Programming Paradigms

Functional Languages (Part 3)

Prof. Dr. Michael Pradel

Software Lab, University of Stuttgart

Summer 2020

Plan for Today

- **Introduction**
- **A Bit of Scheme**
- **Evaluation Order** ←

Evaluation Order

- **In what order to evaluate subcomponents of an expression?**
  - Applicative-order: Evaluate arguments before passing them to the function
  - Normal-order: Pass arguments unevaluated and evaluate once used

```
(define double (lambda (x) (+ x x)))
```

Applicative-order

```
(double (* 3 4))
⇒ (double 12)
⇒ + 12 12
⇒ 24
```

Normal-order

```
(double (* 3 4))
⇒ + (* 3 4) (* 3 4)
⇒ + 12 (* 3 4)
⇒ + 12 12
⇒ 24
```

Doing extra-work with normal order!

```
(define switch (lambda (x a b c)
    (cond ((< x 0) a)
          ((= x 0) b)
          ((> x 0) c))))
```

→ Doing extra-work with applicative order!

Applicative-order

```
(switch -1 (+ 1 2) (+ 2 3) (+ 3 4))
⇒ (switch -1 3 (+ 2 3) (+ 3 4))
⇒ (switch -1 3 5 (+ 3 4))
⇒ (switch -1 3 5 7)
⇒ (cond ((< -1 0) 3)
        ((= -1 0) 5)
        ((> -1 0) 7))
⇒ (cond (#t 3
         ⋮  )
⇒ 3
```

Normal-order

```
(switch -1 (+ 1 2) (+ 2 3) (+ 3 4))
⇒ (cond ((< -1 0) (+ 1 2))
        ((= -1 0) (+ 2 3))
        ((> -1 0) (+ 3 4)))
⇒ (cond (#t (+ 1 2)
         ⋮          )
⇒ (+ 1 2)
⇒ 3
```

Impact on Correctness

- **Evaluation order also affects correctness**
- **E.g., runtime error when evaluating an ”unneeded” subexpression**
  - Terminates program in applicative-order
  - Not noticed in normal-order

Lazy Evaluation

- **Evaluate subexpressions on-demand**
- **Avoid re-evaluating the same expression**
  - Memorize its result
- **Transparent to programmer only in PL without side effects, e.g., Haskell**
  - In PLs with side effects, e.g., Scheme: Programmer can explicitly ask for lazy evaluation with `delay`

Quiz: Evaluation Order

```
(define double (lambda (x) (+ x x)))
(define avg (lambda (x y) (/(+ x y) 2)))
```

How many evaluation steps are needed to evaluate

```
(double(avg 2 4))
```

under applicative-order and normal-order evaluation?

Please vote via Ilias.

Quiz: Evaluation Order

```
(define double (lambda (x) (+ x x)))
(define avg (lambda (x y) (/(+ x y) 2)))
```

How many evaluation steps are needed to evaluate

```
(double(avg 2 4))
```

under applicative-order and normal-order evaluation?

5 and 8

Please vote via Ilias.

Applicative-order

```
(double (avg 2 4))
⇒ (double (/ (+ 2 4) 2))
⇒ (double (/ 6 2))
⇒ (double 3)
⇒ (+ 3 3)
⇒ 6
```

5 steps

Normal-order

```
(double (avg 2 4))
⇒ (+ (avg 2 4) (avg 2 4))
⇒ (+ (/ (+ 2 4) 2) (avg 2 4))
⇒ (+ (/ 6 2) (avg 2 4))
⇒ (+ 3 (avg 2 4))
⇒ (+ 3 (/ (+ 2 4) 2))
⇒ (+ 3 (/ 6 2))
⇒ (+ 3 3)
⇒ 6
```

8 steps

Plan for Today

- **Introduction**
- **A Bit of Scheme**
- **Evaluation Order** ✔