

Contents

Comparisons and Ordering	1
Arithmetic	1
Boolean	2
Sequences	2
Shift And Rotate	3
Random Values	3
Debugging	3

Comparisons and Ordering

```
(==)  : {a} (Cmp a) => a -> a -> Bit
(!=)  : {a} (Cmp a) => a -> a -> Bit
(===) : {a,b} (Cmp b) => (a -> b) -> (a -> b) -> a -> Bit
(!==) : {a,b} (Cmp b) => (a -> b) -> (a -> b) -> a -> Bit
```

```
(<)   : {a} (Cmp a) => a -> a -> Bit
(>)   : {a} (Cmp a) => a -> a -> Bit
(<=)  : {a} (Cmp a) => a -> a -> Bit
(>=)  : {a} (Cmp a) => a -> a -> Bit
```

```
min   : {a} (Cmp a) => a -> a -> a
max   : {a} (Cmp a) => a -> a -> a
```

```
instance Cmp Bit
// No instance for functions.
instance (Cmp a, fin n) => Cmp [n] a
instance (Cmp a, Cmp b) => Cmp (a,b)
instance (Cmp a, Cmp b) => Cmp { x : a, y : b }
```

Arithmetic

```
(+)   : {a} (Arith a) => a -> a -> a
(-)   : {a} (Arith a) => a -> a -> a
(*)   : {a} (Arith a) => a -> a -> a
(/)   : {a} (Arith a) => a -> a -> a
(%)   : {a} (Arith a) => a -> a -> a
(^^^) : {a} (Arith a) => a -> a -> a
```

```
// No instance for `Bit`.
instance (fin n)           => Arith ( [n] Bit)
instance (Arith a)        => Arith ( [n] a)
instance (Arith b)        => Arith (a -> b)
```

```
instance (Arith a, Arith b) => Arith (a,b)
instance (Arith a, Arith b) => Arith { x : a, y : b }
```

Note that because there is no instances for `Arith Bit` the top two instances do not actually overlap.

Boolean

```
False : Bit
True  : Bit
```

```
zero : a
(&&) : a -> a -> a
(||) : a -> a -> a
(^)  : a -> a -> a
(~)  : a -> a
```

Sequences

```
length : {n,a,m} (m >= width n) => [n]a -> [m]

join    : {parts,ench,a} (fin each) => [parts][each]a -> [parts * each]a
split   : {parts,each,a} (fin each) => [parts * each]a -> [parts][each]a

(#)     : {front,back,a} (fin front) => [front]a -> [back]a -> [front + back]a
splitAt : {front,back,a} (fin front) => [from + back] a -> ([front] a, [back] a)

reverse : {n,a} (fin n) => [n]a -> [n]a
transpose : {n,m,a} [n] [m]a -> [m] [n]a

(@)      : {n a m} [n]a -> [m] -> a
(@@)     : {n a m i} [n]a -> [m] [i] -> [m]a
(!)      : {n a m} (fin n) => [n]a -> [m] -> a
(!!)     : {n a m i} (fin n) => [n]a -> [m] [i] -> [m]a
```

// Abbreviations

```
groupBy n = split`{each = n}
tail n    = splitAt`{front = 1}.1
take n    = splitAt`{front = n}.0
drop n    = splitAt`{front = n}.1
```

/* Also, `length`` is not really needed:

```
length : {n,a,m} (m >= width n) => [n]a -> [m]
```

```
length _ = `n
*/
```

Shift And Rotate

New types:

```
(<<)  : {n,a,m} (fin n) => [n]a -> [m] -> [n]a
(>>)  : {n,a,m} (fin n) => [n]a -> [m] -> [n]a
(<<<) : {n,a,m} (fin n) => [n]a -> [m] -> [n]a
(>>>) : {n,a,m} (fin n) => [n]a -> [m] -> [n]a
```

Random Values

```
random : {a} => [256] -> a
```

Debugging

```
undefined : {a} a
error      : {n a} [n] [8] -> a
```