Programming Language Concepts

Alvin Lin

January 2018 - May 2018

Scheme

Scheme was invented at MIT in the 1970's and originally called Schemer. Common-LISP was developed in the 1980's after a merger between LISP and Scheme failed. We will use DrRacket in class. Since we are doing functional programming, we will not be using any `do` or `begin` commands or any functions ending with `!`.

Scheme: an example of REPL

A REPL is a read-evaluate-print-loop. In the console, expressions entered are evaluated one at a time.

Scheme Expression Types

- **Atoms**: numbers, strings, characters, symbols
- **Symbols**: Scheme's identifiers. Symbols can be bound to values.
- **Lists**: zero or more expressions enclosed in parentheses

Evaluation Rules

- Literals (numbers, strings, characters) evaluate to themselves.
- Symbols evaluate to the values to which they are bound. They are both names and values.
- Lists are treated as function calls.

`(* 9 8)`

- The symbol `*` is bound to the function's definition.
- The two arguments to the function are 9 and 8, which are evaluated left to right.
- The values of the arguments are passed to the function bound to `*` as a list.
- Function `*` returns the value 72.

`(/ (+ 5 3) (- 4 2))` evaluates to 4.

`define`

We can use the `define` function to set the values of global variables. Example: `(define x 3.14)`. Note that the first argument is not evaluated. This is normally only used for functions, because global variables are bad.

Delayed Evaluation

The quote operator has a special function. It takes a symbol and returns it verbatim.

Lists

Lists are linked, using 2-part *cons cells*. The `cons` function builds cells from two values. The null value `()` is a proper list.

- `(cons 4 (cons 3 '()))`
- `(cons 4 otherlist)`

An unquoted list is treated as an expression: evaluation of a list returns the value returned from the function call. Thus we use the function `list` to build lists.

Pulling Lists Apart

`car` (Contents of Address Register) extracts the first element of a list. `cdr` (Contents of Decrement Register) extracts everything after the first element.

Booleans

- `#f` and `'()` are false.
- `#t` and everything else are true.
- Predicate functions: `list?`, `number?`, etc
- Boolean functions: `not`, `and`, and `or`
- Number comparision: `=`, `<=`, etc

Alternation

`if` is another special function.

```
(if then-expression else-expression)
```

`cond` allows for multi-way conditionals.

```
(cond [(text-expression) (then-body)] [(test-expression) (then-body)]
                                  ...)
```

Functions

The `lambda` function allows you to declare an anonymous function, which can be bound to a symbol using `define`. Example:

```
(lambda (args..)  expr)
(lambda (x y) (+ x y))
```

IO (non-functional)

For input there is the functions `read readline`, and for output there is `display newline`.

Acceptable Variable Binding

`(let ((symbol1 expr1) (symbol2 expr2)) expr...)`
Binds a variable locally within `expr`.

Command Line Programming

`racket [-i]`: Loads standard library; runs REPL
`racket -f file`: Loads standard library and `file`; exits
`racket -if file`: Loads standard library and `file`; runs REPL
The comment character is `;`, which comments to the end of the line. Loaded programs just execute everything in them. In REPL mode, each line you type is loaded, and the result is displayed in the console.

High Order Functions

Scheme treats functions like data.
`(map car '((1 2) (2 3))) -> (1 3)`

You can find all my notes at `http://omgimanerd.tech/notes`. If you have any questions, comments, or concerns, please contact me at alvin@omgimanerd.tech