Graduate Blog

Rantings of a semi-conscious graduate student -- take with a grain of salt.

Anonymous Fibonacci in Haskell

Posted by chathaway on Sept. 14, 2016, 3:55 p.m.

We all know how to write the Fibonacci function; but can we do it using only anonymous functions?It's easy enough in something like Javascript; we make the Fibonacci function take in the function as an argument, then our normal variables. i.e.

```
(function (x, n) { return x(x, 1, 0, n) }) (function(m, a, b, n) {if (n == 0) { return a } else { return m(m, a+b, a, n-1) }}, 5)
```

Or, if you want to use ES6:

```
((x, n) => { return x(x, 1, 0, n) }) ((m, a, b, n) => {return n == 0 ? a : m(m, a+b, a, n-1)}, 5)
```

But I digress.The trouble is that doing this directly in Haskell produces a "Occurs check: cannot construct the infinite type:"

The solution to this is to use the fix function, which basically calls "function (function) ".For example:

```
fix (\next a b n -> if n == 0 then a else (next (a+b) a (n-1))) 1 0 10
```

So much fun :)
