

apply-generic

```

> (define (apply-generic op . args)
  (let ((type-tags (map type-tag args)))
    (let ((proc (get op type-tags)))
      (if proc
          (apply proc (map contents args))
          (error
           "No method for these types – APPLY-GENERIC"
           (list op type-tags))))))

> (define z3+4i (make-from-real-imag 3 4))

> z3+4i
(rectangular 3 . 4)

```

1

apply

apply is a built-in procedure taking a procedure and a list of arguments

```

(apply foo (list arg1 ... argn))
reduces to
(foo arg1 ... argn)

```

Example:

```

> (+ 1 2 3 4 5)
15
> (apply + '(1 2 3 4 5))
15

```

3

(apply-generic op . args)

Fixed number of arguments n

```

(lambda (<x1 ... <xn>) <body>)
(define (name <x1 ... <xn>) <body>)

```

One argument must be a list:

```

(lambda <x> <body>)

```

At least n arguments, $n \geq 1$, more than n arguments form list bound to x :

```

(lambda (<x1 ... <xn> . <x>) <body>)
(define (name <x1 ... <xn> . <x>) <body>)

```

2

Reduce (apply-generic-op real-part z3+4i)

```

(apply-generic-op 'real-part z3+4i)

(define (apply-generic op . args)
  (let ((type-tags (map type-tag '(rectangular 3 . 4))))
    (let ((proc (get 'real-part type-tags)))
      (if proc
          (apply proc (map contents '(rectangular 3 . 4)))
          (error
           "No method for these types – APPLY-GENERIC"
           (list 'real-part type-tags))))))

```

4

Reduce (apply-generic-op real-part z3+4i)

```
(let ((type-tags (map type-tag '((rectangular 3 . 4))))  
  (let ((proc (get 'real-part type-tags))  
        (if proc  
            (apply proc (map contents '((rectangular 3 . 4))))  
            (error  
              "No method for these types – APPLY-GENERIC"  
              (list 'real-part type-tags))))))  
  (let ((type-tags '(rectangular))  
        (let ((proc (get 'real-part type-tags))  
              (if proc  
                  (apply proc (map contents '((rectangular 3 . 4))))  
                  (error  
                    "No method for these types – APPLY-GENERIC"  
                    (list 'real-part type-tags))))))
```

5

Reduce (apply-generic-op real-part z3+4i)

```
(let ((type-tags '(rectangular))  
  (let ((proc (get 'real-part type-tags))  
        (if proc  
            (apply proc (map contents '((rectangular 3 . 4))))  
            (error  
              "No method for these types – APPLY-GENERIC"  
              (list 'real-part type-tags))))))  
  (let ((type-tags '(rectangular))  
        (let ((proc (get 'real-part '(rectangular)))  
              (if proc  
                  (apply proc (map contents '((rectangular 3 . 4))))  
                  (error  
                    "No method for these types – APPLY-GENERIC"  
                    (list 'real-part '(rectangular))))))
```

6

Reduce (apply-generic-op real-part z3+4i)

```
(let ((proc (get 'real-part '(rectangular))))  
  (if proc  
      (apply proc (map contents '((rectangular 3 . 4))))  
      (error  
        "No method for these types – APPLY-GENERIC"  
        (list 'real-part '(rectangular))))))  
  
(let ((proc {proc (z) (car z)}))  
  (if proc  
      (apply proc (map contents '((rectangular 3 . 4))))  
      (error  
        "No method for these types – APPLY-GENERIC"  
        (list 'real-part '(rectangular))))))
```

7

Reduce (apply-generic-op real-part z3+4i)

```
(let ((proc {proc (z) (car z)}))  
  (if proc  
      (apply proc (map contents '((rectangular 3 . 4))))  
      (error  
        "No method for these types – APPLY-GENERIC"  
        (list 'real-part '(rectangular))))))  
  
(let ((proc {proc (z) (car z)}))  
  (if {proc (z) (car z)}  
      (apply {proc (z) (car z)} (map contents '((rectangular 3 . 4))))  
      (error  
        "No method for these types – APPLY-GENERIC"  
        (list 'real-part '(rectangular))))))
```

8

Reduce (apply-generic-op real-part z3+4i)

```
(if {proc (z) (car z)}  
  (apply {proc (z) (car z)} (map contents '((rectangular 3 . 4))))  
  (error  
    "No method for these types – APPLY-GENERIC"  
    (list 'real-part '(rectangular))))  
  
(if {proc (z) (car z)}  
  (apply {proc (z) (car z)} (map contents '((rectangular 3 . 4))))  
  (error  
    "No method for these types – APPLY-GENERIC"  
    (list 'real-part '(rectangular))))
```

Reduce (apply-generic-op real-part z3+4i)

```
(apply {proc (z) (car z)} (map contents '((rectangular 3 . 4))))  
  
(apply {proc (z) (car z)} '(3 . 4))  
  
({proc (z) (car z)} '(3 . 4))  
  
(car '(3 . 4))  
  
3
```

9

10

Summary

Understand **apply-generic**:

- Understand new features:
 - Understand Scheme's notation for variable length arguments.
 - Understand the **apply** primitive.
- Understand action of **apply-generic** through use of substitution model, assuming behavior of **get**, **type-tag**, and **contents** is as per section 2.4