CSE 341: Programming Languages

Dan Grossman

Fall 2004

Lecture 15— Macros

Today

- What are macros and what do they mean?
  - Why do they have a bad reputation?
- Scheme's macro system and hygiene
  - Free variables in macros
  - Bound variables in macros
  - Why hygiene is usually what you want
- What macros are good and not good for

Macros

To oversimplify, a macro is just a rule for rewriting programs as a prepass to evaluation. So it's very syntactic.

The "level" at which macros are defined affects their usefulness.

- "Sublexical" e.g.: Replace `car` with `hd` would turn `cart` into `hdt`.
  - No macro system does this; so macro-expander must know how to break programs into tokens.
- "Pre-parsing" e.g.: Replace `add(x,y)` with `x + y` (where `x` and `y` stand for expressions) would turn `add(x,y) * z` into `x + y * z`.
  - Some macro systems are this "dumb" (i.e., token-based); macro writers use more parens than Schemers.
- "Pre-binding" e.g.: Replace `car` with `hd` would turn `(let* ([hd 0] [car 1]) hd)` into `(let* ([hd 0] [hd 1]) hd)`.
  - Few macro systems let bindings shadow macros; Scheme does