2013W1-lecture4

September 12, 2013

Contents

1 Question of the Day

Which of these is a function in Racket: `/` (division) or `and` (logical "and")?

1.1 Solution

How would we even tell? Why would we care?

Most “operations” we use in Racket are really functions; so, it can be surprising when one isn't. When it's not, it's generally because its desired behavior breaks the semantics of function application in Racket.

So, how do we tell if an operation is a function? We ask. Try evaluating the expressions `/` and `and` at the REPL in Racket (or `plai-typed`).

Now, think about why one is a function and one isn't. Hint: consider this code:

```
(/ 1 (/ 1 0))

(and false (= (/ 1 0) 1))

; Or maybe better than the first line, try this:
(define (ignore-second x y)
  x)

(ignore-second false (= (/ 1 0) 1))
(and false           (= (/ 1 0) 1))
```

Interestingly, even *more* “operations” in the Haskell programming language are functions than in Racket, because the rules in Haskell are more permissive for just the sort of issue we ran into here. Does that make Haskell better? Of course not. For example, Racketeers can often be found complaining about the fact that in allowing the `+` operator to be overridden like any other operation on objects, C++ makes a terrible and confusing mistake.

The irony of this is the following Racket code, which works in `plai-typed` as well:

```
(define (+ a b)
  (- a b))

(+ 1 2)
```

2 Quiz!

3 Logistics

3.1 How to submit your quiz corrections (even if you got everything right)

See the quiz document itself (posted in the assignments area of the website). The first one is handin target `ca1`.

3.2 Reminder of where to find office hours

The syllabus!

3.3 Reminder of Piazza (e.g., to answer "Where do I find `plai-typed` documention?")

https://piazza.com/ubc.ca/winterterm12013/cpsc311

WARNING: I will soon **close** Piazza registration! (After the add/drop deadline.)

3.4 Reminder of assignment due Fri at 8PM

4 Finishing the last TWO lectures! :)

file:2013W1-lecture2.org file:2013W1-lecture3.org

5 What have we learned today?

There was no new material here except the QotD, Quiz, and Logistics.

From QotD:

- More on the definition/application of the term *semantics*.

  The semantics of function application in Racket includes the idea that argument expressions are evaluated to values *before* the function is called. Therefore, a "short-circuit `and`" cannot be written as a function, at least not with the type `(boolean boolean -> boolean)`.