Unstable Find: May Change Without Warning

Version 6.0

Ryan Culpepper <ryanc@racket-lang.org>

February 18, 2014

This library is *unstable*; compatibility will not be maintained. See *Unstable: May Change Without Warning* for more information.

```
(require unstable/find)
```
package: `macro-debugger`

```
(find pred
      x
     [#:stop-on-found? stop-on-found?
      #:stop stop
      #:get-children get-children])   → list?
  pred : (-> any/c any/c)
  x : any/c
  stop-on-found? : any/c = #f
  stop : (or/c #f (-> any/c any/c)) = #f
  get-children : (or/c #f (-> any/c (or/c #f list?))) = #f
```

Returns a list of all values satisfying *pred* contained in *x* (possibly including *x* itself).

If *stop-on-found?* is true, the children of values satisfying *pred* are not examined. If *stop* is a procedure, then the children of values for which *stop* returns true are not examined (but the values themselves are; *stop* is applied after *pred*). Only the current branch of the search is stopped, not the whole search.

The search recurs through pairs, vectors, boxes, and the accessible fields of structures. If *get-children* is a procedure, it can override the default notion of a value's children by returning a list (if it returns false, the default notion of children is used).

No cycle detection is done, so `find` on a cyclic graph may diverge. To do cycle checking yourself, use *stop* and a mutable table.

Examples:

```
> (find symbol? '((all work) and (no play)))
'(all work and no play)
> (find list? '#((all work) and (no play)) #:stop-on-found? #t)
'((all work) (no play))
> (find negative? 100
        #:stop-on-found? #t
        #:get-children (lambda (n) (list (- n 12))))
'(-8)
> (find symbol? (shared ([x (cons 'a x)]) x)
        #:stop (let ([table (make-hasheq)])
                 (lambda (x)
                   (begin0 (hash-ref table x #f)
                           (hash-set! table x #t)))))
'(a)
```

```
(find-first  pred
             x
            [#:stop stop
             #:get-children get-children
             #:default default])          → any/c
  pred : (-> any/c any/c)
  x : any/c
  stop : (or/c #f (-> any/c any/c)) = #f
  get-children : (or/c #f (-> any/c (or/c #f list?))) = #f
  default : any/c = (lambda () (error ....))
```

Like `find`, but only returns the first match. If no matches are found, *`default`* is applied as a thunk if it is a procedure or returned otherwise.

Examples:

```
> (find-first symbol? '((all work) and (no play)))
'all
> (find-first list? '#((all work) and (no play)))
'(all work)
> (find-first negative? 100
              #:get-children (lambda (n) (list (- n 12))))
-8
> (find-first symbol? (shared ([x (cons 'a x)]) x))
'a
```