CS 152: *Programming Language Paradigms*

# More Higher Order Functions

Prof. Tom Austin
San José State University

# Map/filter lab review

# Additional higher-order functions
(in class)

# Fold variants

- `foldr`
  - Traverses from the right
  - `(foldr * 1 '(2 4 8))` is the same as `(* 2 (* 4 (* 8 1)))`
- `foldl`
  - Traverses from the left
  - `(foldl * 1 '(2 4 8))` is the same as `(* 8 (* 4 (* 2 1)))`
- WARNING: Different languages define fold slightly differently.

# `foldr` evaluation

```
   (foldr cons '() '(1 2 3))
-> (cons 1 (foldr cons '() '(2 3)))
-> (cons 1 (cons 2 (foldr cons '() '(3))))
-> (cons 1 (cons 2 (cons 3 (foldr cons '()
                                  '()))))
-> (cons 1 (cons 2 (cons 3 '())))
-> (cons 1 (cons 2 '(3)))
-> (cons 1 '(2 3))
-> '(1 2 3)
```

# `foldl` evaluation


```
    (foldl cons '() '(1 2 3))
-> (foldl cons (cons 1 '()) '(2 3))
-> (foldl cons '(1) '(2 3))
-> (foldl cons (cons 2 '(1)) '(3))
-> (foldl cons '(2 1) '(3))
-> (foldl cons (cons 3 '(2 1)) '())
-> (foldl cons '(3 2 1) '())
-> '(3 2 1)
```

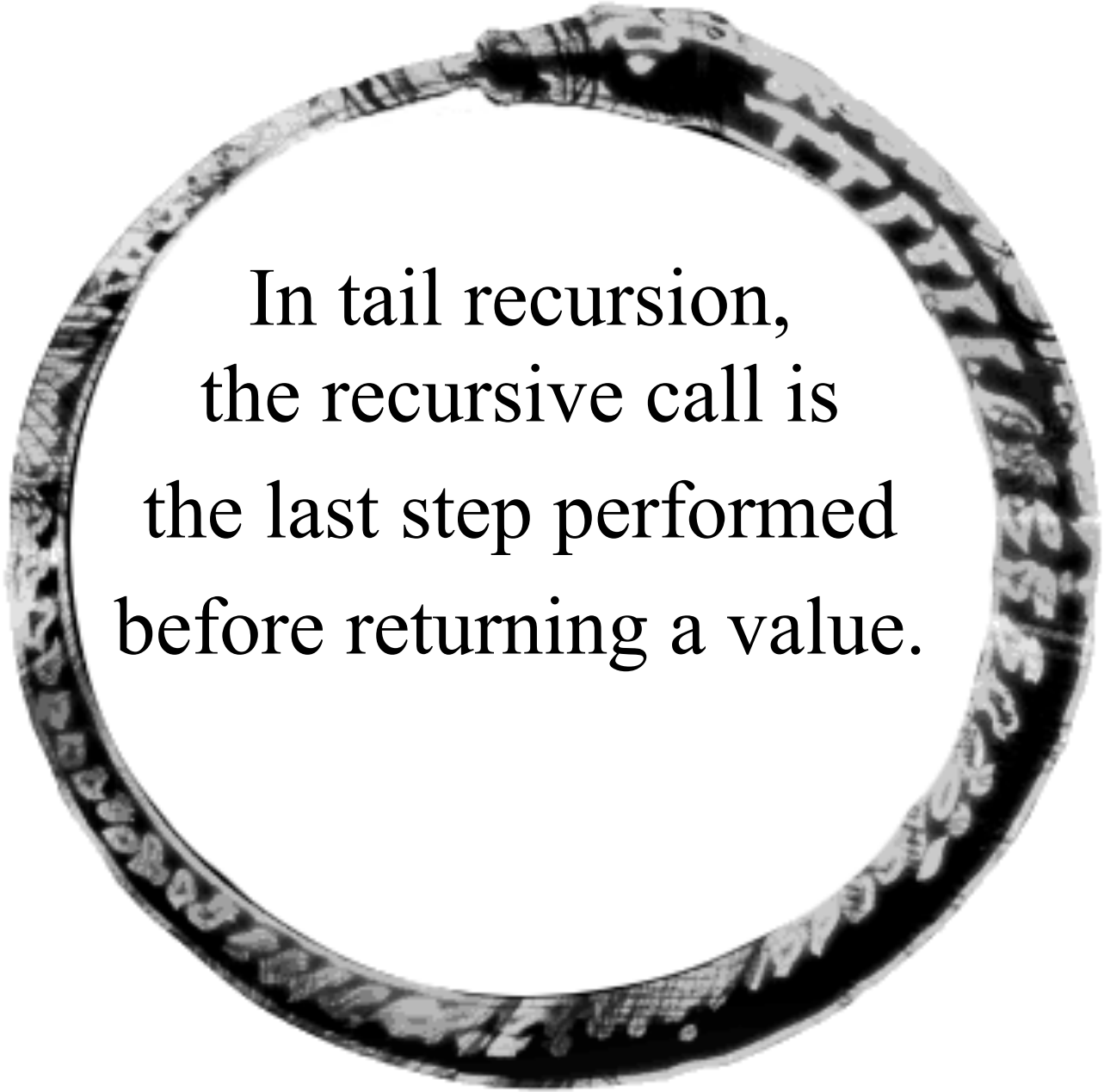
# Which fold?

- When in doubt, use `foldl`.
- Use `foldr` when you need to traverse the list **right-to-left**.
- Remember – other languages may define these differently.

# Tail Recursion

Iterative solutions tend to be more efficient than recursive solutions.

However, compilers are very good at optimizing a *tail recursive* functions.



In tail recursion,
the recursive call is
the last step performed
before returning a value.

# Is this function tail-recursive?

```
public int factorial(int n) {
    if (n==1) return 1;
    else {
        return n * factorial(n-1);
    }
}
```

No: the last step is multiplication

# Is this function tail-recursive?

```
public int factorialAcc(int n, int acc){
    if (n==1) return acc;
    else {
        return factorialAcc(n-1, n*acc);
    }
}
```

Yes: the recursive step is the last thing we do

# Which version is tail-recursive?

```
(define (fact n)
 (if (= n 1)
   1
   (* n
      (fact
       (- n 1)))))
```

```
(define (fact n a)
 (if (= n 1)
   a
   (fact
     (- n 1)
     (* n a))))
```

# Lab 4: `foldl` and `foldr`

See Canvas for details.