2013W1-lecture9

September 25, 2013

Contents

1 Question of the Day

Haskell is probably the most prominent programming language that uses lazy evaluation. And it's used *lots* of places, right?

Seriously, does anything actually *use* lazy evaluation?

(Hint for what *I'm* thinking of—but I'd rather hear what *you're* thinking of—try running `man yes` from a UNIX prompt.)

```
 SOLUTION
Hopefully you came up with some examples of your own.  Here's my
favorite example: UNIX pipes.

Pipes aren't exactly "lazy evaluation", but they do let you chain
```

```
commands together.  So, for example, let's say you want to install
some incredibly complex program (gcc?), but there's a bunch of prompts
during the process where it asks you "Do you want to do this? (y/n)".
You'd like to say "yes" to all of them.  Just run:

   yes | install-big-program

yes will output y over and over again.  But, its write system call
will block when the output pipe fills.  At that point, it stops
running.  "Evaluation" only goes partway.

Meanwhile, install-big-program will block when it doesn't have
anything to read from the pipe.. but it always will in this case.
Why?  Because yes is busy pumping an infinite number of y's into the
pipe.  As soon as install-big-program consumes enough of the y's to
unblock yes's write system call, it'll fill that pipe right back up
again!

Then, when install-big-program terminates, the whole pipe will
terminate!

So, not exactly lazy evaluation, but some of the same ideas, used to
great effect.. and not just to say yes to things.

[[file:homer_dipping_bird.png]]



Not convinced?  Here's an example stolen from
http://www.unix.com/shell-programming-scripting/173755-counting-occurrences-all-words-

cat *.txt | \                         # concatenate all text files together
  tr '[A-Z]' '[a-z]' | \              # to-lower-case
  tr ' \t' '\n\n' | \                 # change all whitespace to newlines
  sed -e "/^$/d"| \                   # eliminate blank lines
  sort | \                            # sort (puts occurrences of the same word togeth
  uniq -c | \                         # eliminate duplicates (but retain count of dupl
  sort -nr                            # sort in reverse numerical order (by count)

It counts the number of occurrences of each word in all the text files
```

```
in the current directory, with the result sorted by frequency of
occurrence.  (sed has its OWN "little language" that it understands
for its commands.)




Here's a link into the history of pipes in UNIX:
http://www.bell-labs.com/history/unix/streams.html

A couple of pages later on, there's fascinating stuff about how
thinking about the syntax of how to do this stymied work on actually
doing it.  This brings up the critical point that we *think* in
representations.  If you cannot come up with *any* syntax for a thing,
then you probably can't think it.  (Even for describing semantics, we
use some kind of syntax!)
```

2 Logistics

2.1 Quiz 2!

2.1.1 Group quiz 2!

2.1.2 Corrections due Tue at 8PM

2.2 Programming Assignment #2

Due Friday. Questions?

2.3 Programming Assignment #3

Released. Note that the milestone and the final submission EACH count as a full assignment. Get a *great* score on the milestone! (Milestone due 1 week from Friday; full due 2 weeks from Friday.)

We encourage you to remain in the same team for PA3 as you used for PA2, but you don't have to.

3 Finishing the previous class's previous class's notes!

file:2013W1-lecture7.org

4 What have we learned today?

- From QotD: A bit of what lazy evaluation can be good for: chaining together processes (including infinite ones!) in a demand-driven way.