6.945 Spring 2009

Generic Operator Discovery

Laura Harris <lch> and Bryan Newbold <bnewbold>

Once generics are Registered...

```
; As a referesher, in case you forgot...
; these procedures and generic-operator-table are defined in ghelper.scm

(define add1 (make-generic-operator 1 #f 'add1))      ; note the symbols
(define sub1 (make-generic-operator 1 #f 'sub1))
(define double (make-generic-operator 1 #f 'double))
(define square (make-generic-operator 1 #f 'square))
(defhandler add1 (lambda (x) (+ x 1)) number?)
(defhandler sub1 (lambda (x) (- x 1)) number?)
(defhandler double (lambda (x) (* 2 x)) number?)
(defhandler square (lambda (x) (* x x)) number?)
```

... they can be Discovered...

```
; discover:named-opers-for searches *generic-operator-table* for
; the set of named (meaning defined with name symbols) operators
; which can be applied to the arguments

(discover:named-opers-for 4.5)

;Value 84: (double square add1 sub1 thingaling)
```

... and leveraged

```
; discover:satisfy-sequence takes a [predicate?] and [. args]

(discover:satisfy-sequence (lambda (x) (eq? x 9)) (/ 1 2))

;Value 8:  (((9) square double add1)
            ((9) square add1 inverse))
```

Assumptions

- Dispatch predicates are *fast*, operator procedures may be *slow*
- Operator “return value” type is indeterminant

Remarks

- This isn't a good way! Brute force, but...
- Unknown systems with existing operators
- Flexibility, reusability
- Lazily Distribute

Possible Applications

- File/media conversion and visualization
- Fix data corruption
  - CRC or hash as predicate
  - bitshifts and simple operations as operators
- Exploration of new systems, libraries
- Tool in the toolbox

Example Screenshot