State

Dr. Mattox Beckman

University of Illinois at Urbana-Champaign
Department of Computer Science

Objectives

- Explain how to use `let` and function declarations to control when a variable is created.
- Use functions to encapsulate state in a safe manner.

Local Variable Example

```
# let foo x =
    let a = 10 + 20 in
      a + x;;
val foo : int -> int = <fun>
# foo 15;;
- : int = 45
# foo 30;;
- : int = 60
```

How many times does the 10 + 20 get computed?

Global Variable Example

```
# let a = 10 + 20;;
val a : int = 30
# let foo x =
    a + x;;
val foo : int -> int = <fun>
# foo 15;;
- : int = 45
# foo 30;;
- : int = 60
```

How many times does the 10 + 20 get computed?

Encapsulated Variable Example

```
# let foo =
    let a = 10 + 20 in
      fun x -> a + x;;
val foo : int -> int = <fun>
# foo 15;;
- : int = 45
# foo 30;;
- : int = 60
```

How many times does the 10 + 20 get computed?

Using Local State

```
# let counter =
    let ct = ref 0 in
    fun () -> ct := !ct + 1; !ct;;
val counter : unit -> int = <fun>
# counter ();;
- : int = 1
# counter ();;
- : int = 2
```

- This protects `ct`, making it available only to `counter`.

Bad Pun

```
# fun twice f x = f (f x)
# twice counter () + twice counter ();;
res4 : Int = 6
# twice counter () + twice counter ();;
res4 : Int = 14
```

- Function `twice` is the Church numeral for 2.
- You know what this means, right?

Random Number Generators

```
# let mkRandom s =
    let s = ref s in
      fun () -> s := (!s * 541 + 5) mod 1024; !s;;
val mkRandom : int ref -> unit -> int = <fun>
# let rnd0 = mkRandom (ref 1);;
val rnd0 : unit -> int = <fun>
# rnd0 ();;
- : int = 546
# rnd0 ();;
- : int = 479
# rnd0 ();;
- : int = 72
```

Function Tuples

```
# let (counter, reset) =
    let ct = ref 0 in
      (fun () -> ct := !ct + 1; !ct),
      (fun nv -> ct := nv);;
val counter : unit -> int = <fun>
val reset : int -> unit = <fun>
# counter ();;
- : int = 1
# reset 5;;
- : unit = ()
# counter ();;
- : int = 6
```