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 \texttt{car} with \texttt{hd} would turn \texttt{car t} into \texttt{hd}.  
  - No macro system does this; so macro-expander must know how to break programs into tokens.

- “Pre-parsing” e.g.: Replace \texttt{add(x,y)} with \texttt{x + y} (where \texttt{x} and \texttt{y} stand for expressions) would turn \texttt{add(x,y) * z} into \texttt{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 \texttt{car with hd} would turn \texttt{(let* ([\texttt{hd 0}] [\texttt{car 1}]) \texttt{hd})} into \texttt{(let* ([\texttt{hd 0}] [\texttt{hd 1}]) \texttt{hd})}.
  - Few macro systems let bindings shadow macros; Scheme does

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

The bad news

- Macros are very hard to use well.

- Most macro systems are so impoverished they make it harder.

- Actual uses of macros often used to ameliorate shortcomings in the underlying language.

But:

- Macros have some good uses

- Scheme has a very sensible, integrated macro system

- So let’s “do macros justice” for the day.
Hygiene

A “hygienic” macro system:

- Gives fresh names to local variables in macros at each use of the macro
- Binds free variables in macros where the macro is defined

Without hygiene, macro programmers:

- Get very creative with local-variable names
- Get creative with helper-function names too
- Try to avoid local variables, which conflicts with predictable effects

Hygiene is a big idea for macros, but sometimes is not what you want.

Note: Letting variables shadow macros is also useful, but a separate issue.

Why macros

Non-reasons:

- Anything where an ordinary binding would work just as well.
- Including manual control of inlining.

Reasons:

- Cosmetics
- “Compiling” a domain-specific language
  - But error messages a tough issue
- Changing evaluation-order rules
  - Function application will not do here
- Introducing binding constructs
  - A function here makes no sense