CS4450 Lecture 33 Notes

Monday, November 5, 2018 3:01 PM

Today, we'll be talking about imperative programming languages.

StandardML: First functional programming language that wasn't Scheme or Lisp.

Reference Types

ML has types for "assignment variables" called reference types.

References are like pointers, but type-safe. Different than C, where you can store anything into an address.

Examples

Create a reference with "ref".

Read a reference with "!" (bang)

Write a reference with "!"

Sequencing

(e1 ; … ; en)

The arguments to a sequence can be anything:

```
(1 ; "hey" ; 3.14);
```

Reference is like a pointer, except there is no explicit allocation/deallocation of memory, and no possibility of dereferencing a null pointer.

No casting (i.e., references are type-safe)

Aliasing: Two different program variables pointing to the same thing.

Store: Something that takes an address and returns a value.

How can someone represent Store in Haskell?

It could be a function from address to value.

It could also be an association list of address-value pairs.

Extension: Multiple Return Values

We consider two extensions to this language:

1. Return values: i.e., what if you want to define "+"?
2. Errors: i.e., what happens when something goes wrong? (e.g. when something isn't declared)