CSE 341: Programming Languages

Dan Grossman

Spring 2008

Lecture 17— define-struct; Implementing higher-order functions

Data in Scheme

Recall ML's approach to each-of, one-of, and self-referential types:

```
datatype t =
  Foo of int | Bar of int * int | Baz of string * t
```

Pure Scheme's approach:

- There is One Big Datatype holding *every value*.
- Built-in predicates like `null?`, `number?`, `procedure?`
- Primitives implicitly raise errors for "wrong variant"
- Use pairs (lists) for each-of types
- Can also use for one-of types with explicit "tags"
  - Like our force/delay with a boolean field
  - Symbols better style
- Use helper functions like `caddr` (and/or define your own).

Dynamic typing

There is still good reason to have support for *constructors*:

- Make a `foo` that has fields `x`, `y`, `z`
- Test to see if you have a `foo` or not

But with dynamic typing:

- Constructors are not "grouped" into types (just added to the One Big Datatype)
- The fields can hold anything

Orthogonally: We don't have pattern-matching.