

Class 52: Metaprogramming

Held: Friday, May 4, 2007

Summary: Today we consider techniques for *metaprogramming*, writing programs that generate programs.

Related Pages:

- EBoard.
- Lab: Metaprogramming.
- Reading: Metaprogramming.

Due

- Exam 3

Notes:

- No reading for Monday.

Overview:

- Code vs. Data in Scheme.
- Rewriting Named `let` structures.
- Generating Code for Objects.

Code vs. Data in Scheme

- By this point in your career, you may have noticed that code and data in Scheme look remarkably the same.
- Consider `(f a b)`
 - Is this the application of the function `f` to the values `a` and `b`?
 - Is this a three element list of the values `f`, `a`, and `b`?
- In Scheme, the two are a bit interchangeable.
- For example, if you put that value in a file, you can read it with `read` and treat it like data or you can read it with `load` and treat it like code.
- What are the implications?
 - You can write procedures that analyze the instructions in other procedures.
 - You can write procedures that modify the instructions in other procedures.
 - You can write procedures that generate other procedures.

Rewriting Named Lets

- As an example, let us consider one of the syntactic forms that many of you find difficult, the named `let`.
- As you may recall, named lets have this form

```
(let proc ((param1 init1)
          (param2 init2)
          ...
          (paramn initn))
  body1
  ...
  bodym)
```

- Named lets are a more concise (and, some say clearer) way to express

```
(letrec ((proc
          (lambda (param1 ... paramn)
            body1
            ...
            bodym)))
  (proc init1 ... initn))
```

- We should be able to automatically convert the former to the latter.
- What is the form of the named `let`? A list of four or more elements such that
 - The `car` is the symbol `let`
 - The `cadr` is a symbol.
 - The `caddr` is a list of two element lists.
 - The `cddddr` is a list of expressions.
- To turn it into a `letrec`, we need to make a three element list in which
 - The `car` is the keyword `letrec`
 - The `cadr` is a list of the definition of `proc`
 - The `caddr` is the procedure call.
- To make the procedure call, we build a list of the procedure name (the `cadr` of the original thing) and all the initial values.
 - We get the list of initial values by mapping `cadr` onto the `param+inits`
- ...

Generating Code

- We can also use these techniques to generate code.
- Why would we want to generate code? Because we often find that we write very similar code, but higher-order techniques don't suffice.
- Consider the structure of a typical object.
 - They begin with `lambda let lambda`
 - They use similar `conds` to select the methods
 - They have similar `:save-to-file` and `:restore-from-file` methods.
- We can therefore write procedures that generate the common code.

Copyright © 2007 Samuel A. Rebelsky. This work is licensed under a Creative Commons Attribution-NonCommercial 2.5 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/2.5/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.