list list` ...

Strong static type systems

- OCaml has a *strong*, *static* type system
- It is a *safe* language.
- What does "safe" mean?
- *strong* = program never execute type-incorrect operation or invalid memory access
- *static* = this is checked before the program runs.
- *safe* = strong, static type system

Type systems

The challenge - design strong static type systems that are

1. expressive, and
   - It is difficult to have a static type for non-zero integers.
   - So the question becomes

     "What properties can types express?"
2. easy to use.
   - Type inference can help with this as we don't need to write down all the types. But it is recommended to write types for some parts to provide machine-checked documentation.