Programming Paradigms

Functional Languages (Part 1)

Prof. Dr. Michael Pradel

Software Lab, University of Stuttgart

Summer 2020

Wake-up Exercise

What does the following Scheme code evaluate to?

```
(let ((a 3))
  (let ((a 4)
        (b a))
    (+ a b)))
```

Please vote via Ilias.

Wake-up Exercise

What does the following Scheme code evaluate to?

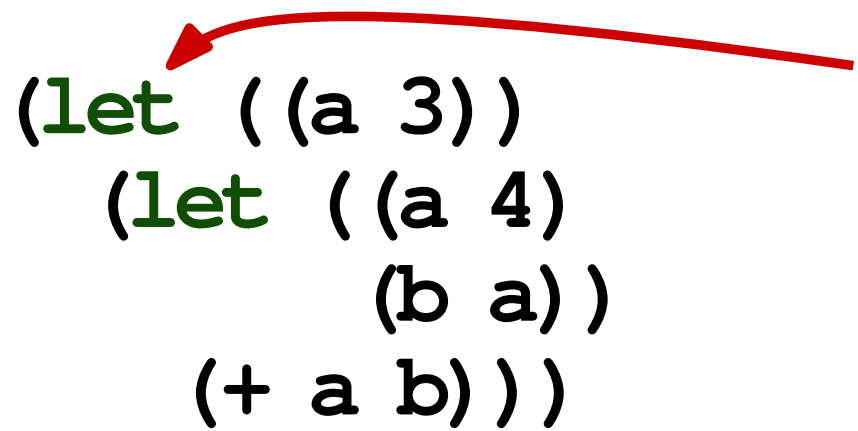
```
(let ((a 3))
  (let ((a 4)
        (b a))
    (+ a b)))
```

Result: 7

Please vote via Ilias.

Wake-up Exercise

What does the following Scheme code evaluate to?



`let` **binds names to values**

Result: 7

Please vote via Ilias.

Wake-up Exercise

What does the following Scheme code evaluate to?

```
(let ((a 3))
  (let ((a 4)
        (b a))
    (+ a b)))
```

`let` binds names to values

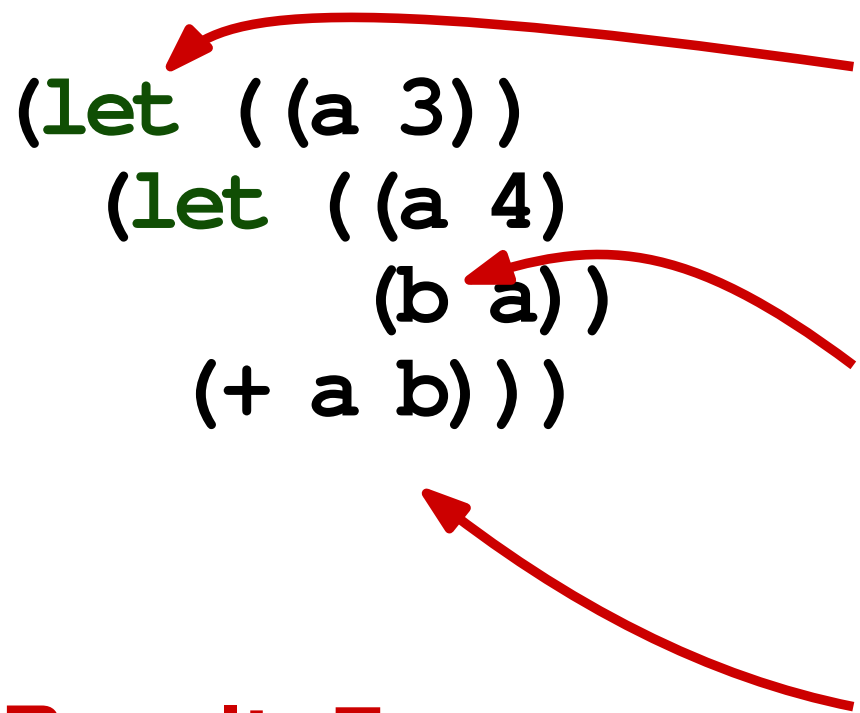
Scope of bindings: Second argument only

Result: 7

Please vote via Ilias.

Wake-up Exercise

What does the following Scheme code evaluate to?



`let` binds names to values

`b` takes the value of the outer `a`

Scope of bindings: Second argument only

Result: 7

Please vote via Ilias.

Functional Languages

- **Functional paradigm: Alternative to imperative PLs**
  - Output: Mathematical function of input
  - No internal state, no side effects
- **In practice: Fuzzy boundaries**
  - "Functional" features in many "imperative" PLs
    - E.g., higher-order functions
  - "Imperative features" in many "functional" PLs
    - E.g., assignment and iteration

Historical Origins

- **Lambda calculus**
  - Alonzo Church, 1930s
- **Express computation based on**
  - Abstraction into functions
    - E.g., $(\lambda x.M)$
  - Function application
    - E.g., $(M\ N)$

Features

- First-class function values and higher-order function
- Extensive polymorphism
- List types and operators
- Structured function returns
- Constructors for structured objects
- Garbage collection

Features

- First-class function values and higher-order function
- Extensive polymorphism
- List types and operators
- Structured function returns
- Constructors for structured objects
- Garbage collection

Functions assigned to variables, passed as arguments, or return values

Features

- First-class function values and higher-order function
- Extensive polymorphism
- List types and operators
- Structured function returns
- Constructors for structured objects
- Garbage collection

Use a function on different kinds of values, e.g., using type inference

Features

- First-class function values and higher-order function
- Extensive polymorphism
- List types and operators
- Structured function returns
- Constructors for structured objects
- Garbage collection

Ideal for recursion (handle first element and then recursively the remainder)

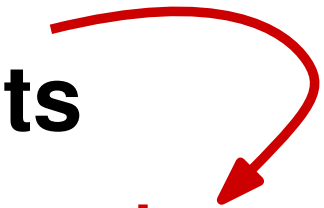
Features

- First-class function values and higher-order function
- Extensive polymorphism
- List types and operators
- Structured function returns
- Constructors for structured objects
- Garbage collection

Functions can return any structured data, e.g., lists and functions

Features

- First-class function values and higher-order function
- Extensive polymorphism
- List types and operators
- Structured function returns
- Constructors for structured objects
- Garbage collection



Construct aggregate objects inline and all-at-once

Features

- First-class function values and higher-order function
- Extensive polymorphism
- List types and operators
- Structured function returns
- Constructors for structured objects
- Garbage collection

Necessary because evaluation tends to create lots of temporary data

Purely Functional PLs

- **Functions depend only on their parameters**
  - Not on any other global or local state
  - Order of evaluation is irrelevant
    - Eager and lazy evaluation yield same result
- **E.g., Haskell**
  - By Philip Wadler et al., first released in 1990
  - Actively used as a research language

Non-Pure Functional PLs

- **Mix of functional features with assignments**
- **E.g., Scheme**
  - Dialect of Lisp
  - By Guy Steele and Gerlad Jay Sussman (MIT)
- **E.g., OCaml**
  - Extends ML with OO features
  - Developed at INRIA (France)