CSE 341: Programming Languages

Autumn 2005
Lecture 23 — define-struct & misc
A Few Miscellaneous Scheme Topics

In DrScheme, you can use square brackets as well as parenthesis. (You need to match left parentheses with right parentheses, left square brackets with right square brackets.) Suggestion: use not at all, or sparingly for readability.

’ is a macro — ’x and ’(a b c) are equivalent to (quote x) and (quote (a b c)).

Scheme functions can take a variable number of arguments.

(define (squid a b . c)
  (print a)
  (print b)
  (print c))

squid requires at least 2 arguments. Any remaining arguments (perhaps 0) are put into a list, which is bound to c.
Data in Scheme

Recall ML’s approach to each-of, one-of, and self-referential types.

Pure Scheme’s approach:

• There is One Big Datatype with built-in predicates.
• Use pairs (lists) for each-of types.
• Primitives implicitly raise errors for “wrong variant”
• Use helper functions like `caddr` and your own.
define-struct

MzScheme extends Scheme with define-struct, e.g.:

(define-struct square (x y))
(define-struct piece (squares))

Semantics:

• Binds constructors (make-square, make-piece) that take arguments and make values.

• Binds predicates (square?, piece?) that take one argument and return \#t only for values built from the right constructor.

• Binds accessors (square-x, square-y, piece-squares) that take one argument, return the appropriate field, and call error for values not built from the right constructor.

• Binds mutators (set-square-x!, set-square-y!, set-piece-squares!).
**define-struct is special**

define-struct creates a new variant for the One Big Datatype.

Claim: `define-struct` is not a function.

Claim: `define-struct` is not a macro.

It could be a macro except for one key bit of its semantics: Values built from the constructor cause every *other* predicate (including all built-in ones) to return `#f`.

Advantage: abstraction

Disadvantage: Can't write “generic” code that has a case for every possible variant in every Scheme program.
Idiom for ML datatypes

Instead of a datatype with $n$ constructors, you just use
define-struct $n$ times.

That “these $n$ go together” is just convention.

Instead of case, you have a cond with $n$ predicates and one
“catch-all” error case.