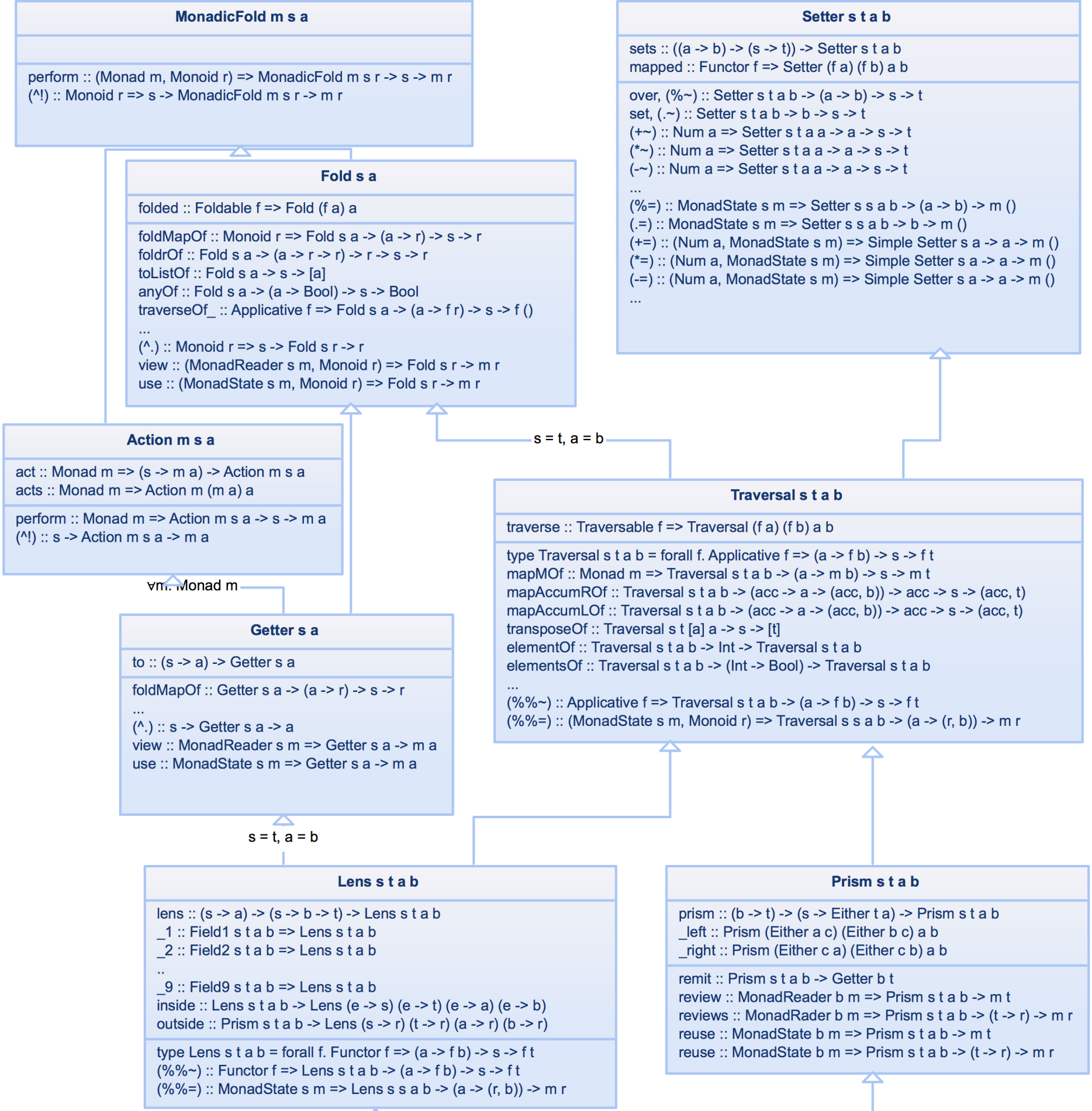
MonadicFold m s a
perform :: (Monad m, Monoid r) => MonadicFold m s r -> s -> m r
(^!) :: Monoid r => s -> MonadicFold m s r -> m r
Setter s t a b
sets :: ((a -> b) -> (s -> t)) -> Setter s t a b
mapped :: Functor f => Setter (f a) (f b) a b
over, (%~) :: Setter s t a b -> (a -> b) -> s -> t
set, (.~) :: Setter s t a b -> b -> s -> t
(+~) :: Num a => Setter s t a a -> a -> s -> t
(*~) :: Num a => Setter s t a a -> a -> s -> t
(-~) :: Num a => Setter s t a a -> a -> s -> t
...
(%=) :: MonadState s m => Setter s s a b -> (a -> b) -> m ()
(.=) :: MonadState s m => Setter s s a b -> b -> m ()
(+=) :: (Num a, MonadState s m) => Simple Setter s a -> a -> m ()
(*=) :: (Num a, MonadState s m) => Simple Setter s a -> a -> m ()
(-=) :: (Num a, MonadState s m) => Simple Setter s a -> a -> m ()
...
Fold s a
folded :: Foldable f => Fold (f a) a
foldMapOf :: Monoid r => Fold s a -> (a -> r) -> s -> r
foldrOf :: Fold s a -> (a -> r -> r) -> r -> s -> r
toListOf :: Fold s a -> s -> [a]
anyOf :: Fold s a -> (a -> Bool) -> s -> Bool
traverseOf_ :: Applicative f => Fold s a -> (a -> f r) -> s -> f ()
...
(^.) :: Monoid r => s -> Fold s r -> r
view :: (MonadReader s m, Monoid r) => Fold s r -> m r
use :: (MonadState s m, Monoid r) => Fold s r -> m r
s = t, a = b
Action m s a
act :: Monad m => (s -> m a) -> Action m s a
acts :: Monad m => Action m (m a) a
perform :: Monad m => Action m s a -> s -> m a
(^!) :: s -> Action m s a -> m a
∀m. Monad m
Traversal s t a b
traverse :: Traversable f => Traversal (f a) (f b) a b
type Traversal s t a b = forall f. Applicative f => (a -> f b) -> s -> f t
mapMOf :: Monad m => Traversal s t a b -> (a -> m b) -> s -> m t
mapAccumROf :: Traversal s t a b -> (acc -> a -> (acc, b)) -> acc -> s -> (acc, t)
mapAccumLOf :: Traversal s t a b -> (acc -> a -> (acc, b)) -> acc -> s -> (acc, t)
transposeOf :: Traversal s t [a] a -> s -> [t]
elementOf :: Traversal s t a b -> Int -> Traversal s t a b
elementsOf :: Traversal s t a b -> (Int -> Bool) -> Traversal s t a b
...
(%%~) :: Applicative f => Traversal s t a b -> (a -> f b) -> s -> f t
(%%=) :: (MonadState s m, Monoid r) => Traversal s s a b -> (a -> (r, b)) -> m r
Getter s a
to :: (s -> a) -> Getter s a
foldMapOf :: Getter s a -> (a -> r) -> s -> r
...
(^.) :: s -> Getter s a -> a
view :: MonadReader s m => Getter s a -> m a
use :: MonadState s m => Getter s a -> m a
s = t, a = b
Lens s t a b
lens :: (s -> a) -> (s -> b -> t) -> Lens s t a b
_1 :: Field1 s t a b => Lens s t a b
_2 :: Field2 s t a b => Lens s t a b
..
_9 :: Field9 s t a b => Lens s t a b
inside :: Lens s t a b -> Lens (e -> s) (e -> t) (e -> a) (e -> b)
outside :: Prism s t a b -> Lens (s -> r) (t -> r) (a -> r) (b -> r)
type Lens s t a b = forall f. Functor f => (a -> f b) -> s -> f t
(%%~) :: Functor f => Lens s t a b -> (a -> f b) -> s -> f t
(%%=) :: MonadState s m => Lens s s a b -> (a -> (r, b)) -> m r
Prism s t a b
prism :: (b -> t) -> (s -> Either t a) -> Prism s t a b
_left :: Prism (Either a c) (Either b c) a b
_right :: Prism (Either c a) (Either c b) a b
remit :: Prism s t a b -> Getter b t
review :: MonadReader b m => Prism s t a b -> m t
reviews :: MonadRader b m => Prism s t a b -> (t -> r) -> m r
reuse :: MonadState b m => Prism s t a b -> m t
reuse :: MonadState b m => Prism s t a b -> (t -> r) -> m r
Iso s t a b
iso :: (s -> a) -> (b -> t) -> Iso s t a b
from :: Iso s t a b -> Iso a b s t
wrapping :: Wrapped s s a a => (s -> a) -> Iso s s a a
enum :: Enum a => Simple Iso Int a
simple :: Simple Iso a a
mapping :: Functor f => Iso s t a b -> Iso (f s) (f t) (f a) (f b)
curried :: Iso ((a,b) -> c) ((d,e) -> f) (a -> b -> c) (d -> e -> f)
uncurried :: curried :: Iso (a -> b -> c) (d -> e -> f) ((a,b) -> c) ((d,e) -> f)
au :: Iso s t a b -> ((s -> a) -> e -> b) -> e -> t
auf :: Iso s t a b -> ((r -> a) -> e -> b) -> (b -> s) -> e -> t
under :: Iso s t a b -> (t -> s) -> b -> a
Equality s t a b
id :: Equality a b a b
mapEq :: Equality s t a b -> f s -> f a
fromEq :: Equality s t a b -> Equality b a t s
substEq :: Equality s t a b -> ((s ~ a, t ~ b) => r) -> r