CSE 341 AB: Section 4

Josh Pollock
joshpoll@cs.uw.edu
OH: Thursdays 4:30pm - 5:30pm

Questions?

Lecture Material

- Higher-Order Functions
- Currying
- Mutable References
- Modules

Homework 3 (due Monday)

In-class midterm (May 3/next Friday)

Agenda

1. Currying
2. Recursion
    a. Tracing Recursion
    b. Mutual Recursion
    c. Tracing Mutual Recursion
3. Modules
4. Abstract Machine Example

Currying

Curried functions are just *syntactic sugar*.

Because functions are first class in SML, we can return them so currying works!

```
fun foo x y = e

val foo = fn x => (fn y => e) (* the same except for recursion! *)

fun foo x = fn y => e (* actually the same! *)
```

Currying is usually highly optimized.

Fully applying a function (e.g. calling `foo` with 2 args) generates the same code as calling it's uncurried equivalent.

Tracing Recursion

Tracing recursion follows naturally from the tracing we've already seen.

Mutual Recursion

Mutually recursive datatypes

We sometimes need two datatypes that refer to each other.

Mutually recursive functions

We also sometimes need two functions that refer to each other.

See **`section4_ab.sml`**.

Tracing Mutual Recursion

Mutual recursion is almost as easy as regular recursion.

Let's say `f` and `g` are mutually recursive.

Just add both functions' closures to both of their environments.

Modules

Makes multi-file and multi-author projects easier.

- Namespaces and Code Separation
  - Encourages **modularity**.
- Abstraction (Information Hiding)

Abstraction (Information Hiding)

- Module signatures hide implementation details
- Author is in control of how users interact with their code (interface)
- Users don’t have to think about implementation details
  - Don’t have to worry about them
  - Can’t abuse them (intentionally or otherwise)
- Authors can maintain internal invariants

Modules are second class!

They were added after the rest of the core language was developed.

You can’t pass them around like functions.

Making them first class an active area of research.

```
NOT POSSIBLE!!!

structure Table = if size > threshold
                  then (structure HashMap)
                  else (structure TreeMap)

NOT POSSIBLE!!!
```

Abstract Machine Example

Use higher-order functions and modules to implement language semantics.

See **`section4_ab.sml`**.