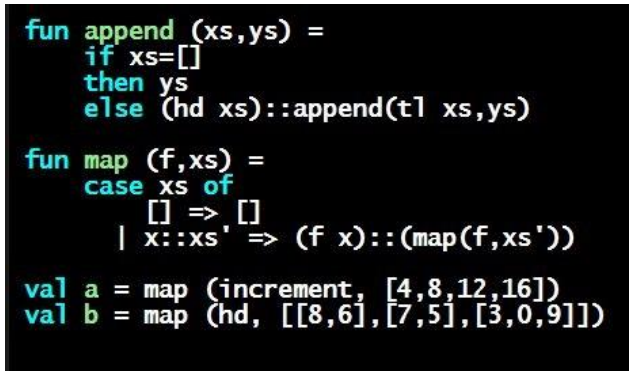


Programming Languages

Dan Grossman

2013

Syntax and Parentheses

Racket syntax

Ignoring a few “bells and whistles,”

Racket has an amazingly simple *syntax*

A *term* (anything in the language) is either:

- An *atom*, e.g., **`#t`**, **`#f`**, **`34`**, **`"hi"`**, **`null`**, **`4.0`**, **`x`**, …
- A *special form*, e.g., **`define`**, **`lambda`**, **`if`**
  - Macros will let us define our own
- A *sequence* of terms in parens: **`(t1 t2 … tn)`**
  - If **`t1`** a special form, semantics of sequence is special
  - Else a function call

- Example: **`(+ 3 (car xs))`**
- Example: **`(lambda (x) (if x "hi" #t))`**

Brackets

Minor note:

Can use `[` anywhere you use `(`, but must match with `]`

- Will see shortly places where `[…]` is common style
- DrRacket lets you type `)` and replaces it with `]` to match

Why is this good?

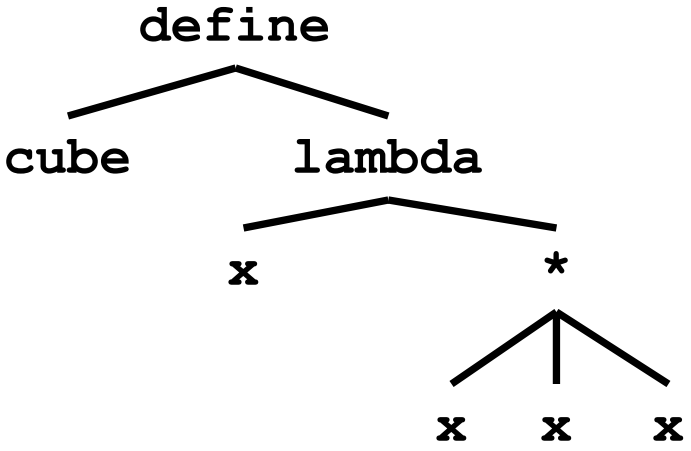
By parenthesizing everything, converting the program text into a tree representing the program (*parsing*) is trivial and unambiguous

- Atoms are leaves
- Sequences are nodes with elements as children
- (No other rules)

Also makes indentation easy

Example:

```
(define cube
  (lambda (x)
    (* x x x)))
```



No need to discuss “operator precedence” (e.g., `x + y * z`)

Parenthesis bias

- If you look at the HTML for a web page, it takes the same approach:
  - `(foo` written `<foo>`
  - `)` written `</foo>`

- But for some reason, LISP/Scheme/Racket is the target of subjective parenthesis-bashing
  - Bizarrely, often by people who have no problem with HTML
  - You are entitled to your opinion about syntax, but a good historian wouldn't refuse to study a country where he/she didn't like people's accents

For fun…

http://xkcd.com/297