CSC 520, Spring 2020

Principles of Programming Languages

Michelle Strout

THE UNIVERSITY OF ARIZONA

Version: 1

April 27, 2020

Plan

- **Announcements**
  - HW10 is due Wednesday April 29th
  - Project talk recordings posted online due Monday May 4th, https://github.com/UofA-CSc-520-Spring-2020/CSc520Spr20-CourseMaterials/blob/master/project.md
  - Reviews of other people's talks will be due Wednesday May 6th
- **Last time**
  - Lambda Calculus Overview
  - Programming in the Lambda Calculus
  - Operational Semantics of Lambda Calculus
- **Today**
  - Recursion in lambda calculus with the Y combinator (only slides for this lecture, no notes)

Recursive Functions

- If lambda calculus is going to allow us to compute any function, we need for it to handle recursion.
- Example:

$$\text{fact} \equiv (\lambda n.\text{if } (\text{zero } n)\ 1\ (\text{mult } n\ (\text{fact } (\text{pred } n))))$$

- Unfortunately, the name fact appears in the expression itself. Remember that we defined $\equiv$ -operator as *macro-expansion*, and recursive macros make no sense.
- Recursion is defined in normal programming languages, but not in lambda calculus.

Fixed Points

- A fixed point is a value x in the domain of a function that is the same in the range $f(x)$.
- In other words, a fixed point of a function is a value left fixed by that function; for example, 0 and 1 are fixed points of the squaring function.
- Formally, a value x is a fixed point of a function f if

$$f(x) = x$$

-Spring 2005—23 [20]

Slide content credits: Christian Collberg

Fixed Points — Examples

- Every value in the domain of the identity function is a fixed point: $((\lambda x.x)$)
- $\texttt{factorial}(1) = 1$
- $\texttt{fibonacci}(0) = 0$
- $\texttt{fibonacci}(1) = 1$
- $\texttt{square}(0) = 0$
- $\texttt{square}(1) = 1$
- $\frac{de^x}{dx} = e^x$

-Spring 2005—23 [21]

Slide content credits: Christian Collberg

Fixed Points — Examples. . .

f	fixed point
$(\lambda x.6)$	6
$(\lambda x.6 - x)$	3
$(\lambda x.x^2 + x - 4)$	2,-2
$(\lambda x.x)$	every value
$(\lambda x.x + 1)$	no value

When 6-x = x, 6 = 2x, x = 3

When x^2 + x - 4 = x, ???

When x+1=x, 1=0

- I.e., a fixed point is where you get back whatever you put in!

Fixed Point Combinators

THE UNIVERSITY OF ARIZONA
Computer Science

- A combinator is a lambda-expression with no free variables.
- A fixed point combinator is a function Y which, given another function f, computes a fixed point of f, so that

$$f(\mathsf{Y}(f)) = \mathsf{Y}(f)$$

  for all functions f.
- Let's look at the fact function again:

$$\mathsf{fact} \equiv (\lambda n.) if\ (\mathsf{zero}\ n)\ 1\ (\mathsf{mult}\ n\ (\mathsf{fact}\ (\mathsf{pred}\ n)))$$

Slide content credits: Christian Collberg

Fixed Point Combinators...

- Let's turn

$$\text{fact} \equiv (\lambda n.\text{if } (\text{zero } n)\ 1\ (\text{mult } n\ (\text{fact } (\text{pred } n))))$$

into a higher-order function, by replacing the call to fact with a function f

$$\text{ffact} \equiv (\lambda f.(\lambda n.\text{if } (\text{zero } n)\ 1\ (\text{mult } n\ (f\ (\text{pred } n)))))$$

- Now, pass fact to ffact as a parameter, and do a β-reduction:

$$(\text{ffact fact}) \Rightarrow_\beta (\lambda n.\text{if } (\text{zero } n)\ 1\ (\text{mult } n\ (\text{fact } (\text{pred } n))))$$

Slide content credits: Christian Collberg

Fixed Point Combinators. . .

- But, the right-hand side of

$$(\text{ffact fact}) \Rightarrow_\beta (\lambda n.\text{if } (\text{zero } n)\ 1\ (\text{mult } n\ (\text{fact } (\text{pred } n))))$$

is just the body of fact

$$\text{fact} \equiv (\lambda n.\text{if } (\text{zero } n)\ 1\ (\text{mult } n\ (\text{fact } (\text{pred } n))))$$

so we can write the identity:

$$(\text{ffact fact}) = \text{fact}$$

- Thus, fact is a fixed point for ffact.

Fixed Point Combinators...

THE UNIVERSITY OF ARIZONA
Computer Science

- In lambda calculus, the fixed point combinator Y is defined as

$$\mathsf{Y} \equiv (\lambda h.((\lambda x.(h\ (x\ x)))\ (\lambda x.(h\ (x\ x)))))$$

- Let's see what happens when we apply that to an expression E:

$$
\begin{aligned}
&(\mathsf{Y}\ \ E) = \\
&\quad ((\lambda h.((\lambda x.(h\ \ (x\ \ x)))\ \ (\lambda x.(h\ \ (x\ \ x)))))\ \ E) \Rightarrow_\beta \\
&\quad ((\lambda x.(E\ \ (x\ \ x)))\ \ (\lambda x.(E\ \ (x\ \ x)))) \Rightarrow_\beta \\
&\quad (E\ \ ((\lambda x.(E\ \ (x\ \ x)))\ \ (\lambda x.(E\ \ (x\ \ x))))) = \\
&\quad (E\ \ (\mathsf{Y}\ \ E))
\end{aligned}
$$

-Spring 2005—23 [26]

Slide content credits: Christian Collberg

Fixed Point Combinators. . .

- So, we saw that

$$(\mathsf{Y}\ E) \Rightarrow_\beta^* (E\ (\mathsf{Y}\ E))$$

- In other words,

$$E(\mathsf{Y}E) = \mathsf{Y}E$$

or for any expression E, $\mathsf{Y}E$ is a fixed point for E.

Slide content credits: Christian Collberg

Fixed Point Combinators — Example

- Let's get back to our definition of fact:

$$\text{fact} \equiv (\lambda n.\text{if } (\text{zero } n)\ 1\ (\text{mult } n\ (\text{fact } (\text{pred } n))))$$

and the beta abstracted version ffact (we'll call it F for brevity)

$$\text{F} \equiv (\lambda f.(\lambda n.\text{if } (\text{zero } n)\ 1\ (\text{mult } n\ (f\ (\text{pred } n)))))$$

- And, so we can define

$$\text{fact} \equiv (\text{Y F})$$

- Let's try to evaluate

$$(\text{fact } 3)$$

Evaluating (fact 3)

THE UNIVERSITY OF ARIZONA
Computer Science

```
F = \f.(\n.(zero? n) 1 (* n (f (pred n))))
fact = (Y F)

fact 3
= (Y F) 3
= F (Y F) 3
= \f.(\n.(zero? n) 1 (* n (f (pred n)))) (Y F) 3
= (\n.(zero? n) 1 (* n ((Y F) (pred n))) 3
= (zero? 3) 1 (* 3 ((Y F) (pred 3))
= false 1 (* 3 ((Y F) (pred 3))
= (* 3 ((Y F) (pred 3))) ; in * def, 2nd param evaluated
= (* 3 (F (Y F) 2 ))
= (* 3 (* 2 ((Y F) (pred 2))))
= (* 3 (* 2 ( F (Y F) 1 )))
= (* 3 (* 2 (* 1 ((Y F) (pred 1)))))
= (* 3 (* 2 (* 1 (F (Y F) 0))))
= (* 3 (* 2 (* 1 (\n.(zero? n) 1 (* n ((Y F) (pred n)))
0)))))
= (* 3 (* 2 (* 1 1) ) )
```

Recursion for lambda calculus

- **References**
  - Christian Collberg slides (see his more detailed, and slightly cutoff on left, derivation after this slide)
  - See *https://medium.com/@ayanonagon/the-y-combinator-no-not-that-one-7268d8d9c46* for similar level of abstraction of (fact 3) evaluation we did in slide 13.

- **HW10 hints**
  - In the class recording from today, we went over hints for doing HW10.
  - The hints are not included in the posted slide deck.

xed Point Combinators — Example...

$$\mathsf{F} \equiv (\lambda f.(\lambda n.\mathsf{if}\ \ (\mathsf{zero}\ \ n)\ \ 1\ \ (\mathsf{mult}\ \ n\ \ (f\ \ (\mathsf{pred}\ \ n)))))$$
$$\mathsf{fact} \equiv (\mathsf{Y}\ \ \mathsf{F})$$
$$\mathsf{Y} \equiv (\lambda h.((\lambda x.(h\ \ (x\ \ x)))\ \ (\lambda x.(h\ \ (x\ \ x)))))$$

$$(\mathsf{fact}\ \ 3) = ((\mathsf{Y}\ \ \mathsf{F})\ \ 3) =$$

$$(((\lambda h.((\lambda x.(h\ \ (x\ \ x)))\ \ (\lambda x.(h\ \ (x\ \ x)))))\ \ \mathsf{F})\ \ 3) \Rightarrow_\beta$$

$$(((\lambda x.(F\ \ (x\ \ x)))\ \ (\lambda x.(F\ \ (x\ \ x))))\ \ 3) =$$

$$((K\ \ K)\ \ 3) = \ldots$$

- Where we've used the abbreviation

$$\mathsf{K} \equiv (\lambda x.(F\ (x\ x)))$$

xed Point Combinators — Example. . .

THE UNIVERSITY OF ARIZONA. Computer Science

$\mathsf{F} \equiv (\lambda f.(\lambda n.\mathsf{if}\ \ (\mathsf{zero}\ \ n)\ \ 1\ \ (\mathsf{mult}\ \ n\ \ (f\ \ (\mathsf{pred}\ \ n)))))$

$\mathsf{K} \equiv (\lambda x.(F\ \ (x\ \ x)))$

$((K\ \ K)\ \ 3) =$

$(((\lambda x.(F\ \ (x\ \ x)))\ \ K)\ \ 3) \Rightarrow_\beta$

$((F\ \ (K\ \ K))\ \ 3) =$

$(((\lambda f.(\lambda n.\mathsf{if}\ \ (\mathsf{zero}\ \ n)\ \ 1\ \ (\mathsf{mult}\ \ n\ \ (f\ \ (\mathsf{pred}\ \ n)))))\ \ (K\ \ K))\ \ 3) \Rightarrow_\beta$

$((\lambda n.\mathsf{if}\ \ (\mathsf{zero}\ \ n)\ \ 1\ \ (\mathsf{mult}\ \ n\ \ ((K\ \ K)\ \ (\mathsf{pred}\ \ n))))\ \ 3) \Rightarrow_\beta$

$\mathsf{if}\ \ (\mathsf{zero}\ \ 3)\ \ 1\ \ (\mathsf{mult}\ \ 3\ \ ((K\ \ K)\ \ (\mathsf{pred}\ \ 3))) \Rightarrow_\beta \ldots$

Slide content credits: Christian Collberg

xed Point Combinators — Example. . .

$\equiv (\lambda f.(\lambda n.\text{if}\ \ (\text{zero}\ \ n)\ \ 1\ \ (\text{mult}\ \ n\ \ (f\ \ (\text{pred}\ \ n)))))$

$\equiv (\lambda x.(F\ \ (x\ \ x)))$

$\equiv (\lambda l.(\lambda m.(\lambda n.((l\ \ m)\ \ n))))$

alse $\equiv (\lambda t.(\lambda f.f))$

$(K\ \ K)\ \ 3) \Rightarrow^*_\beta \text{if}\ \ (\text{zero}\ \ 3)\ \ 1\ \ (\text{mult}\ \ 3\ \ ((K\ \ K)\ \ (\text{pred}\ \ 3))) \Rightarrow_\beta$

$((\lambda l.(\lambda m.(\lambda n.((l\ \ m)\ \ n))))\ \ (\text{zero}\ \ 3)\ \ 1\ \ (\text{mult}\ \ 3\ \ ((K\ \ K)\ \ (\text{pred}\ \ 3)))) \Rightarrow^*_\beta$

$(\text{zero}\ \ 3)\ \ 1\ \ (\text{mult}\ \ 3\ \ ((K\ \ K)\ \ (\text{pred}\ \ 3) \Rightarrow_\delta$

$\text{false}\ \ 1\ \ (\text{mult}\ \ 3\ \ ((K\ \ K)\ \ (\text{pred}\ \ 3) =$

$((\lambda t.(\lambda f.f))\ \ 1\ \ (\text{mult}\ \ 3\ \ ((K\ \ K)\ \ (\text{pred}\ \ 3)) \Rightarrow_\beta \ldots$

Slide content credits: Christian Collberg

xed Point Combinators — Example. . .

$\equiv (\lambda f.(\lambda n.\text{if (zero } n) \; 1 \; (\text{mult } n \; (f \; (\text{pred } n)))))$

$\equiv (\lambda x.(F \; (x \; x)))$

$(K \; K) \; 3) \Rightarrow_\beta^* ((\lambda t.(\lambda f.f)) \; 1 \; (\text{mult } 3 \; ((K \; K) \; (\text{pred } 3)) \Rightarrow_\beta$

$\text{ult } 3 \; ((K \; K) \; (\text{pred } 3)) \Rightarrow_\delta$

$\text{ult } 3 \; ((K \; K) \; 2) =$

$\text{ult } 3 \; (((\lambda x.(F \; (x \; x))) \; K) \; 2) \Rightarrow_\beta$

$\text{ult } 3 \; ((F \; (K \; K)) \; 2) =$

$\text{ult } 3 \; (((\lambda f.(\lambda n.\text{if (zero } n) \; 1 \; (\text{mult } n \; (f \; (\text{pred } n))))) \; (K \; K)) \; 2) \Rightarrow_\beta^* 6$

Slide content credits: Christian Collberg