Anonymous Functions

CS 151: Introduction to Computer Science I

July 10, 2019

Returning functions: local

Previously:

```
(define-type Int-Set (Integer -> Boolean))

(:  complement :  Int-Set -> Int-Set)
;; take the complement of a set of integers
(define (complement set)
  (local
   {
    (:  comp-set :  Int-Set)
    (define (comp-set n)
      (not (set n)))}
   comp-set))
```

Lambdas

```
(lambda ([n :  Integer]) (+ n 1))
```

```
(lambda ([nickels :  Integer] [dimes :  Integer])
  (+ (* 5 nickels) (* 10 dimes)))
```

```
((lambda ([s :  String]) (string-ref s 0)) "qwop")
    ⟹ #\q
```

```
(:  my-func :  Integer -> Integer)
(define my-func (lambda ([n :  Integer]) (remainder
n 75)))
```

Returning functions: lambdas

Today:

```
(define-type Int-Set (Integer -> Boolean))

(:  complement :  Int-Set -> Int-Set)
;; take the complement of a set of integers
(define (complement set)
  (lambda ([n :  Integer]) (not (set n))))
```

Lambdas for higher order programming

```
(map (lambda ([s :  String]) (string-append s "!"))
      (list "no" "go"))
⟹ '("no!" "go!")
```

Compute the sum of squares of a `(Listof Real)`:

```
(:  sum-sq :  (Listof Real) -> Real)
(define (sum-sq nums)
  (foldr
    (lambda ([x :  Real] [acc :  Real]) (+ (* x x)
acc))
    0
    nums))
```

Lambda functions should be short. Otherwise give them a name with local

Currying

Normally, addition takes two `Numbers` and adds them:

```
(: add : Number Number -> Number)
(define (add x y) (+ x y))
```

Alternatively, can provide the inputs one at a time (or "fix" an input)

```
(: add : Number -> (Number -> Number))
(define (add x)
  (lambda ([y : Number]) (+ x y)))
```

Equivalent type:

```
(: add: Number -> Number -> Number)
```

Some History

Bertrand Russell, early 1900s: "What is mathematics? What does $f(x) = x^2$ mean? Is it the same as $f(y) = y^2$?"

David Hilbert, 1900s: Hilbert's 10th problem, Entscheidungsproblem

Alonzo Church, 1928: "Here are the rules for defining functions, *lambda calculus*"

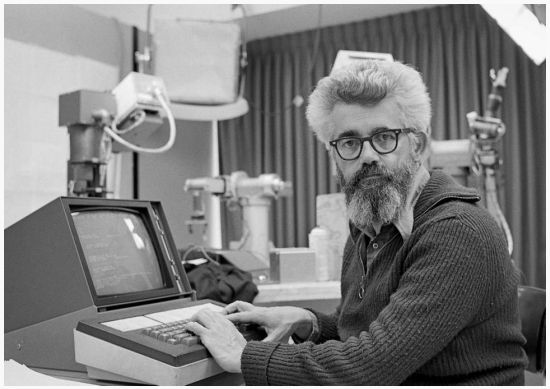
```
λx. x + 1
```

$$\mathsf{Y} = \lambda f.\,(\lambda x.\,f\,(x\,x))\,(\lambda x.\,f\,(x\,x))$$

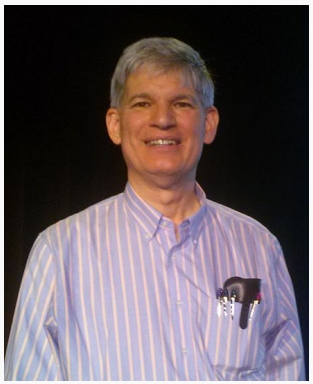
1927, briefly: Alonzo Church teaches at UofC

Alan Turing, 1936: "Here's another (equivalent) way to

Some History



John McCarthy, 1958: Lisp, a high-level programming language



Guy Steele and Gerald Sussman, 1970s: Scheme, a minimalist Lisp dialect

Matthias Felleisen, 1990s: Racket, a slightly less minimalist Lisp dialect

What to know

- Lambda expressions
- Currying