MASSACHVSETTS INSTITVTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.001—Structure and Interpretation of Computer Programs
Fall Semester, 1996

Lecture Notes, October 8 – Generic Operations

Some Strategies for Managing Data Complexity

Complex Numbers - Rectangular and Polar Coordinates

Complex Number Arithmetic

```
(define (add-complex z1 z2)
  (make-from-real-imag (+ (real-part z1) (real-part z2))
                       (+ (imag-part z1) (imag-part z2))))
(define (sub-complex z1 z2)
  (make-from-real-imag (- (real-part z1) (real-part z2))
                       (- (imag-part z1) (imag-part z2))))
(define (mul-complex z1 z2)
  (make-from-mag-ang (* (magnitude z1) (magnitude z2))
                     (+ (angle z1) (angle z2))))
(define (div-complex z1 z2)
  (make-from-mag-ang (/ (magnitude z1) (magnitude z2))
                     (- (angle z1) (angle z2))))
```

Alternative Implementations

```
;;;  Ben's representation – using rectangular coordinates
;;;  RepRect = Sch-Num X Sch-Num
(define (make-from-real-imag x y) (cons x y))
(define (real-part z) (car z))
(define (imag-part z) (cdr z))
(define (magnitude z) (sqrt (+ (square (real-part z))
                               (square (imag-part z)))))
(define (angle z) (atan (imag-part z) (real-part z)))
(define (make-from-mag-ang r a)
  (cons (* r (cos a)) (* r (sin a))))

;;;  Alyssa's representation – using polar coordinates
;;;  RepPolar = Sch-Num X Sch-Num
(define (make-from-mag-ang r a) (cons r a))
(define (magnitude z) (car z))
(define (angle z) (cdr z))
(define (real-part z) (* (magnitude z) (cos (angle z))))
(define (imag-part z) (* (magnitude z) (sin (angle z))))
(define make-from-real-imag x y)
  (cons (sqrt (+ (square x) (square y)))  (atan y x)))
```

Tagged Data Conventions

```
(define (attach-tag type-tag contents)
  (cons type-tag contents))

(define (type-tag datum)
  (if (pair? datum)
      (car datum)
      (error "Bad tagged datum - TYPE-TAG" datum)))

(define (contents datum)
  (if (pair? datum)
      (cdr datum)
      (error "Bad tagged datum - CONTENTS" datum)))
```

Complex Numbers with Tagged Data

```
;;  Complex Number Type Predicates
(define (rectangular? z) (eq? (type-tag z) 'rectangular))
(define (polar? z) (eq? (type-tag z) 'polar))

;;;  Ben's representation – Rectangular = 'rectangular X RepRect
(define (make-from-real-imag x y)
  (attach-tag 'rectangular (cons x y)))
(define (make-from-mag-ang r a)
  (attach-tag 'rectangular (cons (* r (cos a)) (* r (sin a)))))
;; Rectangular -> Sch-Num
(define (real-part-rectangular z) (car (contents z)))
(define (imag-part-rectangular z) (cdr (contents z)))
(define (magnitude-rectangular z)
  (sqrt (+ (square (real-part-rectangular z))
           (square (imag-part-rectangular z)))))
(define (angle-rectangular z)
  (atan (imag-part-rectangular z)
        (real-part-rectangular z)))

;;;  Alyssa's representation – Polar = 'polar X RepPolar
(define (make-from-mag-ang r a) (attach-tag 'polar (cons r a)))
(define (make-from-real-imag x y)
  (attach-tag 'pola (cons (sqrt (+ (square x) (square y)))
                          (atan y x))))
;; Polar -> Sch-Num
(define (magnitude-polar z) (car (contents z)))
(define (angle-polar z) (cdr (contents z)))
(define (real-part-polar z) (* (magnitude-polar z) (cos (angle-polar z))))
(define (imag-part-polar z) (* (magnitude-polar z) (sin (angle-polar z))))
```

Corresponding Generic Operators & Constructors

```
;; Complex = Rectangular U Polar
;; accessors: Complex -> Sch-Num
(define (real-part z)
  (cond ((rectangular? z) (real-part-rectangular z))
        ((polar? z) (real-part-polar z))
        (else (error "Unknown type - REAL-PART" z))))
(define (imag-part z)
  (cond ((rectangular? z) (imag-part-rectangular z))
        ((polar? z) (imag-part-polar z))
        (else (error "Unknown type - IMAG-PART" z))))
(define (magnitude z)
  (cond ((rectangular? z) (magnitude-rectangular z))
        ((polar? z) (magnitude-polar z))
        (else (error "Unknown-type - MAGNITUDE" z))))
(define (angle z)
  (cond ((rectangular? z) (angle-rectangular z))
        ((polar? z) (angle-polar z))
        (else (error "Unknown-type - ANGLE" z))))

;; make-from-real-imag: Sch-Num, Sch-Num -> Complex
(define (make-from-real-imag x y)
  (make-from-real-imag-rectangular x y))
;; make-from-mag-ang: Sch-Num, Sch-Num -> Complex
(define (make-from-mag-ang r a)
  (make-from-mag-ang-polar r a))
```

Problems with this Approach

Generic Interface - Table Driven

```
(put <op> <type> <procedure>)
(get <op> <type>)

(define (real-part z) (apply-generic 'real-part z))
(define (imag-part z) (apply-generic 'imag-part z))
(define (magnitude z) (apply-generic 'magnitude z))
(define (angle z)     (apply-generic 'angle z))

;; A simple version...
(define (simple-apply-generic op arg)
  (let ((type-tag (type-tag arg)))
    (let ((proc (get op type-tag)))
      (if proc
          (apply proc (list arg))
           (error "No method for types - APPLY-GENERIC"
                 (list op type-tag))))))

;; Version to support variable number of arguments:
(define (multiple-apply-generic op . args)
  (let ((type-tags (map type-tags args)))
    (let ((proc (get op type-tags)))
      (if proc
          (apply proc args)
          (error "No method for types - APPLY-GENERIC"
                (list op type-tags))))))

;; Convention: Generic system manages type tags.
(define (apply-generic op . args)
  (let ((type-tags (map type-tags args)))
    (let ((proc (get op type-tags)))
      (if proc
          (apply proc (map contents args))
          (error "No method for types - APPLY-GENERIC"
                (list op type-tags))))))
```

Table-Driven Implementation & Installation

```
;;;  Ben's (Rectangular) complex implementation...
(define (install-rectangular-package)
  ;; internal procedures on RepRect = Sch-Num X Sch-Num
  (define (real-part z) (car z))
  (define (imag-part z) (cdr z))
  (define (make-from-real-imag x y) (cons x y))
  (define (magnitude z)
    (sqrt (+ (square (real-part z)) (square (imag-part z)))))
  (define (angle z) (atan (imag-part z) (real-part z)))
  (define (make-from-mag-ang r a) (cons (* r (cos a)) (* r (sin a))))

  ;; interface to the rest of the system
  (define (tag x) (attach-tag 'rectangular x))
  (put 'real-part '(rectangular) real-part)
  (put 'imag-part '(rectangular) imag-part)
  (put 'magnitude '(rectangular) magnitude)
  (put 'angle     '(rectangular) angle)
  (put 'make-from-real-imag 'rectangular
    (lambda (x y) (tag (make-from-real-imag x y))))
  (put 'make-from-mag-ang 'rectangular
    (lambda (r a) (tag (make-from-mag-ang r a)))))

;;;  Alyssa's (Polar) complex representation...
(define (install-polar-package)
  ;; internal procedures
  (define (magnitude z) (car z))
  (define (angle z) (cdr z))
  (define (make-from-mag-ang r a) (cons r a))
  (define (real-part z) (* (magnitude z) (cos (angle z))))
  (define (imag-part z) (* (magnitude z) (sin (angle z))))
  (define (make-from-real-imag x y)
    (cons (sqrt (+ (square x) (square y)))  (atan y x)))

  ;; interface to the rest of the system
  (define (tag x) (attach-tag 'polar x))
  (put 'real-part '(polar) real-part)
  (put 'imag-part '(polar) imag-part)
  (put 'magnitude '(polar) magnitude)
  (put 'angle     '(polar) angle)
  (put 'make-from-real-imag 'polar
    (lambda (x y) (tag (make-from-real-imag x y))))
  (put 'make-from-mag-ang 'polar
    (lambda (x y) (tag (make-from-mag-ang r a)))))

(define (make-from-real-imag x y)
  ((get 'make-from-real-imag 'rectangular) x y))
(define (make-from-mag-ang r a)   ((get 'make-from-mag-ang 'polar) r a))
```