爱扑bug的熊

SICP

God’s Programming Book

Lecture-24 Interpreters

爱扑bug的熊

Interpreters

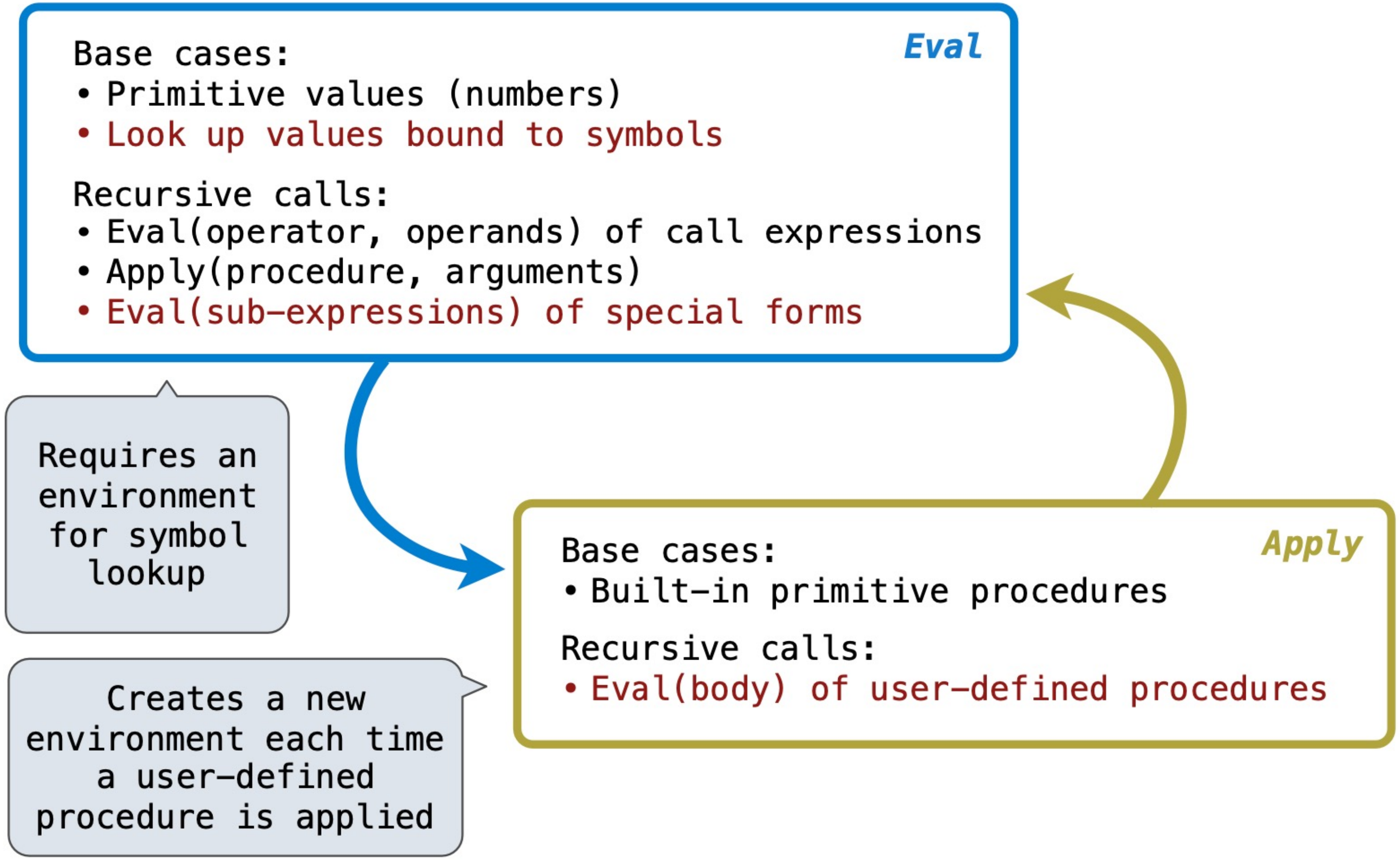
Slides Adapted from cs61a of UC Berkeley

Interpreting Scheme

cc BY NC SA

爱扑bug的熊

The Structure of an Interpreter



Eval

Base cases:

- Primitive values (numbers)
- Look up values bound to symbols

Recursive calls:

- Eval(operator, operands) of call expressions
- Apply(procedure, arguments)
- Eval(sub-expressions) of special forms

Apply

Base cases:

- Built-in primitive procedures

Recursive calls:

- Eval(body) of user-defined procedures

Requires an environment for symbol lookup

Creates a new environment each time a user-defined procedure is applied

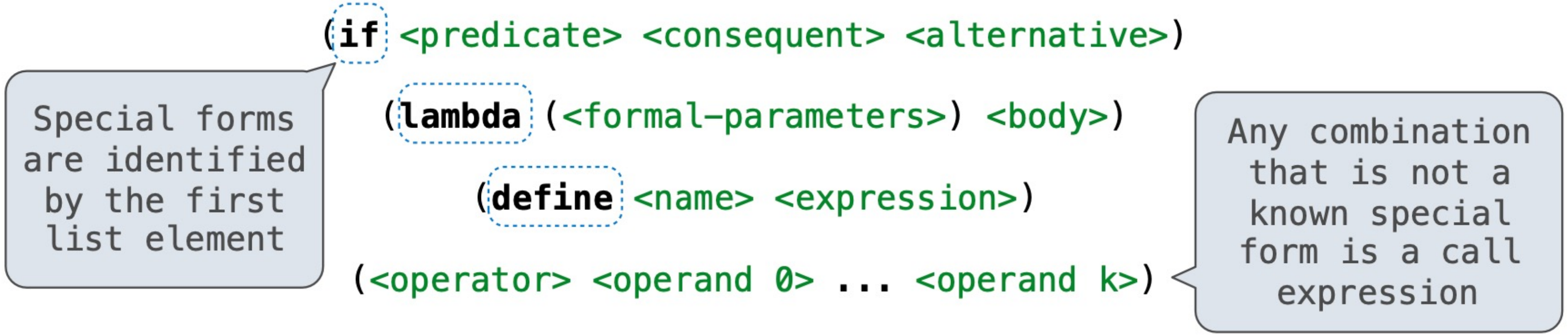
Special Forms

爱扑bug的熊

Scheme Evaluation

The scheme_eval function choose behavior based on expression form:

- Symbols are looked up in the current environment
- Self-evaluating expressions are returned as values
- All other legal expressions are represented as Scheme lists, called combinations



```
(if <predicate> <consequent> <alternative>)
(lambda (<formal-parameters>) <body>)
(define <name> <expression>)
(<operator> <operand 0> ... <operand k>)
```

Special forms are identified by the first list element

Any combination that is not a known special form is a call expression

```
(define (demo s) (if (null? s) '(3) (cons (car s) (demo (cdr s))) ))

(demo (list 1 2))
```

Logical Forms

cc BY NC SA

爱扑bug的熊

Logical Special Forms

Logical forms may only evaluate some sub-expressions

- **If** expression: (**if** <predicate> <consequent> <alternative>)
- **And** and **or:** (**and** <e1> … <en>), (**or** <e1> … <en>)
- **Cond** expression: (**cond** (<p1> <e1>) … (<pn> <en>) (else <e>))

The value of an if expression is the value of a sub-expression:

- Evaluate the predicate
- Choose a sub-expression: <consequent> or <alternative>
- Evaluate that sub-expression to get the value of the whole expression

do_if_form

Quotation

爱扑bug的熊

Quotation

The quote special form evaluates to the quoted expression, which is not evaluated

```
(quote <expression>)
```

```
(quote (+ 1 2))
```

evaluates to the three-element Scheme list

```
(+ 1 2)
```

The <expression> itself is the value of the whole quote expression

```
'<expression> is shorthand for (quote <expression>)

          (quote (1 2))        is equivalent to        '(1 2)
```

The scheme_read parser converts shorthand ' to a combination that starts with quote

Lambda Expressions

爱扑bug的熊

Lambda Expressions

Lambda expressions evaluate to user-defined procedures

```
(lambda (<formal-parameters>) <body>)

(lambda (x) (* x x))
```

```
class LambdaProcedure:
    def __init__(self, formals, body, env):
        self.formals = formals ........................ A scheme list of symbols
        self.body = body .............................. A scheme list of expressions
        self.env = env ................................ A Frame instance
```

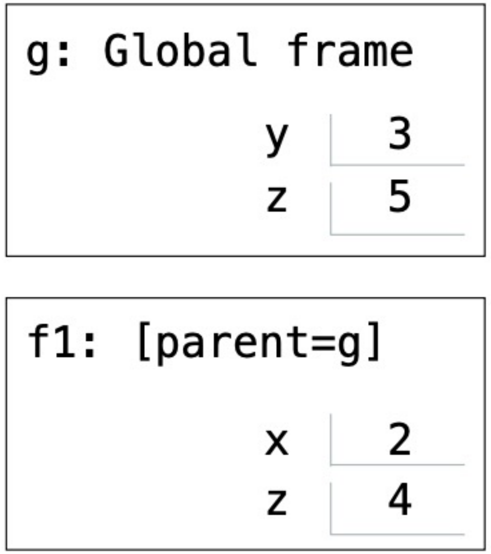
爱扑bug的熊

Frames and Environments

A frame represents an environment by having a parent frame

Frames are Python instances with methods **lookup** and **define**

In Project 4, Frames do not hold return values



g: Global frame

y	3
z	5

f1: [parent=g]

x	2
z	4

Define Expressions

cc BY NC SA

爱扑bug的熊

Define Expressions

Define binds a symbol to a value in the first frame of the current environment.

```
(define <name> <expression>)
```

1. Evaluate the <expression>
2. Bind <name> to its value in the current frame

```
(define x (+ 1 2))
```

Procedure definition is shorthand of define with a lambda expression

```
(define (<name> <formal parameters>) <body>)

(define <name> (lambda (<formal parameters>) <body>))
```

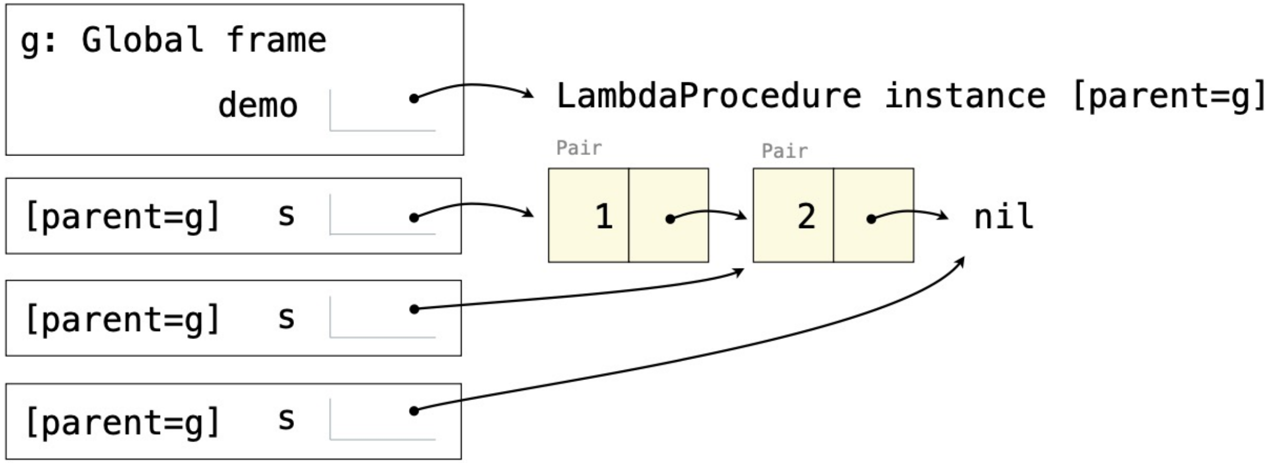
Applying User-Defined Procedures

To apply a user-defined procedure, create a new frame in which formal parameters are bound to argument values, whose parent is the **env** attribute of the procedure

Evaluate the body of the procedure in the environment that starts with this new frame

```
(define (demo s) (if (null? s) '(3) (cons (car s) (demo (cdr s)))))

(demo (list 1 2))
```



CC BY NC SA

Thanks for Listening

cc BY NC SA