Some Additional Racket Topics: Improper Lists, Functions with a Variable Number of Arguments; Quasiquoting

Improper Lists

Normally, the cdr of a cons cell is either the empty list or another cons cell. A list in which the cdr of a cons cell is something else is an *improper list*, and is written using a dot.

For example,

```
'(2 . 3)
```

denotes this list:

```
 _______________
|       |       |
|   o   |   o   |
|___|___|___|___|
    |       |
    |       |
    2       3
```

And

```
'(2 3 . 4)
```

denotes:

```
 _______________         _______________
|       |       |       |       |       |
|   o   |   ----|----->|    o   |   o   |
|___|___|_______|      |____|___|___|___|
    |                        |       |
    |                        |       |
    2                        3       4
```

A common use for improper lists is to store name/value pairs in a so-called *association list*. The MULP interpreter uses an association list of exactly this sort to store environments.

Functions with a Variable Number of Arguments

We've used functions with a variable number of arguments (for example +), but have only written them with a fixed number of arguments.

Example definitions with a variable number of arguments:

```
(define (squid a b . c)
   (display a) (newline)
   (display b) (newline)
   (display c) (newline))
```

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

This function can take 0 or more arguments:

```
(define (average . lst)
   (if (null? lst)
      (error "can't take the average of an empty list")
      (/ (apply + lst) (length lst))))
```

Quasiquotes

"Backquote" or "quasiquote" expressions are useful when you want to construct a list for which you know most, but not all, of the desired structure in advance. For example, suppose that you want to return a list of band members that includes a mysterious 5th Beatle. With what we know so far you could do this:

```
(define (sixties-bands m)
   (list '(peter paul mary)
         (list 'john 'paul 'george 'ringo m)
         '(simon garfunkle)))
```

Since we know all of the structure except for the 5th Beatle, instead we could use quasiquoting:

```
(define (quasibands m)
   `((peter paul mary)
     (john paul george ringo ,m)
     (simon garfunkle)))
```

Here everything in the quasiquoted list is quoted, except when it is preceded by a comma (which unquotes it).

You can also splice in a list in the middle of another using an @ sign.

```
(define x '(clam squid))
```

Then

```
`(octopus ,@x mollusc oyster) => (octopus clam squid mollusc oyster)
```

compare with:

```
`(octopus ,x mollusc oyster) => (octopus (clam squid) mollusc oyster)
```

The `` ` ``, `,` and `@` characters above are a shorthand for `quasiquote`, `unquote`, and `unquote-splicing` respectively. We could rewrite two of the above examples as:

```
(quasiquote (octopus (unquote x) mollusc oyster))
```

```
(quasiquote (octopus (unquote-splicing) x mollusc oyster))
```