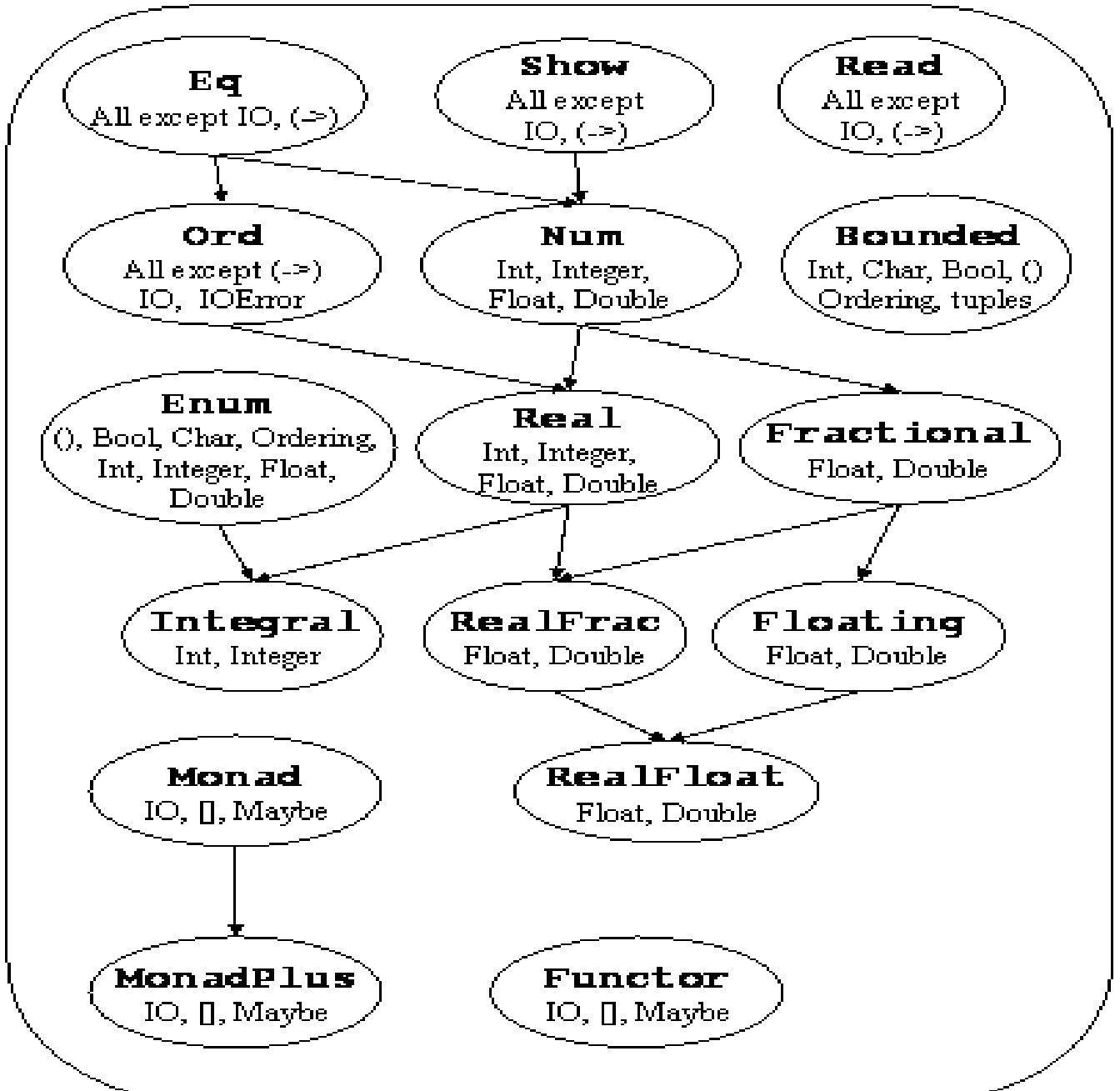
Predefined Types and Classes


Eq
All except IO, (->)
Show
All except IO, (->)
Read
All except IO, (->)
Ord
All except (->) IO, IOError
Num
Int, Integer, Float, Double
Bounded
Int, Char, Bool, () Ordering, tuples
Enum
(), Bool, Char, Ordering, Int, Integer, Float, Double
Real
Int, Integer, Float, Double
Fractional
Float, Double
Integral
Int, Integer
RealFrac
Float, Double
Floating
Float, Double
Monad
IO, [], Maybe
RealFloat
Float, Double
MonadPlus
IO, [], Maybe
Functor
IO, [], Maybe

- The Haskell Prelude contains predefined classes, types, and functions that are implicitly imported into every Haskell program.
- Default class method declarations are provided for many of the methods in standard classes.

The Eq Class

- The Eq class provides equality (==) and inequality (/=) methods.
- All basic datatypes except for functions and IO are instances of this class. Instances of Eq can be derived for any user-defined datatype whose constituents are also instances of Eq.
- This declaration gives default method declarations for both /= and ==, each being defined in terms of the other.
- If an instance declaration for Eq defines neither == nor /=, then both will loop.
- If one is defined, the default method for the other will make use of the one that is defined.
- If both are defined, neither default method is used.

- class Eq a where
  (==), (/=) :: a -> a -> Bool

  x /= y = not (x == y)
  x == y = not (x /= y)

The Ord Class

- The Ord class is used for totally ordered datatypes.
- All basic datatypes except for functions, IO, and IOError, are instances of this class.
- Instances of Ord can be derived for any user-defined datatype whose constituent types are in Ord.
- The declared order of the constructors in the data declaration determines the ordering in derived Ord instances.
- The Ordering datatype allows a single comparison to determine the precise ordering of two objects.
- The default declarations allow a user to create an Ord instance either with a type-specific compare function or with type-specific == and <= functions.

- class (Eq a) => Ord a where
  compare :: a -> a -> Ordering
  (<), (<=), (>=), (>) :: a -> a -> Bool
  max, min :: a -> a -> a

  compare x y | x == y = EQ
  | x <= y = LT
  | otherwise = GT

  x <= y = compare x y /= GT
  x < y = compare x y == LT
  x >= y = compare x y /= LT
  x > y = compare x y == GT

  -- Note that (min x y, max x y) = (x,y) or (y,x)
  max x y | x <= y = y
  | otherwise = x
  min x y | x <= y = x
  | otherwise = y

The Read and Show Classes

- The Read and Show classes are used to convert values to or from strings.

The Enum Class