Generalizing into Workflow

```
(|>) :: Logged Double -> (Double -> Logged Double) -> Logged Double
wrap :: Double -> Logged Double
```

```
(|>) :: [a] -> (a -> [b]) -> [b]
wrap :: a -> [a]
```

```
(|>) :: Maybe a -> (a -> Maybe b) -> Maybe b
wrap :: a -> Maybe a
```

```
(|>) :: Logged Double -> (Double -> Logged Double) -> Logged Double
wrap :: Double -> Logged Double
```

```
(|>) :: [a] -> (a -> [b]) -> [b]
wrap :: a -> [a]
```

```
(|>) :: Maybe a -> (a -> Maybe b) -> Maybe b
wrap :: a -> Maybe a
```

```
class Workflow w where
  (|>) :: ??
  wrap :: ??
```

```
(|>) :: Logged Double -> (Double -> Logged Double) -> Logged Double
wrap :: Double -> Logged Double
```

w -> Logged

```
(|>) :: [a] -> (a -> [b]) -> [b]
wrap :: a -> [a]
```

w -> []

```
(|>) :: Maybe a -> (a -> Maybe b) -> Maybe b
wrap :: a -> Maybe a
```

w -> Maybe

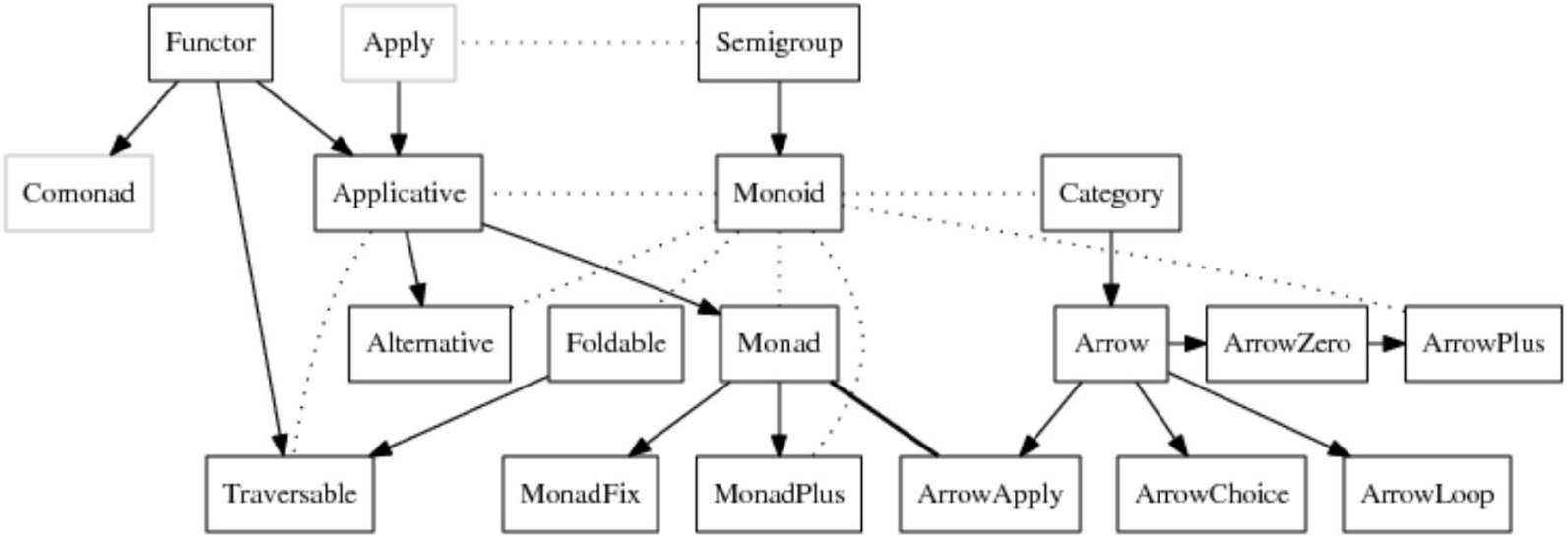
```
class Workflow w where
  (|>) :: w a -> (a -> w b) -> w b
  wrap :: a -> w a
```

Workflow -> Monad

```haskell
class Workflow w where
  (|>) :: w a -> (a -> w b) -> w b
  wrap :: a -> w a
```

```haskell
class Workflow w where
  (|>) :: w a -> (a -> w b) -> w b
  wrap :: a -> w a
```

```haskell
class Monad m where
  (>>=) :: m a -> (a -> m b) -> m b
  return :: a -> m a
```



- Solid arrows point from the general to the specific; that is, if there is an arrow from `Foo` to `Bar` it means that every `Bar` is (or should be, or can be made into) a `Foo`.
- Dotted lines indicate some other sort of relationship.
- `Monad` and `ArrowApply` are equivalent.
- `Apply` and `Comonad` are greyed out since they are not actually (yet?) in the standard Haskell libraries `*`.