Binomial heaps

Version 5.3.4

October 21, 2013

```
(require binomial)
```

Binomial Heaps

A *Binomial heap* is a data structure for maintaining a collection of elements, such that new elements can be added and the element of minimum value extracted efficiently. This implmentation is purely functional hence *immutable*. *Binomial heap* allow only numbers to be stored in them.

heap-lazy? is just a check if the given argument complies with the binomial heap structure adopted in this implementation. *Binomial heaps* have a array-based implementation. All values of the heap are stored in a vector which is pointed by `car` of a pair. The `cdr` is the count/size of the heap. This pair is embedded within another pair's `car`. The `cdr` of the outer pair stores the min value of the heap.

```
(bino-makeheap val) → heap-lazy?
  val : number?
```

Returns a newly allocated heap with only one element `val`.

Examples:

```
> (define h (bino-makeheap 1))

> h
'((#(1) . 1) . 1)
```

```
(bino-findmin h) → number?
  h : heap-lazy?
```

Returns a minimum value in the heap `h`.

Examples:

```
> (define h (bino-meld (bino-makeheap 1) (bino-makeheap 2)))

> (bino-findmin h)
1
```

```
(bino-insert h val) → heap-lazy?
  h : heap-lazy?
  val : number?
```

Returns a newly allocated heap which is a copy of `h` along with `val`.

Examples:

```
> (define h (bino-makeheap 1))

> (bino-insert h 2)
'((#(#f 1 2) . 2) . 1)
```

```
(bino-deletemin h) → heap?
  h : heap?
```

Returns a newly allocated heap with the min value of the given heap `h` removed.

Examples:

```
> (define h (bino-makeheap 1))

> (bino-deletemin h)
'((#() . 0) . #f)
```

```
(bino-meld h1 h2) → heap?
  h1 : heap-lazy?
  h2 : heap-lazy?
```

Returns a newly allocated heap by coupling `h1` and `h2`.

Examples:

```
> (define h (bino-meld (bino-makeheap 1) (bino-makeheap 2)))

> h
'((#(#f 1 2) . 2) . 1)
```

```
(bino-count h) → exact-nonnegative-integer?
  h : heap-lazy?
```

Returns the count of the elements in the heap *h*

Examples:

```
> (define h (bino-meld (bino-makeheap 1) (bino-makeheap 2)))

> (bino-count h)
2
```