Examples of Environments

CS 152 -- Programming Paradigms

San José State University

Michael McThrow

September 16, 2020

When loading the Scheme interpreter, this is the state of the environments that the interpreter maintains:

Global Environment Frame

Name	Value
`define`	`DEFINE-BUILTIN`
`lambda`	`LAMBDA-BUILTIN`
`+`	`PLUS-BUILTIN`
`-`	`MINUS-BUILTIN`
`...`	`...`

Let's run the expression `(define x 10)`

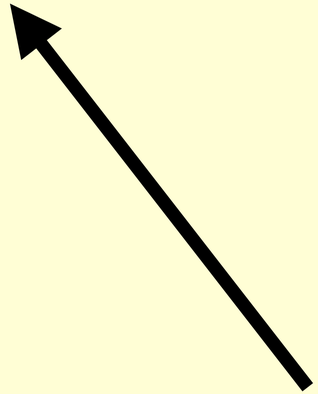
Global Environment Frame

Name	Value
`define`	`DEFINE-BUILTIN`
`lambda`	`LAMBDA-BUILTIN`
`+`	`PLUS-BUILTIN`
`-`	`MINUS-BUILTIN`
`...`	`...`

Let's run the expression `(define x 10)`

Global Environment Frame

Name	Value
`define`	`DEFINE-BUILTIN`
`lambda`	`LAMBDA-BUILTIN`
`+`	`PLUS-BUILTIN`
`-`	`MINUS-BUILTIN`
`...`	`...`



E1 Frame

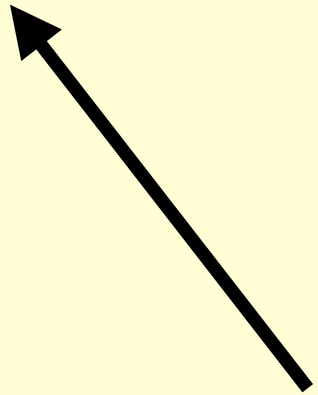
Name	Value

We know that `(define x 10)` is a function call since it is an unquoted list. To begin, we first create a new environment for evaluating this function. This environment is encompassed by the global environment, and the environment's frame points to the global environment.

Let's run the expression `(define x 10)`

Global Environment Frame

Name	Value
`define`	`DEFINE-BUILTIN`
`lambda`	`LAMBDA-BUILTIN`
`+`	`PLUS-BUILTIN`
`-`	`MINUS-BUILTIN`
`...`	`...`



E1 Frame

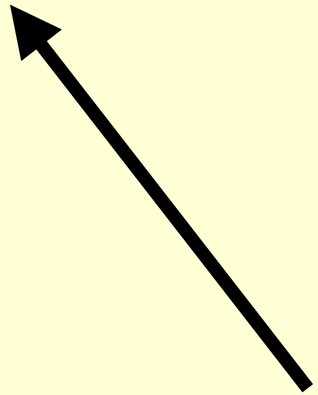
Name	Value

Next, we do a lookup for `define`. We search the E1 environment frame first for the name `define`. Since it does not show up there, we go to its encompassing environment, the global environment, and search its frame. `define` shows up as a built-in function.

Let's run the expression `(define x 10)`

Global Environment Frame

Name	Value
`define`	`DEFINE-BUILTIN`
`lambda`	`LAMBDA-BUILTIN`
`+`	`PLUS-BUILTIN`
`-`	`MINUS-BUILTIN`
`...`	`...`
`x`	`10`



E1 Frame

Name	Value

`define` assigns the name `x` to the value `10`. We first evaluate the expression `10`, resulting in `10`. Then we store `x` and its value `10` in the global environment frame.

Once `(define x 10)` has completed, the E1 environment goes away since it is no longer needed. The global environment stays throughout the lifetime of the interpreter.

Expressions Evaluated:

`(define x 10)`

Global Environment Frame

Name	Value
`define`	`DEFINE-BUILTIN`
`lambda`	`LAMBDA-BUILTIN`
`+`	`PLUS-BUILTIN`
`-`	`MINUS-BUILTIN`
`...`	`...`
`x`	`10`

Let's now run the expression x

Expressions Evaluated:

`(define x 10)`

Global Environment Frame

Name	Value
define	DEFINE-BUILTIN
lambda	LAMBDA-BUILTIN
+	PLUS-BUILTIN
-	MINUS-BUILTIN
...	...
x	10

Let's now run the expression x

Expressions Evaluated:

```
(define x 10)
```

Global Environment Frame

Name	Value
define	DEFINE-BUILTIN
lambda	LAMBDA-BUILTIN
+	PLUS-BUILTIN
-	MINUS-BUILTIN
...	...
x	10

1. Search the current environment's frame for the name x.

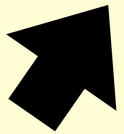
Let's now run the expression x

Expressions Evaluated:

`(define x 10)`

Global Environment Frame

Name	Value
`define`	`DEFINE-BUILTIN`
`lambda`	`LAMBDA-BUILTIN`
`+`	`PLUS-BUILTIN`
`-`	`MINUS-BUILTIN`
`...`	`...`
`x`	`10`



1. Search the current environment's frame for the name x.
2. Since x exists in the frame, retrieve its value 10.

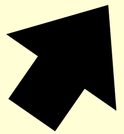
Let's now run the expression x

Expressions Evaluated:

`(define x 10)`

Global Environment Frame

Name	Value
`define`	`DEFINE-BUILTIN`
`lambda`	`LAMBDA-BUILTIN`
`+`	`PLUS-BUILTIN`
`-`	`MINUS-BUILTIN`
`...`	`...`
`x`	`10`



1. Search the current environment's frame for the name x.
2. Since x exists in the frame, retrieve its value 10.
3. x evaluates to 10.

Let's now run the expression `(define (sqrt x) (expt x 0.5))`

Expressions Evaluated:

```
(define x 10)
x
```

Global Environment Frame

Name	Value
define	DEFINE-BUILTIN
lambda	LAMBDA-BUILTIN
+	PLUS-BUILTIN
-	MINUS-BUILTIN
...	...
x	10

Let's now run the expression `(define (sqrt x) (expt x 0.5))`

Expressions Evaluated:

```
(define x 10)
x
```

Global Environment Frame

Name	Value
define	DEFINE-BUILTIN
lambda	LAMBDA-BUILTIN
+	PLUS-BUILTIN
-	MINUS-BUILTIN
...	...
x	10

Recall that `(define (sqrt x) (expt x 0.5))` is syntactic sugar for `(define sqrt (lambda (x) (expt x 0.5)))`

Let's now run the expression `(define (sqrt x) (expt x 0.5))`

Expressions Evaluated:

```
(define x 10)
x
```

Global Environment Frame

Name	Value
define	DEFINE-BUILTIN
lambda	LAMBDA-BUILTIN
+	PLUS-BUILTIN
-	MINUS-BUILTIN
...	...
x	10

Recall that `(define (sqrt x) (expt x 0.5))` is syntactic sugar for `(define sqrt (lambda (x) (expt x 0.5)))`
Therefore, we assign the lambda expression to the name sqrt.

Let's now run the expression `(define (sqrt x) (expt x 0.5))`

Expressions Evaluated:

```
(define x 10)
x
```

Global Environment Frame

Name	Value
define	DEFINE-BUILTIN
lambda	LAMBDA-BUILTIN
+	PLUS-BUILTIN
-	MINUS-BUILTIN
...	...
x	10
sqrt	(lambda (x) (expt x 0.5))

Recall that `(define (sqrt x) (expt x 0.5))` is syntactic sugar for `(define sqrt (lambda (x) (expt x 0.5)))`
Therefore, we assign the lambda expression to the name `sqrt`. We will skip over the temporary environment creation for this example.

Let's now run the expression `(sqrt 3)`

Expressions Evaluated:

```
(define x 10)
x
(define (sqrt x)
  (expt x 0.5))
```

Global Environment Frame

Name	Value
define	DEFINE-BUILTIN
lambda	LAMBDA-BUILTIN
+	PLUS-BUILTIN
-	MINUS-BUILTIN
...	...
x	10
sqrt	(lambda (x) (expt x 0.5))

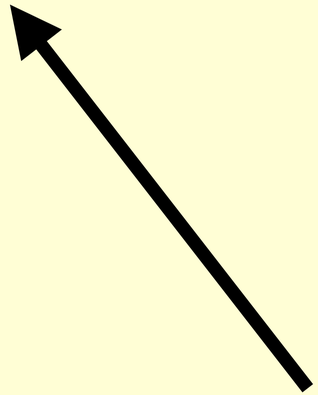
Let's now run the expression `(sqrt 3)`

Expressions Evaluated:

```
(define x 10)
x
(define (sqrt x)
  (expt x 0.5))
```

Global Environment Frame

Name	Value
define	DEFINE-BUILTIN
lambda	LAMBDA-BUILTIN
+	PLUS-BUILTIN
-	MINUS-BUILTIN
...	...
x	10
sqrt	(lambda (x) (expt x 0.5))



E1 Frame

Name	Value

We know that `(sqrt 3)` is a function call since it is an unquoted list. To begin, we first create a new environment for evaluating this function. This environment is encompassed by the global environment, and the environment's frame points to the global environment.

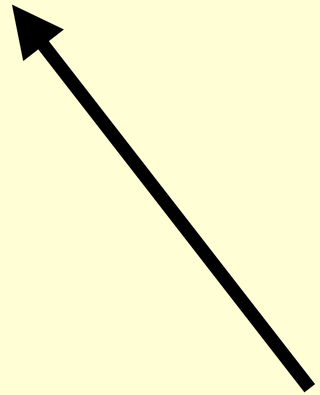
Let's now run the expression `(sqrt 3)`

Expressions Evaluated:

```
(define x 10)
x
(define (sqrt x)
  (expt x 0.5))
```

Global Environment Frame

Name	Value
define	DEFINE-BUILTIN
lambda	LAMBDA-BUILTIN
+	PLUS-BUILTIN
-	MINUS-BUILTIN
...	...
x	10
sqrt	(lambda (x) (expt x 0.5))



E1 Frame

Name	Value

```
(lambda (x) (expt x 0.5))
```

Next, perform lookup of sqrt. Since sqrt is not in the E1 frame, we go to the global environment frame to find sqrt. We then retrieve its value.

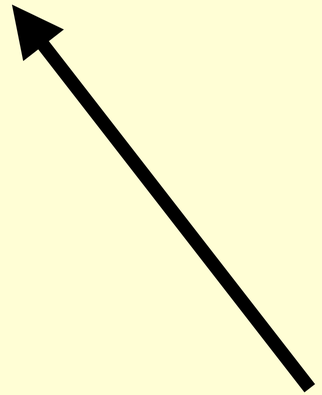
Let's now run the expression `(sqrt 3)`

Expressions Evaluated:

```
(define x 10)
x
(define (sqrt x)
  (expt x 0.5))
```

Global Environment Frame

Name	Value
define	DEFINE-BUILTIN
lambda	LAMBDA-BUILTIN
+	PLUS-BUILTIN
-	MINUS-BUILTIN
...	...
x	10
sqrt	(lambda (x) (expt x 0.5))



E1 Frame

Name	Value
x	3

```
(lambda (x) (expt x 0.5))
```

Since it is a lambda expression, we create a new entry in the current frame, where x (the parameter of the lambda expression) is set to 3 (the argument of the function call).

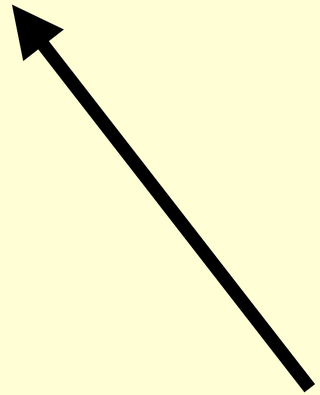
Let's now run the expression `(sqrt 3)`

Expressions Evaluated:

```
(define x 10)
x
(define (sqrt x)
  (expt x 0.5))
```

Global Environment Frame

Name	Value
define	DEFINE-BUILTIN
lambda	LAMBDA-BUILTIN
+	PLUS-BUILTIN
-	MINUS-BUILTIN
...	...
x	10
sqrt	(lambda (x) (expt x 0.5))



E1 Frame

Name	Value
x	3

```
(expt 3 0.5)
```

We then evaluate the body of the lambda expression, replacing x with 3. This happens to be another function call....

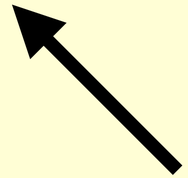
Let's now run the expression `(sqrt 3)`

Expressions Evaluated:

```
(define x 10)
x
(define (sqrt x)
  (expt x 0.5))
```

Global Environment Frame

Name	Value
define	DEFINE-BUILTIN
lambda	LAMBDA-BUILTIN
+	PLUS-BUILTIN
-	MINUS-BUILTIN
...	...
x	10
sqrt	(lambda (x) (expt x 0.5))



E1 Frame

Name	Value
x	3

E2 Frame

Name	Value

`(expt 3 0.5)`

And so we create another environment that is encompassed by the E1 environment.

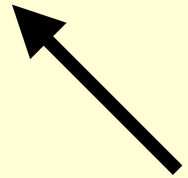
Let's now run the expression `(sqrt 3)`

Expressions Evaluated:

```
(define x 10)
x
(define (sqrt x)
  (expt x 0.5))
```

Global Environment Frame

Name	Value
define	DEFINE-BUILTIN
lambda	LAMBDA-BUILTIN
+	PLUS-BUILTIN
-	MINUS-BUILTIN
...	...
x	10
sqrt	(lambda (x) (expt x 0.5))



E1 Frame

Name	Value
x	3

E2 Frame

Name	Value

`(expt 3 0.5)`

We then do a lookup for expt, first looking through the E2 frame, then through the E1 frame, and finally through the global frame.

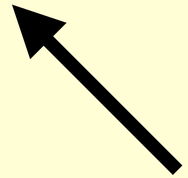
Let's now run the expression `(sqrt 3)`

Expressions Evaluated:

```
(define x 10)
x
(define (sqrt x)
  (expt x 0.5))
```

Global Environment Frame

Name	Value
define	DEFINE-BUILTIN
lambda	LAMBDA-BUILTIN
+	PLUS-BUILTIN
-	MINUS-BUILTIN
...	...
x	10
sqrt	(lambda (x) (expt x 0.5))



E1 Frame

Name	Value
x	3

E2 Frame

Name	Value

`(expt 3 0.5) => 1.73205`

Since expt is a built-in function, and since the function parameters are already evaluated, the interpreter evaluates the expression to give the final answer.

The non-global environments are no longer used when `(sqrt 3)` has been evaluated.

Expressions Evaluated:

```
(define x 10)
x
(define (sqrt x)
  (expt x 0.5))
(sqrt 3)
```

Global Environment Frame

Name	Value
define	DEFINE-BUILTIN
lambda	LAMBDA-BUILTIN
+	PLUS-BUILTIN
-	MINUS-BUILTIN
...	...
x	10
sqrt	(lambda (x) (expt x 0.5))

Let's do a `(set! x 50)`

Expressions Evaluated:

```
(define x 10)
x
(define (sqrt x)
  (expt x 0.5))
(sqrt 3)
```

Global Environment Frame

Name	Value
define	DEFINE-BUILTIN
lambda	LAMBDA-BUILTIN
+	PLUS-BUILTIN
-	MINUS-BUILTIN
...	...
x	10
sqrt	(lambda (x) (expt x 0.5))

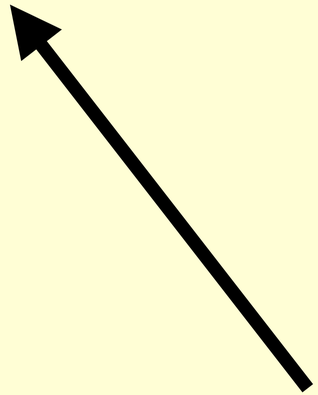
Let's do a `(set! x 50)`

Expressions Evaluated:

```
(define x 10)
x
(define (sqrt x)
  (expt x 0.5))
(sqrt 3)
```

Global Environment Frame

Name	Value
define	DEFINE-BUILTIN
lambda	LAMBDA-BUILTIN
+	PLUS-BUILTIN
-	MINUS-BUILTIN
...	...
x	10
sqrt	(lambda (x) (expt x 0.5))



E1 Frame

Name	Value

Let's create another environment since `(set! x 50)` is a function call.

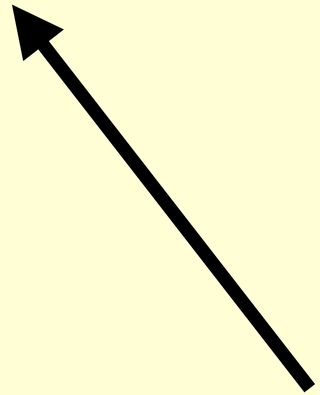
Let's do a `(set! x 50)`

Expressions Evaluated:

```
(define x 10)
x
(define (sqrt x)
  (expt x 0.5))
(sqrt 3)
```

Global Environment Frame

Name	Value
define	DEFINE-BUILTIN
lambda	LAMBDA-BUILTIN
+	PLUS-BUILTIN
-	MINUS-BUILTIN
...	...
x	10
sqrt	(lambda (x) (expt x 0.5))



E1 Frame

Name	Value

Then, perform a lookup of `set!` on first the new environment frame, and then the global environment frame. `set!` is a built-in function.

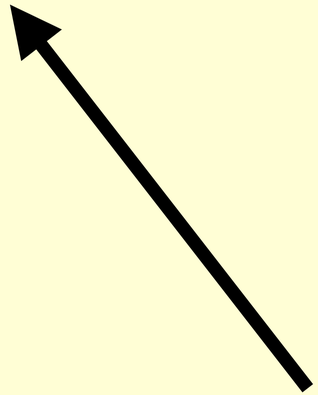
Let's do a `(set! x 50)`

Expressions Evaluated:

```
(define x 10)
x
(define (sqrt x)
  (expt x 0.5))
(sqrt 3)
```

Global Environment Frame

Name	Value
define	DEFINE-BUILTIN
lambda	LAMBDA-BUILTIN
+	PLUS-BUILTIN
-	MINUS-BUILTIN
...	...
x	~~10~~ 50
sqrt	(lambda (x) (expt x 0.5))



E1 Frame

Name	Value

`set!` replaces x with the value 50. First, the interpreter evaluates 50, when evaluates to 50. Next, it searches for x in the environment and then replaces the first occurrence of x with the new value.

What happens when there are multiple parallel function calls?
Let's evaluate `(+ (sqrt 2) (sqrt 3))`

Expressions Evaluated:

```
(define x 10)
x
(define (sqrt x)
  (expt x 0.5))
(sqrt 3)
(set! x 50)
```

Global Environment Frame

Name	Value
define	DEFINE-BUILTIN
lambda	LAMBDA-BUILTIN
+	PLUS-BUILTIN
-	MINUS-BUILTIN
...	...
x	50
sqrt	(lambda (x) (expt x 0.5))

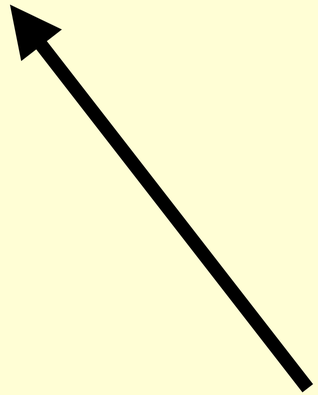
What happens when there are multiple parallel function calls?
Let's evaluate `(+ (sqrt 2) (sqrt 3))`

Expressions Evaluated:

```
(define x 10)
x
(define (sqrt x)
  (expt x 0.5))
(sqrt 3)
(set! x 50)
```

Global Environment Frame

Name	Value
define	DEFINE-BUILTIN
lambda	LAMBDA-BUILTIN
+	PLUS-BUILTIN
-	MINUS-BUILTIN
...	...
x	50
sqrt	(lambda (x) (expt x 0.5))



E1 Frame

Name	Value

1. Since we have a function call, create a new environment.

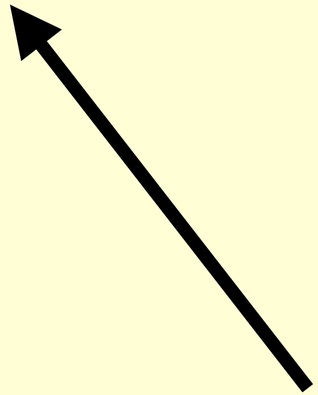
What happens when there are multiple parallel function calls?
Let's evaluate `(+ (sqrt 2) (sqrt 3))`

Expressions Evaluated:

```
(define x 10)
x
(define (sqrt x)
  (expt x 0.5))
(sqrt 3)
(set! x 50)
```

Global Environment Frame

Name	Value
define	DEFINE-BUILTIN
lambda	LAMBDA-BUILTIN
+	PLUS-BUILTIN
-	MINUS-BUILTIN
...	...
x	50
sqrt	(lambda (x) (expt x 0.5))



E1 Frame

Name	Value

1. Since we have a function call, create a new environment.
2. Perform lookup of +. + is a built-in.

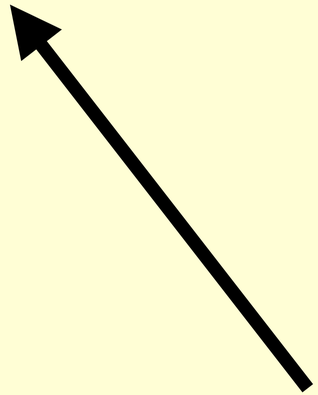
What happens when there are multiple parallel function calls?
Let's evaluate `(+ (sqrt 2) (sqrt 3))`

Expressions Evaluated:

```
(define x 10)
x
(define (sqrt x)
  (expt x 0.5))
(sqrt 3)
(set! x 50)
```

Global Environment Frame

Name	Value
define	DEFINE-BUILTIN
lambda	LAMBDA-BUILTIN
+	PLUS-BUILTIN
-	MINUS-BUILTIN
...	...
x	50
sqrt	(lambda (x) (expt x 0.5))



E1 Frame

Name	Value

1. Since we have a function call, create a new environment.
2. Perform lookup of +. + is a built-in.
3. Evaluate the arguments of +, which are (sqrt 2) and (sqrt 3).

What happens when there are multiple parallel function calls?
Let's evaluate `(+ (sqrt 2) (sqrt 3))`

Expressions Evaluated:

```
(define x 10)
x
(define (sqrt x)
  (expt x 0.5))
(sqrt 3)
(set! x 50)
```

Global Environment Frame

Name	Value
define	DEFINE-BUILTIN
lambda	LAMBDA-BUILTIN
+	PLUS-BUILTIN
-	MINUS-BUILTIN
...	...
x	50
sqrt	(lambda (x) (expt x 0.5))

E1 Frame

Name	Value

E2 Frame

Name	Value

E3 Frame

Name	Value

1. Since we have a function call, create a new environment.
2. Perform lookup of +. + is a built-in.
3. Evaluate the arguments of +, which are (sqrt 2) and (sqrt 3).
4. Since each of these are function calls, two new environments are created: E2 and E3, both of which are encompassed by E1, but in parallel.