CLOSURES

What we know about Lambda Expressions

- Lambda expressions are used to create procedures.
- They bind variables.

What needs to happen when processing a Lambda Expression?

- Store the parameters of the lambda expression
- Store the body of the lambda expression (but do not evaluated it yet.)
- Keep track of the current environment (as we may have to extend it)
- We will store these three items in what we call a "*closure*".

A Simple Example

- Consider: **`(lambda (x) x)`**
- Create a structure that has three parts:
  1) A list of parameters: **`(x)`**
  2) The body: **`x`**
  3) The current environment: **`empty-env`**
- Closure:

(x)	x	( )

More Interesting Example

- `(lambda (x y) (+ x y))`
- Closure:

`(x y)`	`(+ x y)`	`()`

Even More Interesting Example

- `(lambda (x y) (lambda (a b) (+ a b x y)))`
- Closure:

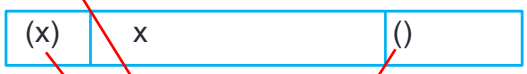
`(x y)`	`(lambda (a b) (+ a b x y))`	`()`

Closures and Environments

- A lambda expression is converted into a *closure*.
- An application-lambda expression is used to create an *environment*.
- Then the body is evaluated.
- Let's consider examples where we have both, to see how they interact.

Simple Application Expression

- ((lambda (x) x) 4)
- Closure:



- Environment:

- Environments are built based on closures.
- We use the parameter list from the closure as the symbol list for the environment to be extended.
- We use the list of arguments from the application expression as the list of values for the environment to be extended.
- We use the environment stored in the closure as the one to be extended.

What's Next After Creating an Environment?

- Evaluate the body of the closure

Evaluating: ((lambda (x) x) 4)

- The parser returns:

```
(app-exp (lambda-exp (x) (var-exp x)) (lit-exp 4))
```

Evaluating: ((lambda (x) x) 4)

- The parser returns:

```
(app-exp (lambda-exp (x) (var-exp x))(lit-exp 4))
```

- The value of the lambda expression is the following closure:

(x)	(var-exp x)	()

- The value of the lit-expression is: `4`

Evaluating: ((lambda (x) x) 4)

- The parser returns:

```
(app-exp (lambda-exp (x) (var-exp x))(lit-exp 4))
```

- The value of the lambda expression is the following closure:

(x)	(var-exp x)	()

- The value of the lit-expression is: `4`
- Next, we extend the environment as captured in the closure:

	()

Evaluating: ((lambda (x) x) 4)

- The parser returns:

```
(app-exp (lambda-exp (x) (var-exp x))(lit-exp 4))
```

- The value of the lambda expression is the following closure:

- The value of the lit-expression is: `4`
- Next, we extend the environment as captured in the closure:

Evaluating: ((lambda (x) x) 4)

- The parser returns:

```
(app-exp (lambda-exp (x) (var-exp x))(lit-exp 4))
```

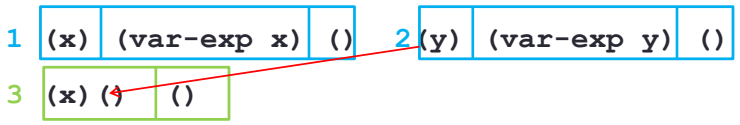
- The value of the lambda expression is the following closure:

- The value of the lit-expression is: `4`
- Next, we extend the environment as captured in the closure:

- Next, evaluate the body captured in the closure:
  `(var-exp x)`
- It evaluates to: 4

Evaluating: (((lambda (x) x) (lambda (y) y)) 4)

```
(app-exp
      (app-exp
            (lambda-exp x (var-exp x))
            (lambda-exp y (var-exp y)))
      (lit-exp 4))
```

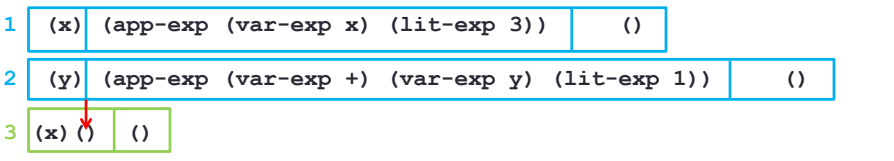


4 evaluate: `(var-exp x)` value: closure 2

6 evaluate: `(var-exp y)` value: 4

Evaluating: ((lambda (x) (x 3)) (lambda (y) (+ y 1)))

```
(app-exp
  (lambda-exp (x) (app-exp (var-exp x) (lit-exp 3)))
  (lambda-exp (y) (app-exp (var-exp +) (var-exp y) (lit-exp 1))))
```



4 Evaluate: (app-exp (var-exp x) (lit-exp 3))

6 Evaluate: (app-exp (var-exp +) (var-exp y) (lit-exp 1))))

Evaluating: ((lambda (x) ((lambda (y) (+ y 1)) (+ x 1))) 4)

```
(app-exp
  (lambda-exp (x)
    (app-exp
       (lambda-exp (y)
          (app-exp (var-exp +) (var-exp y) (lit-exp 1)))
       (app-exp (var-exp +) (var-exp x) (lit-exp 1))))
  (lit-exp 4))
```

1 | (x) | (app-exp (lambda-exp (y) …) | () |

2 | (x) (4) | () |

3 Evaluate:

4 | (y) | (app-exp (var-exp +) …) | |

5 Evaluate: (app-exp (var-exp +) (var-exp x) (lit-exp 1))
Value: 5

6 | (y) (5) | (x) (4) | () |

7 Evaluate: (app-exp (var-exp +) (var-exp y) (lit-exp 1)))
Value: 6

More Interesting App Exp

(((lambda (x y) (lambda (a b) (+ a b x y))) 3 4) 5 6)

c1: | (x y) | (lambda (a b) (+ a b x y)) | () |

e1: | (x y)(3 4) | () |

c2: | (a b) | (+ a b x y) | |

e2: | (a b) (5 6) | (x y)(3 4) | () |

(<primitive proc +> 5 6 3 4) => 18

Review of App and Lambda Expressions

- A Lambda expression is evaluated to a closure
- A closure holds the:
  - Parameters specified by the lambda expression,
  - The body of the lambda expression, and the
  - Current environment
- Based on a closure and an appropriate number of arguments, an application expression extends an environment
- The new environment holds the:
  - Parameters found in the closure and
  - The values provided by the application expression.
  - It extends the environment that was stored in the closure
- An application expression then evaluates the body as found in the closure.