CSE341: Programming Languages

Lecture 10
References, Polymorphic Datatypes, the Value Restriction, Type Inference

Ben Wood, filling in for Dan Grossman

Fall 2011

Callbacks

A common idiom: Library takes functions to apply later, when an *event* occurs – examples:

- When a key is pressed, mouse moves, data arrives
- When the program enters some state (e.g., turns in a game)

A library may accept multiple callbacks

- Different callbacks may need different private data with different types
- Fortunately, a function's type does not include the types of bindings in its environment
- (In OOP, objects and private fields are used similarly, e.g., Java Swing's event-listeners)

Mutable state

While it's not absolutely necessary, mutable state is reasonably appropriate here

- We really do want the "callbacks registered" and "events that have been delivered" to *change* due to function calls

For the reasons we have discussed, ML variables really are immutable, but there are mutable references (use sparingly)

- New types: `t ref` where `t` is a type
- New expressions:
  - `ref e` to create a reference with initial contents e
  - `e1 := e2` to update contents
  - `!e` to retrieve contents (not negation)