**Haskell Platform + Semigroups + Semigroupoids + Either: Most Frequently used types typeclasses**

| | *Class:* | *SemiGroup* Data.Semigroup | *Monoids* Data.Monoid | *Functor* Data.Functor | *Alt* Data.Functor.Alt | *Plus* Data.Functor.Plus | *Apply* Data.Functor.Apply | *Applicative* Control.Applicative | *Alternative* Control.Applicative | *Bind* Data.Functor.Bind | *Monad* Control.Monad | *Monad Plus* Control.Monad |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| "Base" | **Type** | **<>** | **<>, mempty** | **fmap** | **<!>** | **zero** | **<.>** | **<*>, pure** | **<\|>,empty** | **>>-,join** | **>>=, return** | **mplus,mzero** |
| | **Ordering** D.O | x | x | | | | | | | | | |
| | **()** | x | x | | | | | | | | | |
| Bool | **All** D.MONOID | x | x | | | | | | | | | |
| Bool | **Any** D.MONOID | x | x | | | | | | | | | |
| Word8 | **ByteString** D.BS,D.BS.L | x | x | | | | | | | | | |
| Char | **Text** D.T,D.T.L | x | x | | | | | | | | | |
| Num | **Sum a** D.MONOID | Num a | Num a | | | | | | | | | |
| | **Product a** D.MONOID | Num a | Num a | | | | | | | | | |
| Ord | **Min a** D.SEMI | Ord a | Ord a, Bounded a | | | | | | | | | |
| | **Max a** D.SEMI | Ord a | Ord a, Bounded a | | | | | | | | | |
| | **Dual a** D.MONOID | Monoid a | Monoid a | | | | | | | | | |
| [] | **[a]** D.LIST | x | x | | | | | | | | | |
| | **[]** D.LIST | | | x | x | x | x | x | x | x | x | x |
| | **ZipList** C.APP | | | x | | | x | x | * | | | |
| | **Seq a** D.SEQ | x | x | | | | | | | | | |
| | **Seq** D.SEQ | | | x | x | x | x | x | x | x | x | x |
| | **NonEmpty a** D.L.NE | x | | | | | | | | | | |
| | **NonEmpty** D.L.NE | | | x | x | | x | x | | x | x | |
| | **DList a** D.DLIST | | DList a | x | | | | x | x | | x | x |
| | **Set a** D.SET | Ord a | Ord a | | | | | | | | | |
| Int | **IntSet** D.ISET | x | x | | | | | | | | | |
| | **HashSet a** D.HSET | Hashable a, Eq a | Hashable a, Eq a | | | | | | | | | |
| | **Map k a** D.MAP | Ord k | Ord k | | | | | | | | | |
| | **Map k** D.MAP | | | x | Ord k | Ord k | Ord k | | | Ord k | | |
| Int | **IntMap a** D.IMAP | x | x | | | | | | | | | |
| | **IntMap** D.IMAP | | | x | x | x | x | | | x | | |
| | **HashMap k a** D.HMAP | Hashable k, Eq k | Hashable k, Eq k | | | | | | | | | |
| | **HashMap k** D.HMAP | | | x | | | | | | | | |
| | **Tree** D.TREE | | | x | | | | x | | x | x | |
| Maybe | **Maybe** D.M | | | x | x | x | x | x | x | x | x | x |
| | **Maybe a** D.M | Semigroup a | Monoid a | | | | | | | | | |
| Maybe | **Option** D.SEMI | | | x | x | x | x | x | x | x | x | x |
| | **Option a** D.SEMI | Semigroup a | Semigroup a | | | | | | | | | |
| Maybe | **First a** D.M | | x | | | | | | | | | |
| | **Last a** D.M | | x | | | | | | | | | |
| | **First a** D.SEMI | x | | | | | | | | | | |
| | **Last a** D.SEMI | x | | | | | | | | | | |
| | **Either a** D.EITH | | | x | x | | x | x | | x | x | |
| | **Either a b** D.EITH | x | | | | | | | | | | |
| | **Identity** D.F.I | | | x | | | x | x | | x | x | |
| | **IdentityT m** C.M.T.I | | | Functor m | Alt m | Plus m | Apply m | Applicative m | Alternative m | Bind m | Monad m | MonadPlus m |
| Maybe | **MaybeT m** C.M.T.M | | | Functor m | Bind m, Monad m | Bind m, Monad m | Bind m, Monad m | Functor m, Monad m | Functor m, Monad m | Bind m, Monad m | Monad m | Monad m |
| [] | **ListT m** C.M.T.L | | | Functor m | Apply m | Apply m, Applicative m | Apply m | Applicative m | Applicative m | Bind m, Monad m | Monad m | Monad m |
| | **ReaderT e m** C.M.T.R | | | Functor m | Alt m | Plus m | Apply m | Applicative m | Alternative m | Bind m | Monad m | MonadPlus m |
| | **WriterT w m** C.M.T.W | | | Functor m | Alt m | Plus m | Apply m, Semigroup w | Monoid w, Applicative m | Monoid w, Alternative m | Bind m, Semigroup w | Monoid w, Monad m | Monoid w, MonadPlus m |
| | **StateT s m** C.M.T.S | | | Functor m | Alt m | Plus m | Bind m | Functor m, Monad m | Functor m, MonadPlus m | Bind m | Monad m | MonadPlus m |
| Either e | **ErrorT e m** C.M.T.ERR | | | Functor m | Bind m, Monad m | Bind m, Monad m, Error e | Bind m, Monad m | Functor m, Monad m | Functor m, Monad m, Error e | Bind m, Monad m | Monad m, Error e | Monad m, Error e |
| | **RWST r w s m** C.M.T.RWS | | | Functor m | Alt m | Plus m | Bind m, Semigroup w | Monoid w, Functor m, Monad m | Monoid w, Functor m, MonadPlus m | Bind m, Semigroup w | Monoid w, Monad m | Monoid w, MonadPlus m |
| Either | **EitherT e m** C.M.T.E | | | Monad m | Monad m, Semigroup e | | Monad m | Monad m, Monoid e | Monad m, Monoid e | Monad m | Monad m | Monad m, Monoid e |
| Either | **EitherT e m a** C.M.T.E | Semigroup m | | | | | | | | | | |
| | **Parser** D.ATTO | | | x | | | | x | x | | x | x |
| | **ParsecT s u m** T.PARSEC | | | x | | | | x | x | | x | x |
| | **WrappedMonoid m** D.SEMI | Monoid m | Monoid m | | | | | | | | | |
| | **WrappedApplicative f** D.F.A | | | | Alternative f | Alternative f | Applicative f | | | | | |
| | **WrappedMonad m** C.APP | | | Monad m | MonadPlus m | MonadPlus m | Monad m | Monad m | MonadPlus m | Monad m | | |
| | **WrappedArrow a b** C.APP | | | Arrow a | ArrowPlus a | ArrowPlus a | Arrow a | Arrow a | ArrowZero a, ArrowPlus a | | | |
| | **ArrowMonad a** C.ARR | | | Arrow a | | | | Arrow a | ArrowPlus a | | ArrowApply a | ArrowApply a, ArrowPlus a |
| | **IO** S.IO | | | x | x | x | x | x | | x | x | |
| | **ST s** C.M.ST | | | x | | | | x | | | x | |
| | **STM** | | | x | | | | x | x | | x | x |
| | **ReadP** | | | x | | | | x | x | | x | x |
| | **ReadPrec** | | | x | | | | x | x | | x | x |
| Tuples | **(a, b)** | Semigroup a, Semigroup b | Monoid a, Monoid b | | | | | | | | | |
| | **(a, b, c)** | Semigroup a, Semigroup b, Semigroup c | Monoid a, Monoid b, Monoid c | | | | | | | | | |
| | **(,) a** | | | x | | | Semigroup a | Monoid a | | Semigroup m | | |
| | **a → b** | Semigroup b | Monoid b | | | | | | | | | |
| | **Endo a** D.MONOID | x | x | | | | | | | | | |
| | **(→) a** | | | x | | | x | x | | x | x | |
| | **Const a b** C.APP | Semigroup a | | x | | | | | | | | |
| | **Const m** C.APP | | | x | | | Semigroup m | Monoid m | | | | |
| | **Static f a** D.SS | | | Functor f | Alt f | Plus f | Apply f | Applicative f | | | | |
| | **Compose f g** D.SS | | | Functor f, Functor g | | | | Aplicative f, Applicative g | Alternative f, Alternative g | | | |
| | **Product f g** D.F.C | | | Functor f, Functor g | | | Apply f, Apply g | Applicative f, Applicative g | Applicative f, Applicative g | Bind f, Bind g | | |

**Note 1:** **Typeclasses in Haskell imply laws, not (necessarily) semantics**

**Note 2:** **While a uniform semantic / behaviour would be nice to have, most 'class laws' were found insufficient to provide this, and yet no additional laws were specified (hysterical raisins)**

| | | |
|---|---|---|
| C.APP | Control.Applicative | base |
| C.ARR | Control.Arrow | base |
| C.M.ST | Control.Monad.ST | base |
| C.M.T.E | Control.Monad.Trans.Either | transformers |
| C.M.T.ERR | Control.Monad.Trans.Error | transformers |
| C.M.T.I | Control.Monad.Trans.Identity | transformers |
| C.M.T.L | Control.Monad.Trans.List | transformers |
| C.M.T.M | Control.Monad.Trans.Maybe | transformers |
| C.M.T.R | Control.Monad.Trans.Reader | transformers |
| C.M.T.RWS | Control.Monad.Trans.RWST | transformers |
| C.M.T.S | Control.Monad.Trans.State | transformers |
| C.M.T.W | Control.Monad.Trans.Writer | transformers |
| D.ATTO | Data.Attoparsec | attoparsec |

| | | |
|---|---|---|
| D.BS | Data.ByteString | bytestring |
| D.BS.L | Data.ByteString.Lazy | bytestring |
| D.DLIST | Data.Dlist | dlist |
| D.EITH | Data.Either, Prelude | base |
| D.F.A | Data.Functor.Apply | semigroupoids |
| D.F.B | Data.Functor.Bind | semigroupoids |
| D.F.C | Data.Functor.Compose | transformers |
| D.F.I | Data.Functor.Identity | transformers |
| D.HMAP | Data.HashMap | unordered-containers |
| D.HSET | Data.HashSet | unordered-containers |
| D.IMAP | Data.IntMap | containers |
| D.ISET | Data.InsSet | containers |
| D.LIST | Data.List, Prelude | base |
| D.L.NE | Data.List.NonEmpty | semigroups |

| | | |
|---|---|---|
| D.M | Data.Maybe, Prelude | base |
| D.MAP | Data.Map | containers |
| D.MONOID | Data.Monoid, Prelude | base |
| D.ORD | Data.Ord, Prelude | base |
| D.SEMI | Data.Semigroup | semigroups |
| D.SEQ | Data.Sequence | containers |
| D.SET | Data.Set | containers |
| D.SS | Data.Semigroupoid.* | semigroupoids |
| D.T | Data.Text | text |
| D.T.L | Data.Text.Lazy | text |
| D.TREE | Data.Tree | containers |
| S.IO | System.IO, Prelude | base |
| T.PARSEC | Text.Parsec.Prim | parsec |

| | Binary operation semantic / Mempty 'meaning' / Additional Laws | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *Class:* | *SemiGroup* Data.Semigroup | *Monoids* Data.Monoid | | *Alt* Data.Functor.Alt | *Plus* Data.Functor.Plus | *Apply* Data.Functor.Apply | *Applicative* Control.Applicative | *Alternative* Control.Applicative | *Bind* Data.Functor.Bind | *Monad* Control.Monad | *Monad Plus* Control.Monad |
| **"Base"** | **Type** | **<>** | **<>** | **mempty** | **<!>** | **zero** | **<.>** | **<*>, pure** | **<\|>,empty** | **>>-,join** | **>>=, return** | **mplus,mzero** |
| | **Ordering** | Choice | Choice | EQ | | | | | | | | |
| | **()** | None | None | () | | | | | | | | |
| Bool | **All** | Combine | Combine | True | | | | | | | | |
| Bool | **Any** | Combine | Combine | False | | | | | | | | |
| Word8 | **ByteString** | Combine | Combine | Empty | | | | | | | | |
| Char | **Text** | Combine | Combine | Empty | | | | | | | | |
| Num | **Sum a** | Combine | Combine | 0 | | | | | | | | |
| | **Product a** | Combine | Combine | 1 | | | | | | | | |
| Ord | **Min a** | Choice | Choice | maxBound | | | | | | | | |
| | **Max a** | Choice | Choice | minBound | | | | | | | | |
| | **Dual a** | ~ a | ~ a | ~ a | | | | | | | | |
| | **[a]** | Both | Both | Empty | | | | | | | | |
| [] | **[]**<br>**ZipList** | | | | Both | Empty | <*><br><*> | Both<br>Both | x<br>* | x | x | Left Dist. |
| | **Seq a**<br>**Seq** | Both | Both | Empty | Both | Empty | ap | x | x | x | x | x |
| | **NonEmpty a**<br>**NonEmpty** | Both | | | Both | | ap | x | | x | x | |
| | **DList a** | | Both | Empty | | | | Both | Both | | x | x |
| | **Set a** | Both | Both | Empty | | | | | | | | |
| Int | **IntSet** | Both | Both | Empty | | | | | | | | |
| | **HashSet a** | Both | Both | Empty | | | | | | | | |
| | **Map k a**<br>**Map k** | Both | Both | Empty | Both | Empty | (…) | | | Ord k | | |
| Int | **IntMap a**<br>**IntMap** | Both | Both | Empty | Both | Empty | (…) | | | x | | |
| | **HashMap k a**<br>**HashMap k** | Both | Both | Empty | | | | | | | | |
| | **Tree** | | | | | | | x | | x | x | |
| Maybe | **Maybe**<br>**Maybe a** | Combine | Combine | Empty | Choice | Empty | apDefault | x | Choice | x | x | Left Catch |
| Maybe | **Option**<br>**Option a** | Combine | Combine | Empty | Choice | Empty | <*> | x | Choice | x | x | x |
| Maybe | **First a**<br>**Last a** | | Choice<br>Choice | Empty<br>Empty | | | | | | | | |
| | **First a**<br>**Last a** | Choice<br>Choice | | | | | | | | | | |
| | **Either a**<br>**Either a b** | Choice | | ~ Empty | Choice | | (…) | x | | x | x | |
| | **Identity** | | | | | | <*> | x | | x | x | |
| | **IdentityT m** | | | | Combine | ~ m | <.> | Applicative m | Alternative m | Bind m | Monad m | MonadPlus m |
| Maybe | **MaybeT m** | | | | Choice | Empty | apDefault | Functor m, Monad m | Functor m, Monad m | Bind m, Monad m | Monad m | Monad m |
| [] | **ListT m** | | | | Combine | Empty | (…) | Applicative m | Applicative m | Bind m, Monad m | Monad m | Monad m |
| | **ReaderT e m** | | | | Combine | ~ m | (…) | Applicative m | Alternative m | Bind m | Monad m | MonadPlus m |
| | **WriterT w m** | | | | Combine | ~m | (…) | Monoid w, Applicative m | Monoid w, Alternative m | Bind m, Semigroup w | Monoid w, Monad m | Monoid w, MonadPlus m |
| | **StateT s m** | | | | Combine | ~ m | apDefault | Functor m, Monad m | Functor m, MonadPlus m | Bind m | Monad m | MonadPlus m |
| Either e | **ErrorT e m** | | | | Choice | ~ Empty | apDefault | Functor m, Monad m | Functor m, Monad m, Error e | Bind m, Monad m | Monad m, Error e | Monad m, Error e |
| | **RWST r w s m** | | | | Combine | ~ m | apDefault | Monoid w, Functor m, Monad m | Monoid w, Functor m, MonadPlus m | Bind m, Semigroup w | Monoid w, Monad m | Monoid w, MonadPlus m |
| Either | **EitherT e m** | | | | Both? | | (…) | Monad m, Monoid e | Both | Monad m | Monad m | Monad m, Monoid e |
| Either | **EitherT e m a** | Choice | | | | | | | | | | |
| | **Parser** DATTO | | | x | | | | x | x | | x | x |
| | **ParsecT s u m** TPARSEC | | | x | | | | x | x | | x | x |
| | **WrappedMonoid m** | ~ m | ~ m | ~ m | | | | | | | | |
| | **WrappedApplicative f** | | | | ~ f | ~ f | ~ f | | | | | |
| | **WrappedMonad m** | | | | ~ f | ~ m | ~ m | ~ m | ~ m | ~ m | | |
| | **WrappedArrow a b** | | | | ~ a | ~ a | ~ a | ~ a | ArrowZero a, ArrowPlus a | | | |
| | **ArrowMonad a** | | | | | | | ~ a | ~ a | | ArrowApply a | ArrowApply a, ArrowPlus a |
| | **IO** | | | | Choice | error | <*> | x | | x | x | |
| | **ST s** | | | | | | | x | | | x | |
| | **STM** | | | | | | | x | x | | x | Left Catch |
| | **ReadP** | | | | | | | x | x | | x | x |
| | **ReadPrec** | | | | | | | x | x | | x | x |
| Tuples | **(a, b)** | ~a, ~b | ~a, ~b | ~a, ~b | | | | | | | | |
| | **(a, b, c)** | ~a, ~b, ~c | ~ a, ~ b, ~c | ~ a, ~ b, ~c | | | | | | | | |
| | **(,) a** | | | | | | (..) | Monoid a | | Semigroup m | | |
| | **a → b** | ~ b | ~ b | ~ b | | | | | | | | |
| | **Endo a** | Neither | Neither | Neither | | | | | | | | |
| | **(→) a** | | | | | | <*> | x | | x | x | |
| | **Const a b**<br>**Const m** | ~ a | | | | | <><br>Monoid m | | | | | |
| | **Static f a** | | | | ~ f | Plus f | (..) | Applicative f | | | | |
| | **Compose f g** | | | | | | | Aplicative f, Applicative g | Alternative f, Alternative g | | | |
| | **Product f g (D.F.P)** | | | | | | (..) | Applicative f, Applicative g | Applicative f, Applicative g | Bind f, Bind g | | |
| | **Cokleisli w a** | | | | | | (..) | x | | | x | |

| | |
|---|---|
| **Choice** | Binary operation chooses one of the values |
| **Combine** | Binary operation combines both values |
| **Both [Lists/Seq/Nempty]** | Binary operation chooses all possible outcomes, thus combining them |
| **Both [Sets/Maps]** | Binary operation combines both values, choosing from left when conflicts arise |
| **Neither** | None of the above concepts makes sense in this context |
| **~ a** | as a |
| **Empty** | mempty/zero value reflects empty container |