Type introduction illustrated

for Haskell newcomers

get over the foldable

Takenobu T.

Rev. 0.00.0
WIP

NOTE

- This shows one of the mental model.
- Please see also references.
- This is written for Haskell, especially later ghc8.

Contents

1. Introduction

Simple question

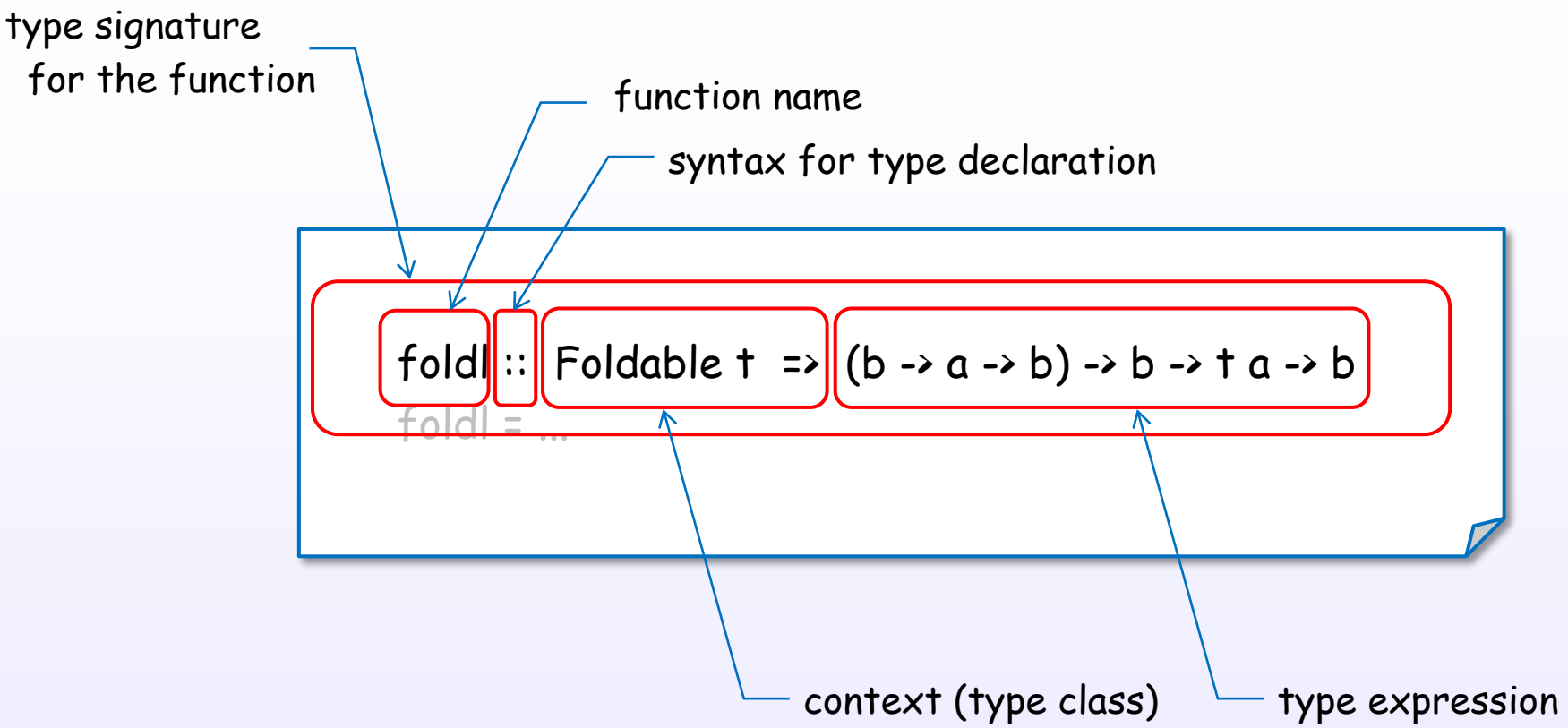
What is this ?!

```
foldl :: Foldable t => (b -> a -> b) -> b -> t a -> b
foldl = ...
```

?

References : @@@

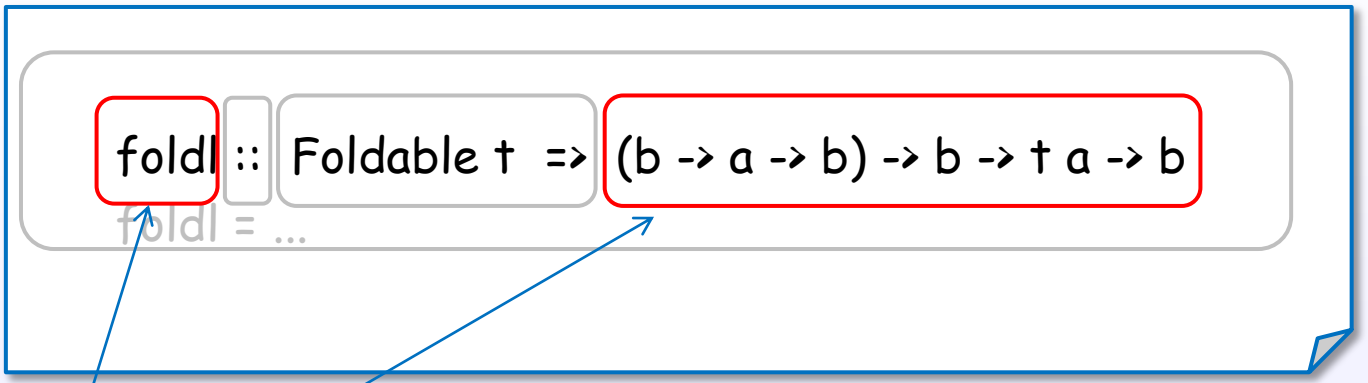
What is this ?!



[H1] 4.1

References : @@@

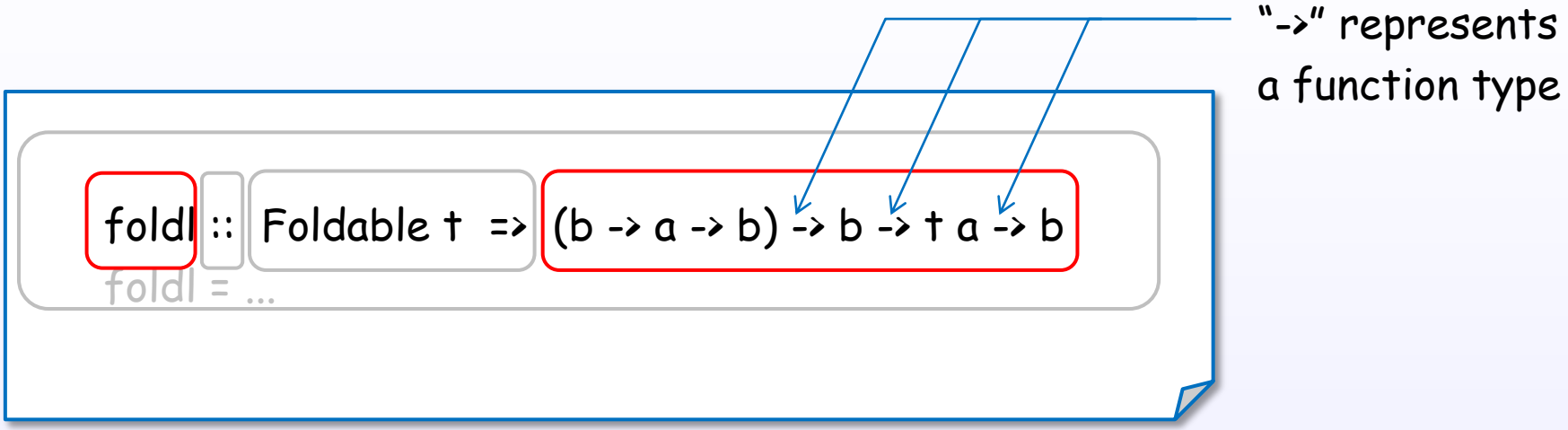
What is this ?!



"foldl" function has a type "(b -> a -> b) -> b -> t a -> b".

References : @@@

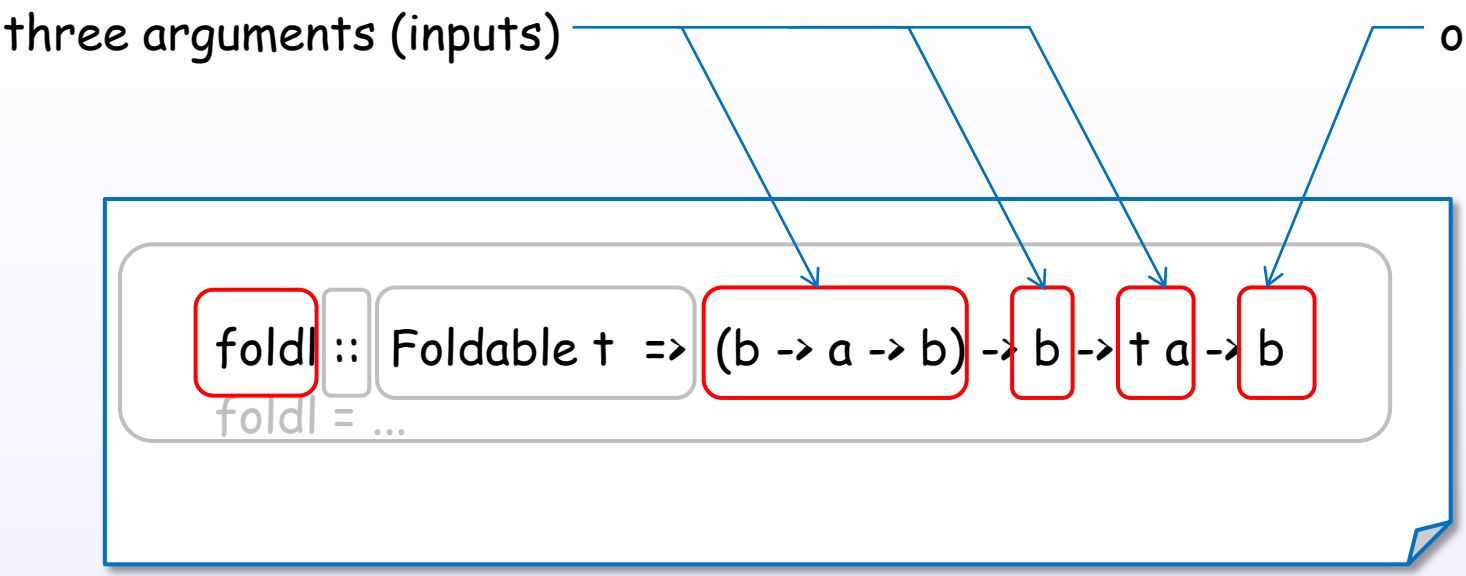
What is this ?!



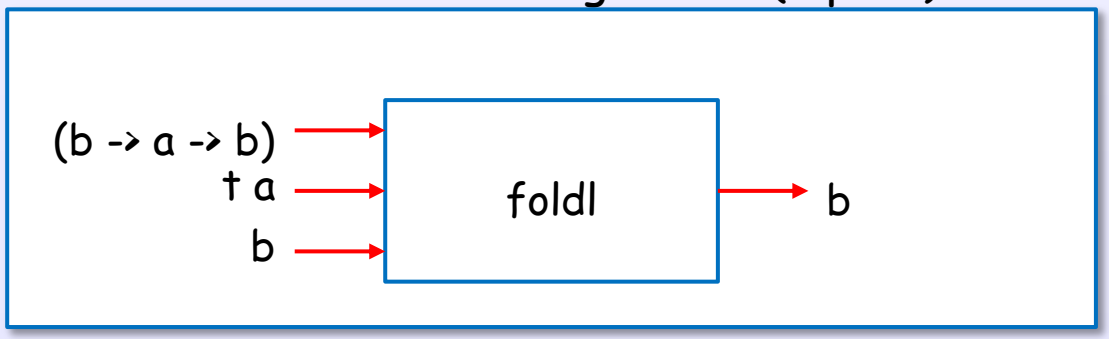
"foldl" is a function.

References : @@@

What is this ?!

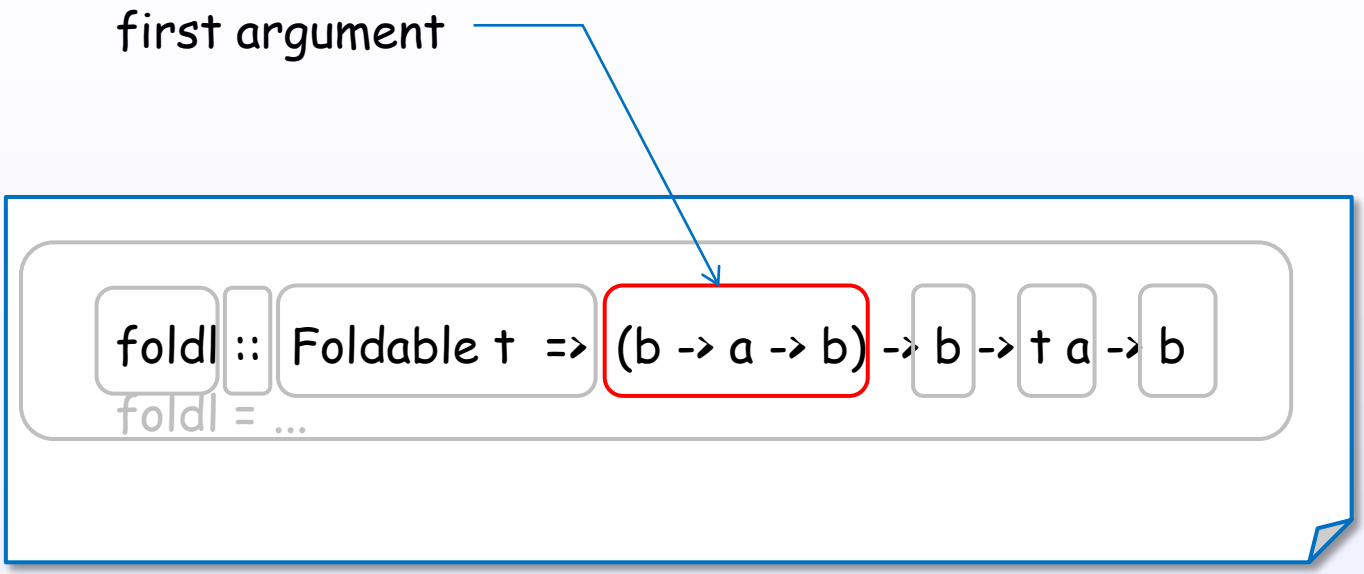


"foldl" function has three arguments(inputs) and one result(output).

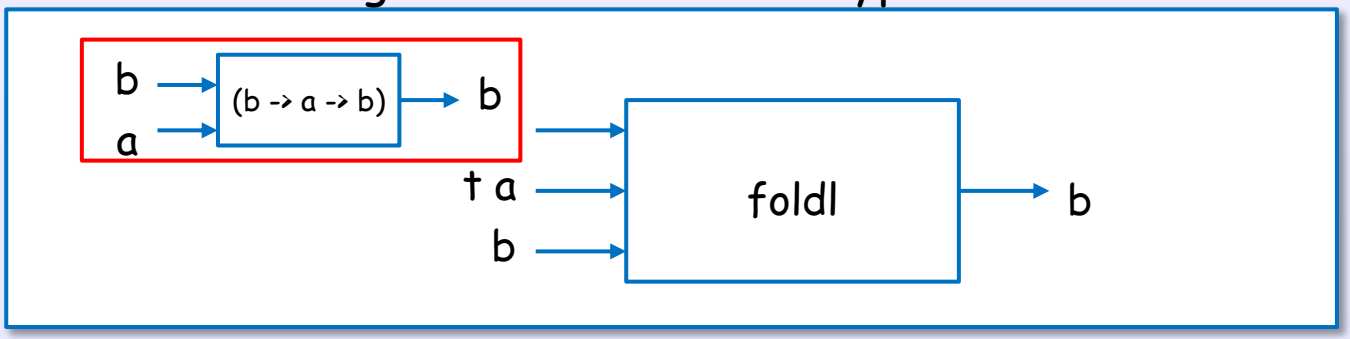


References : @@@

What is this ?!

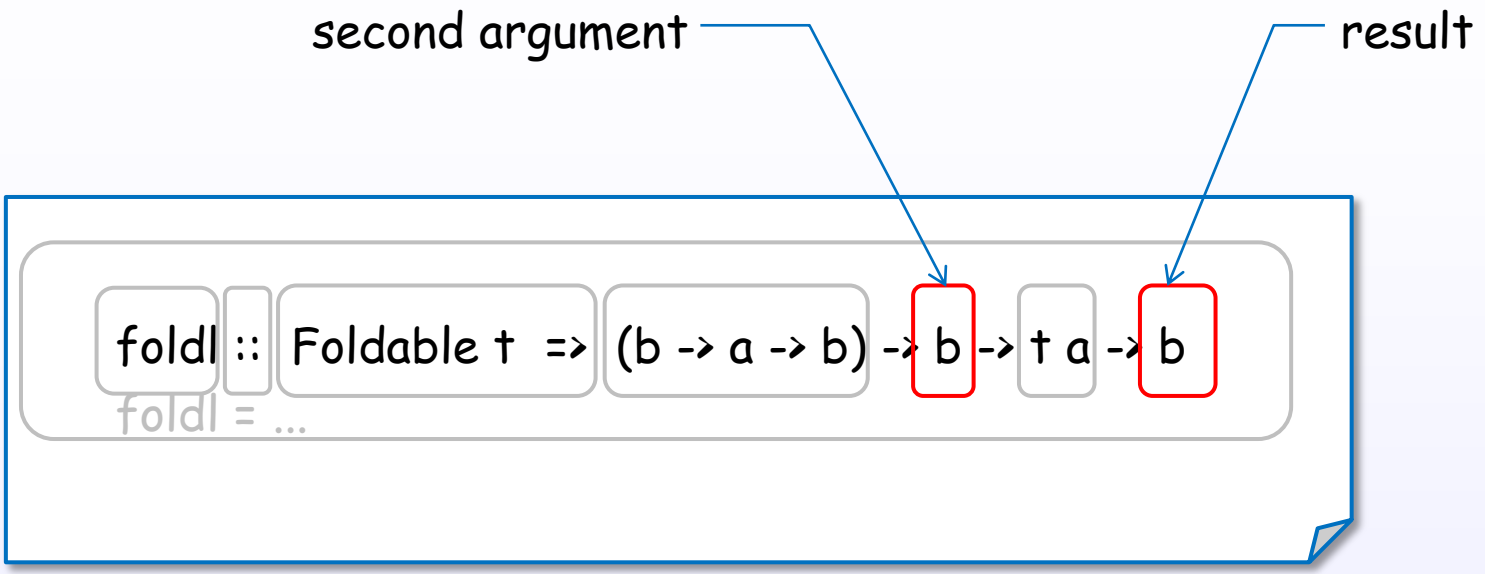


First argument is a function type.

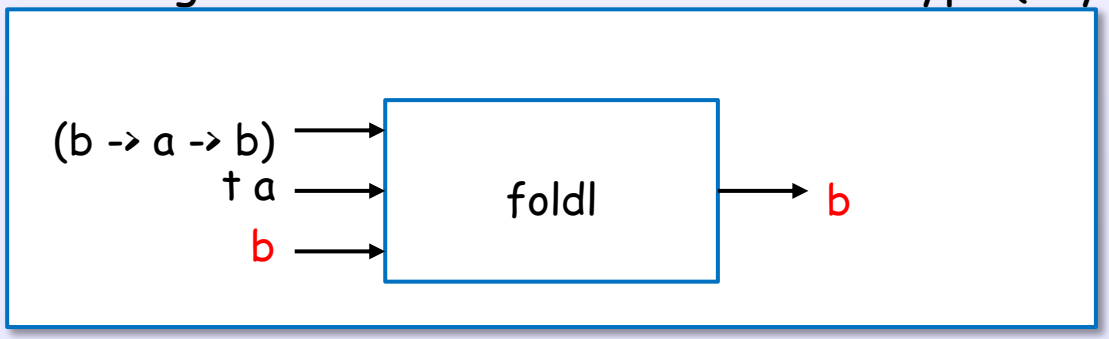


References : @@@

What is this ?!

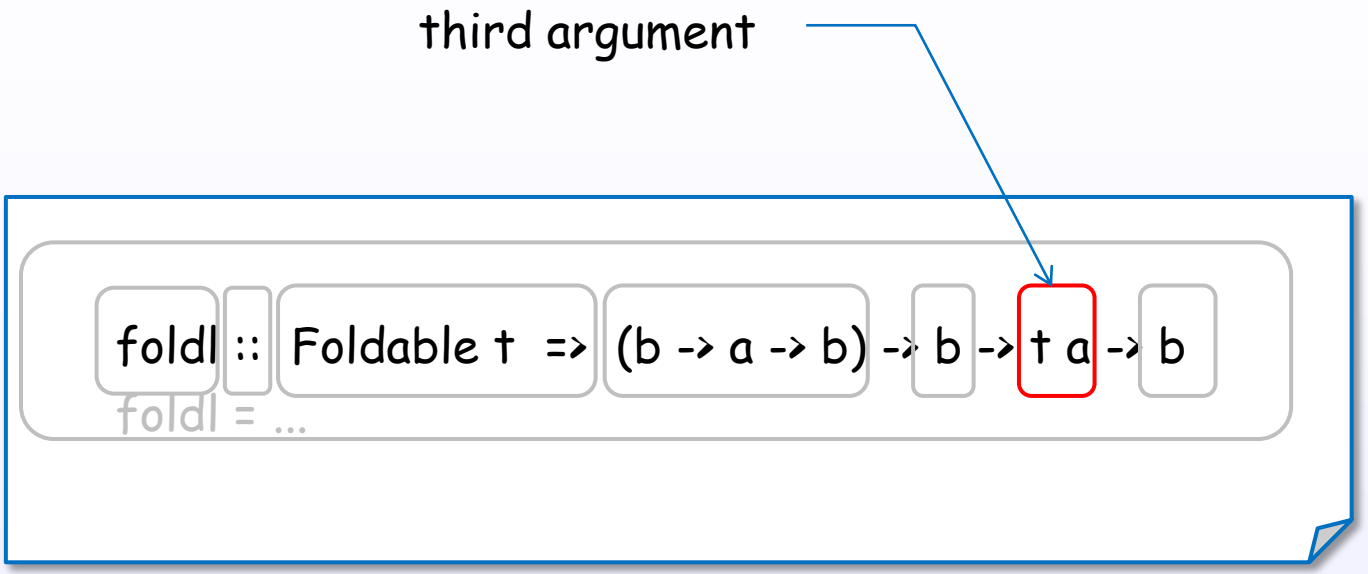


Second argument and the result are same type (any type "b").



References : @@@

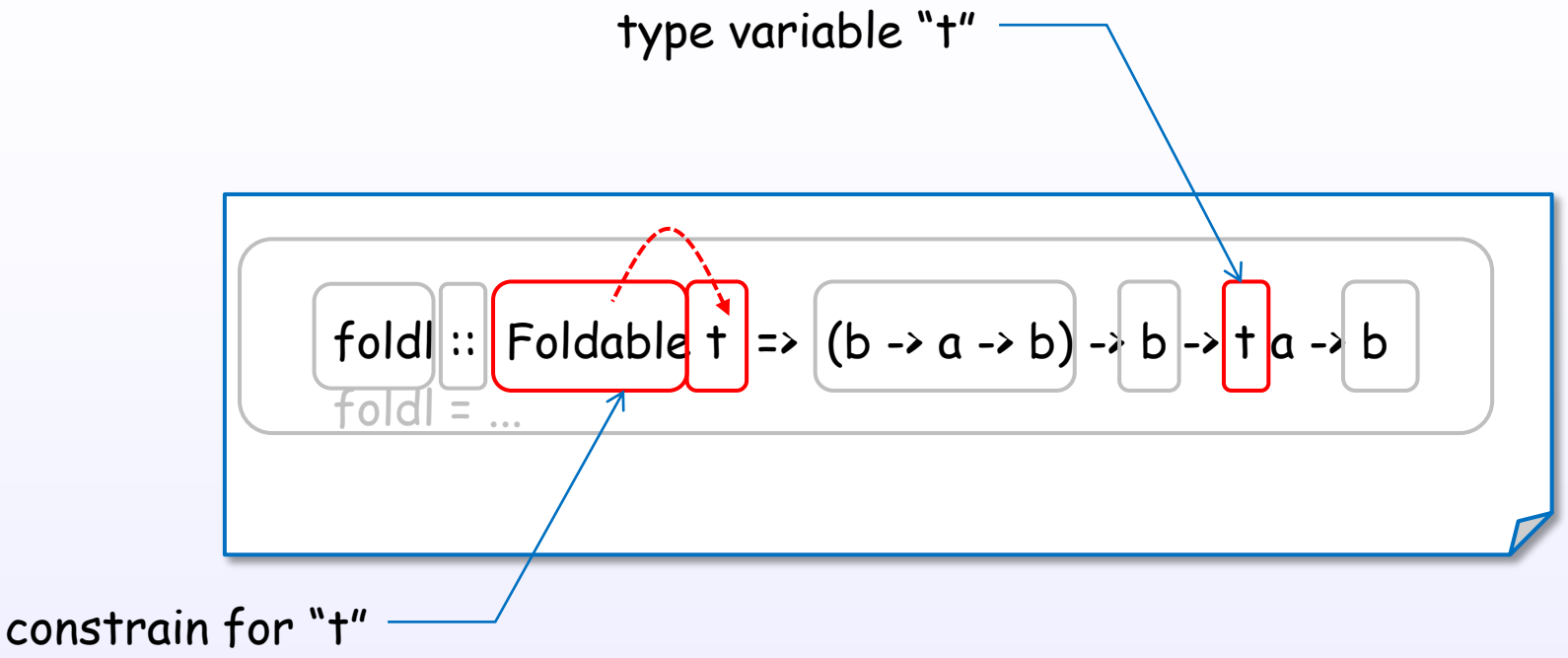
What is this ?!



Third argument is a constructed type with type variable “t” and “a”.

References : @@@

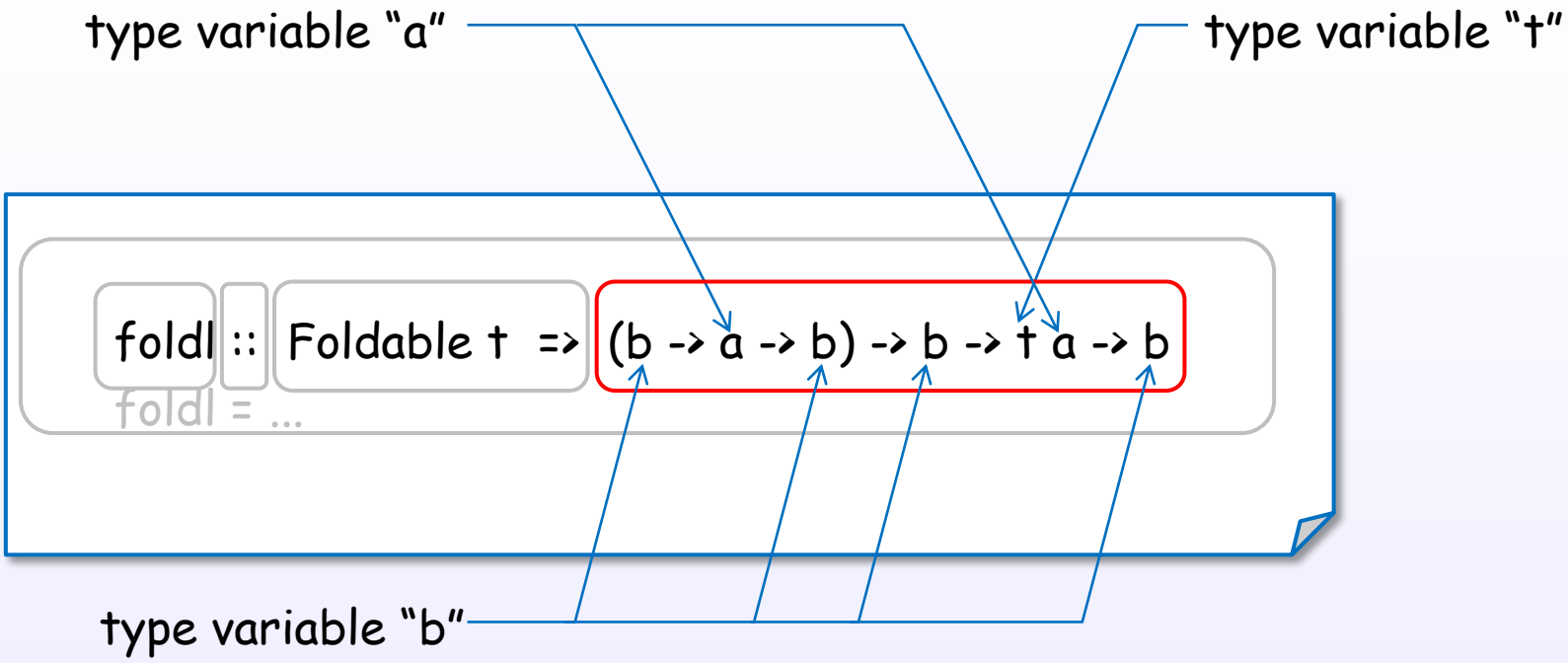
What is this ?!



The type variable "t" is belonged with "Foldable" typeclass.

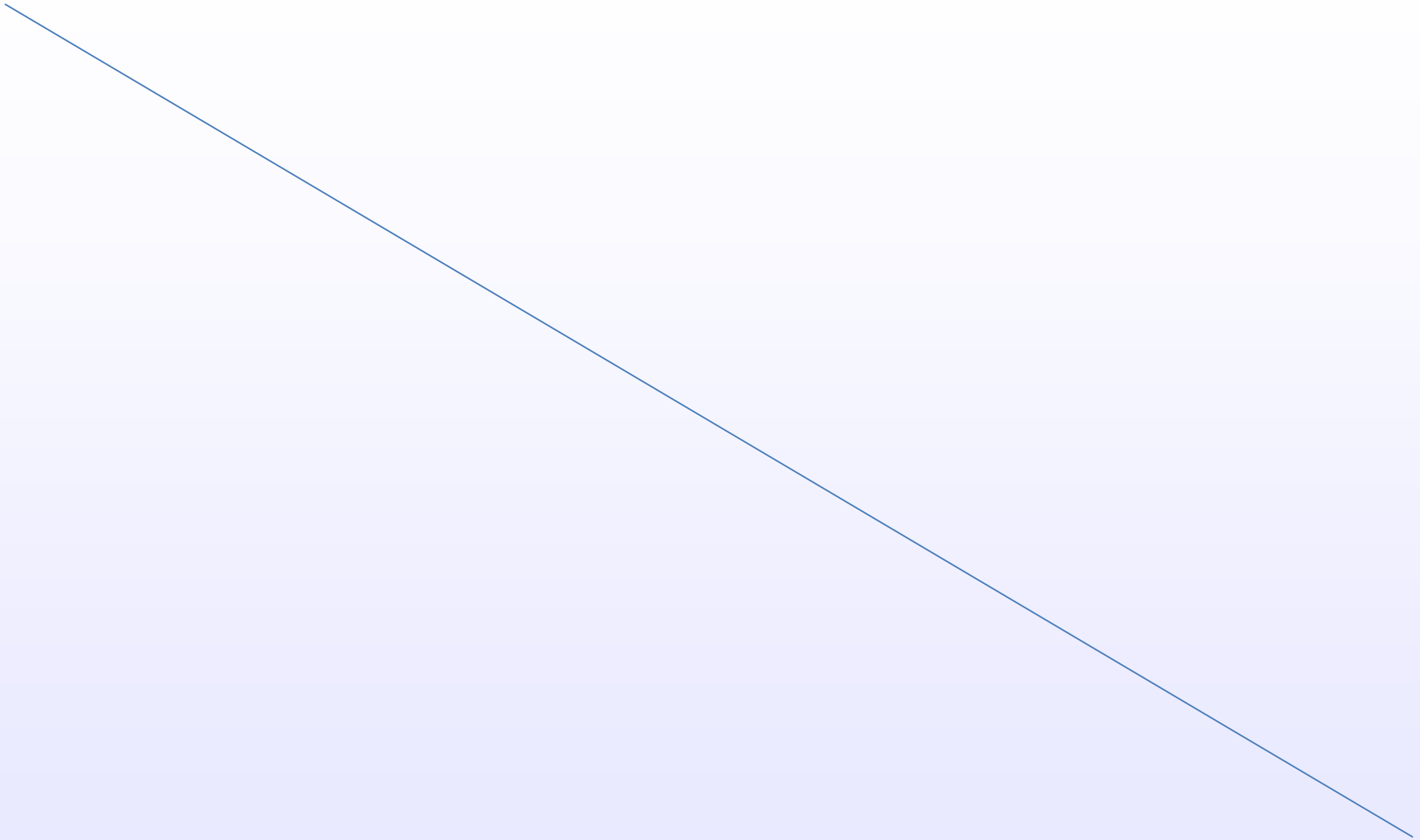
References : @@@

What is this ?!



“foldl” function has three type variables (“a”, “b” and “t”).
Type variable “a” and “b” is any type.
Type variable “t” is belonged with “Foldable” typeclass.

References : @@@



Value, Type, Typeclass

Typeclasses

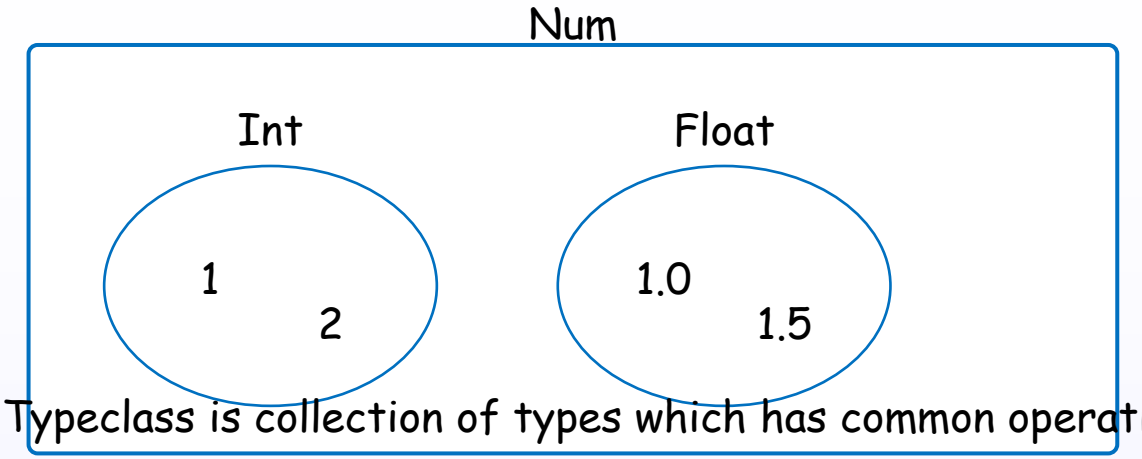
Types

Values

References : @@@

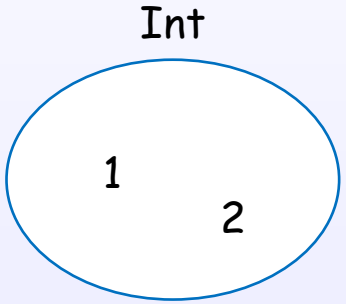
Value, Type, Typeclass

Typeclasses



Typeclass is collection of types which has common operation and XX.

Types



Type is collection of values which has common property.

Values

1

2

A

References : @@@

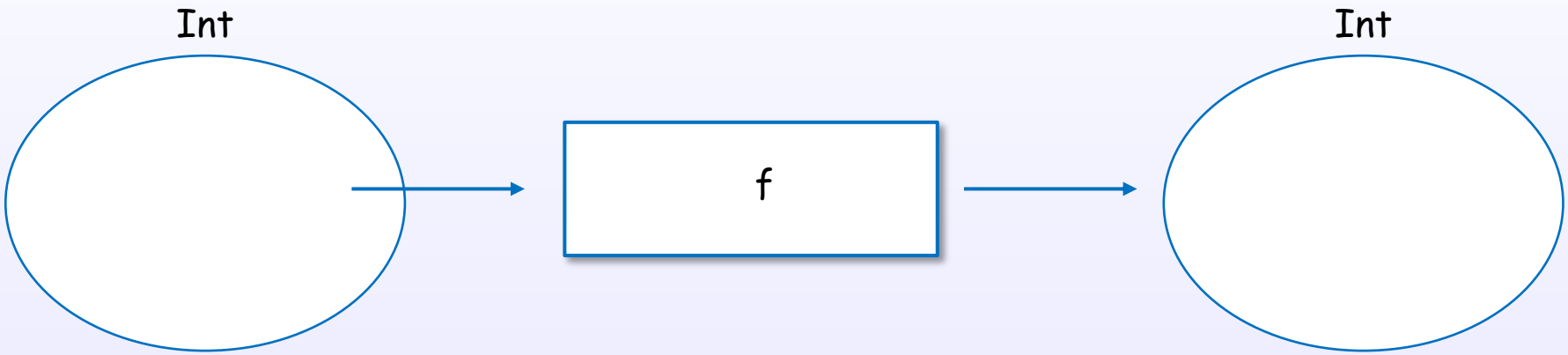
Value, Type, Typeclass

Typeclasses	a	Maybe a
Types	Num => a	Num => Maybe a
Values	Int	Maybe Int

References : @@@

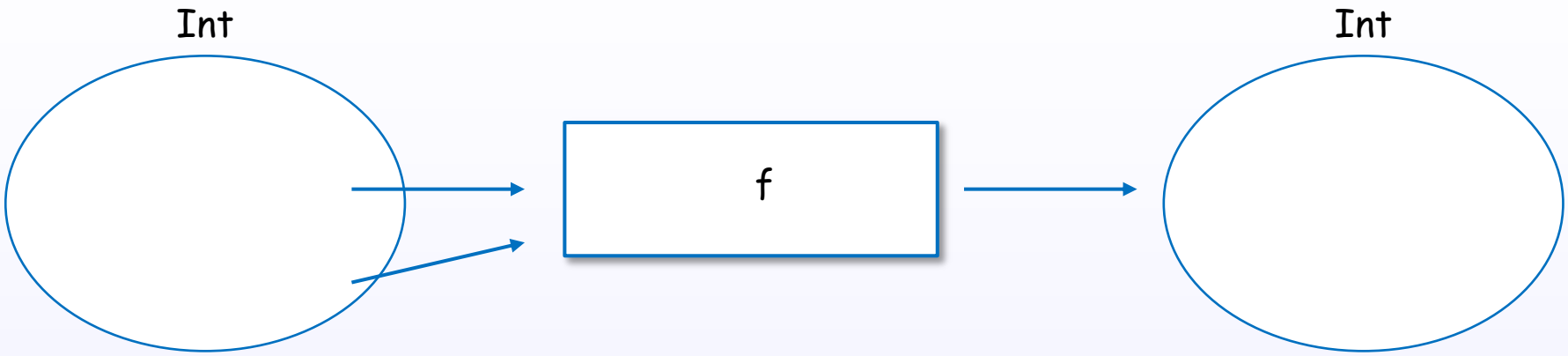
type

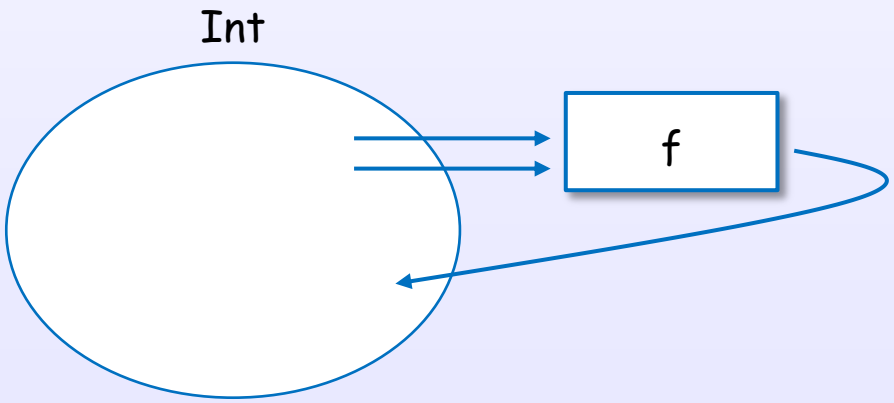
```
f :: Int -> Int
```



References : @@@

Each view



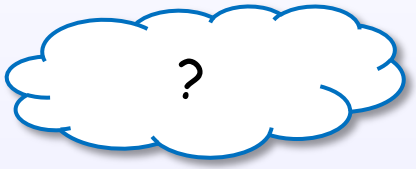


References : @@@

type

```
f :: a -> a
```

for all



All or concrete?

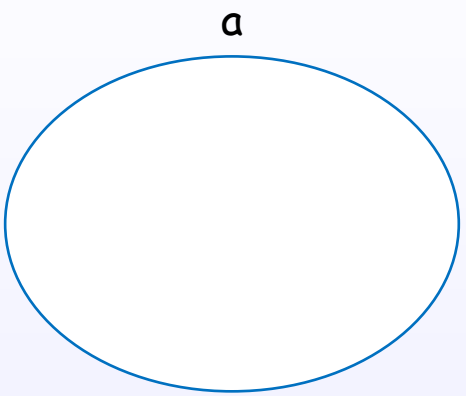
Are there intermediate?

concrete, specialize

```
f :: Int -> Int
```

References : @@@

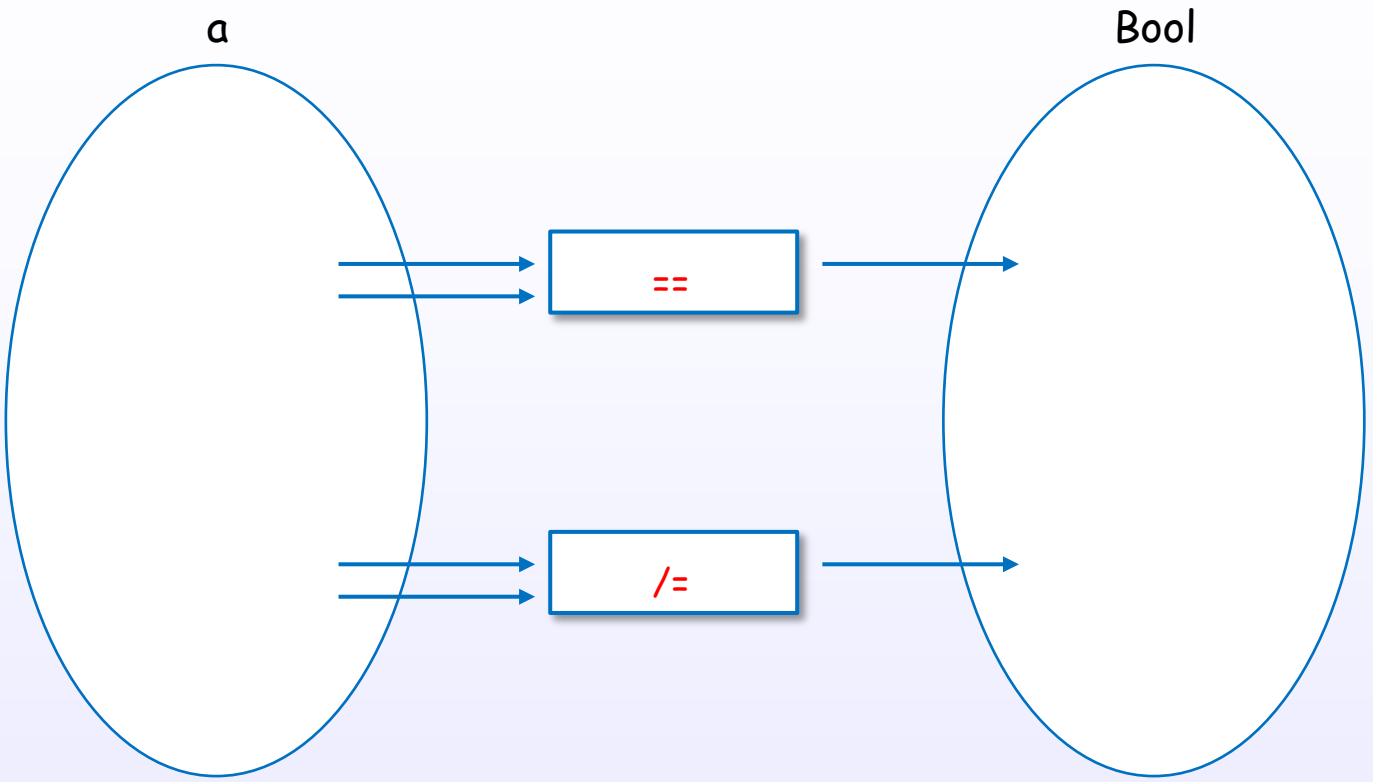
typeclass



class Num
(+) :: ...

References : @@@

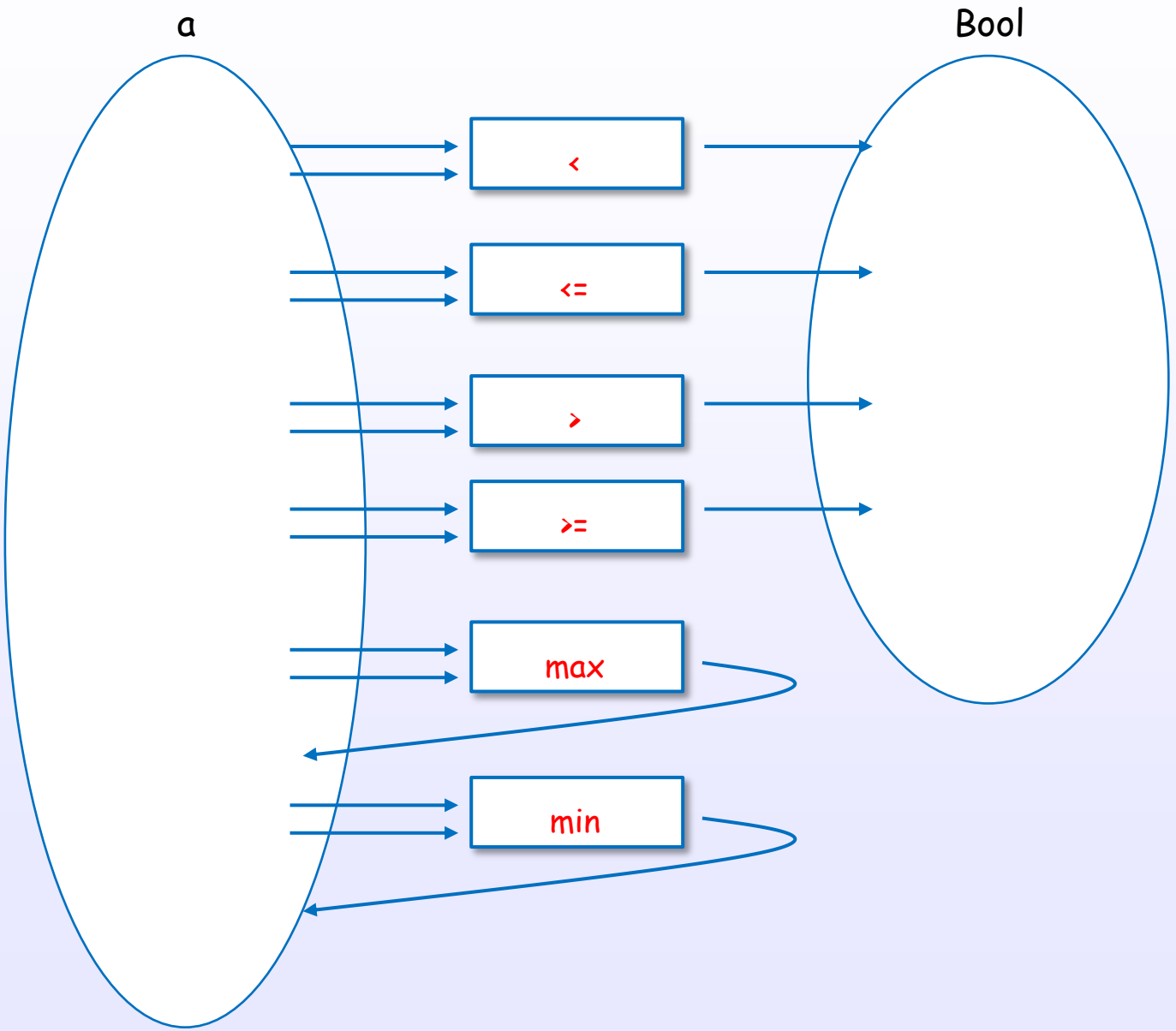
Eq class




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

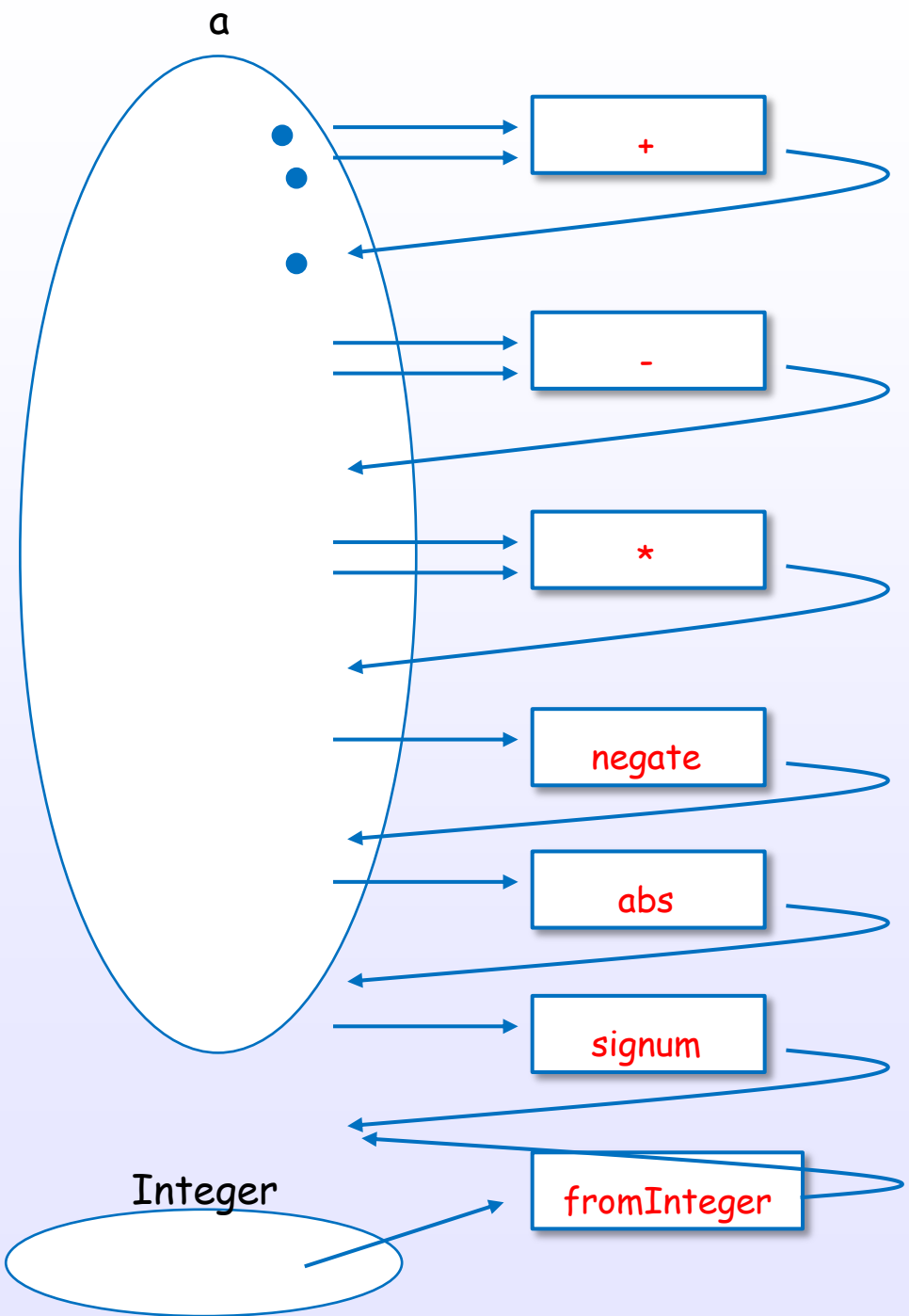
References : @@@

Ord class



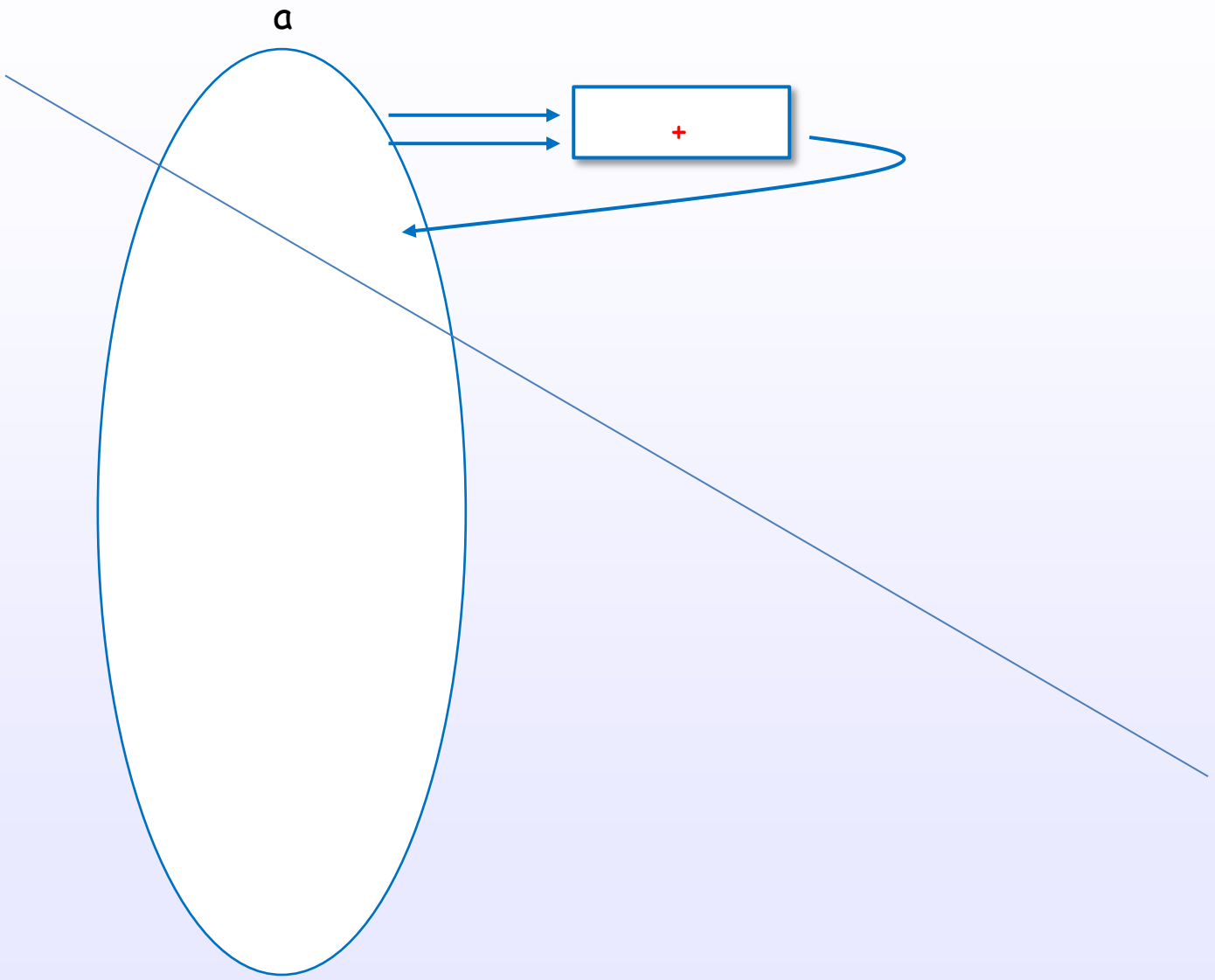
References : @@@

Num class



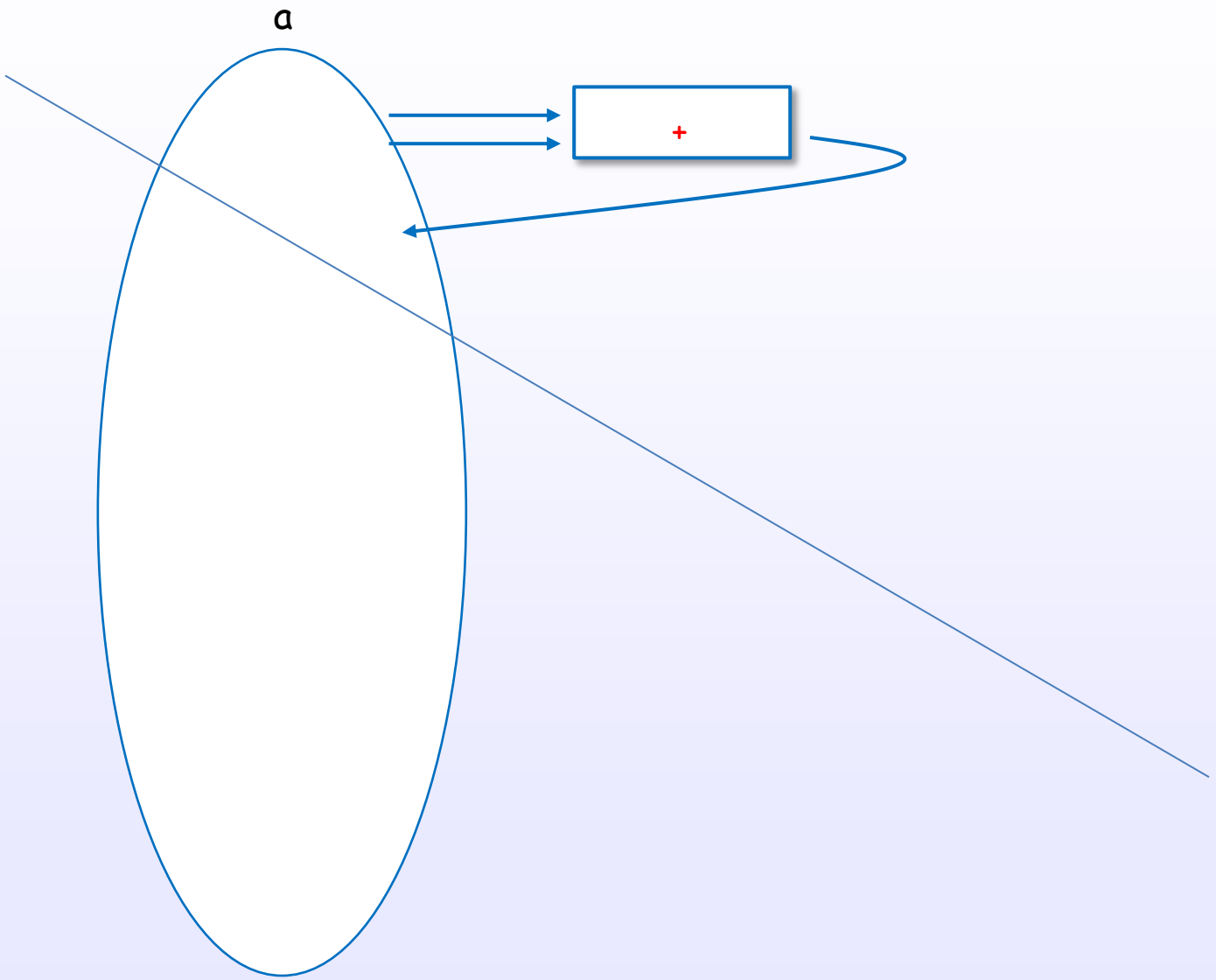
References : @@@

Foldable class



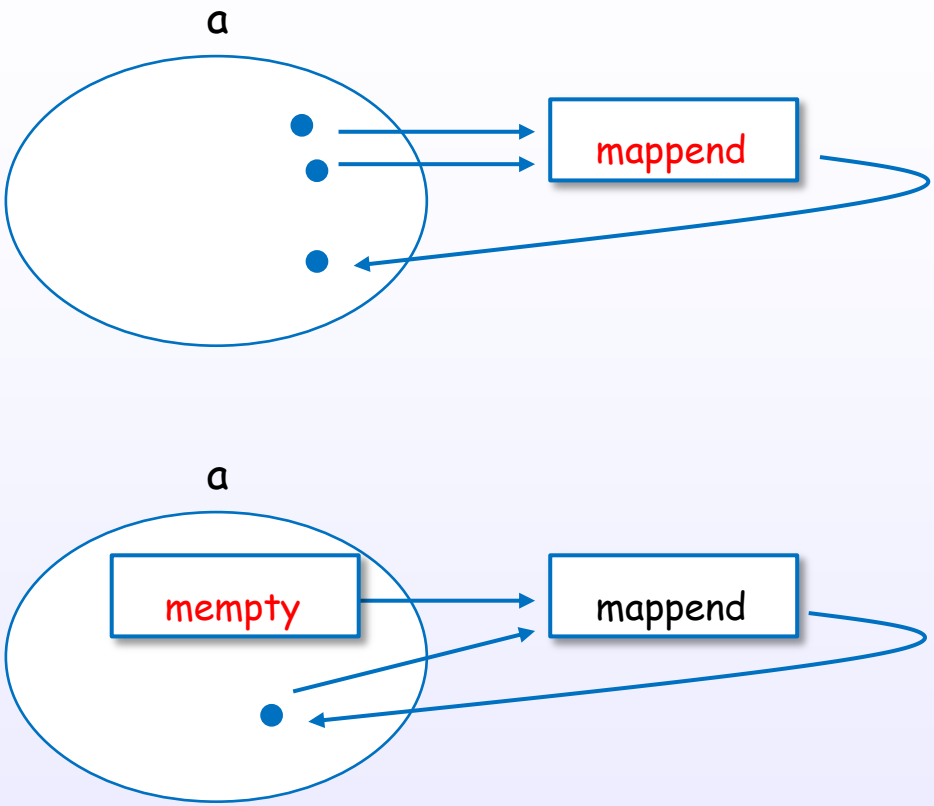
References : @@@

Traversable class



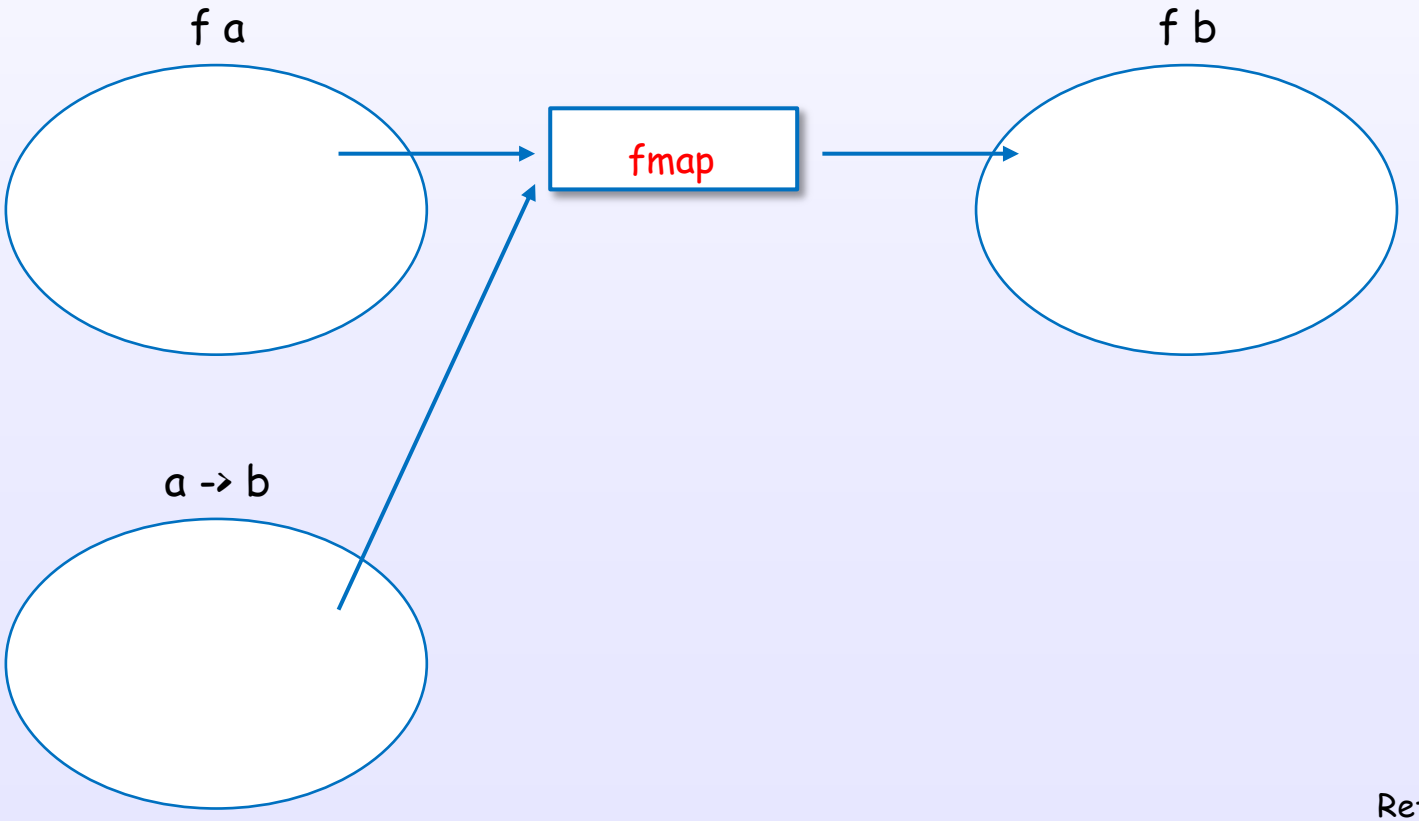
References : @@@

Monoid class



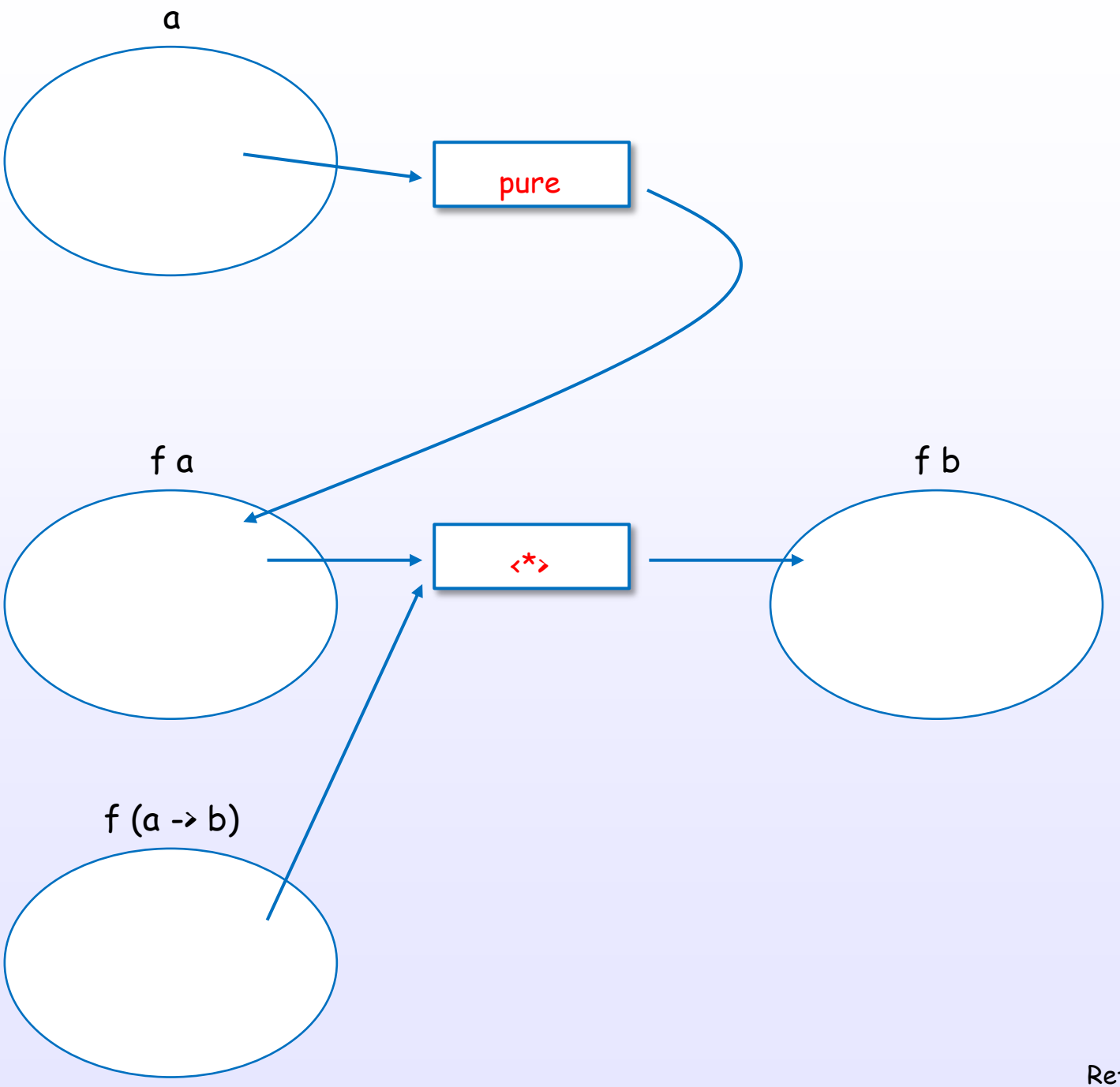
References : @@@

Functor class



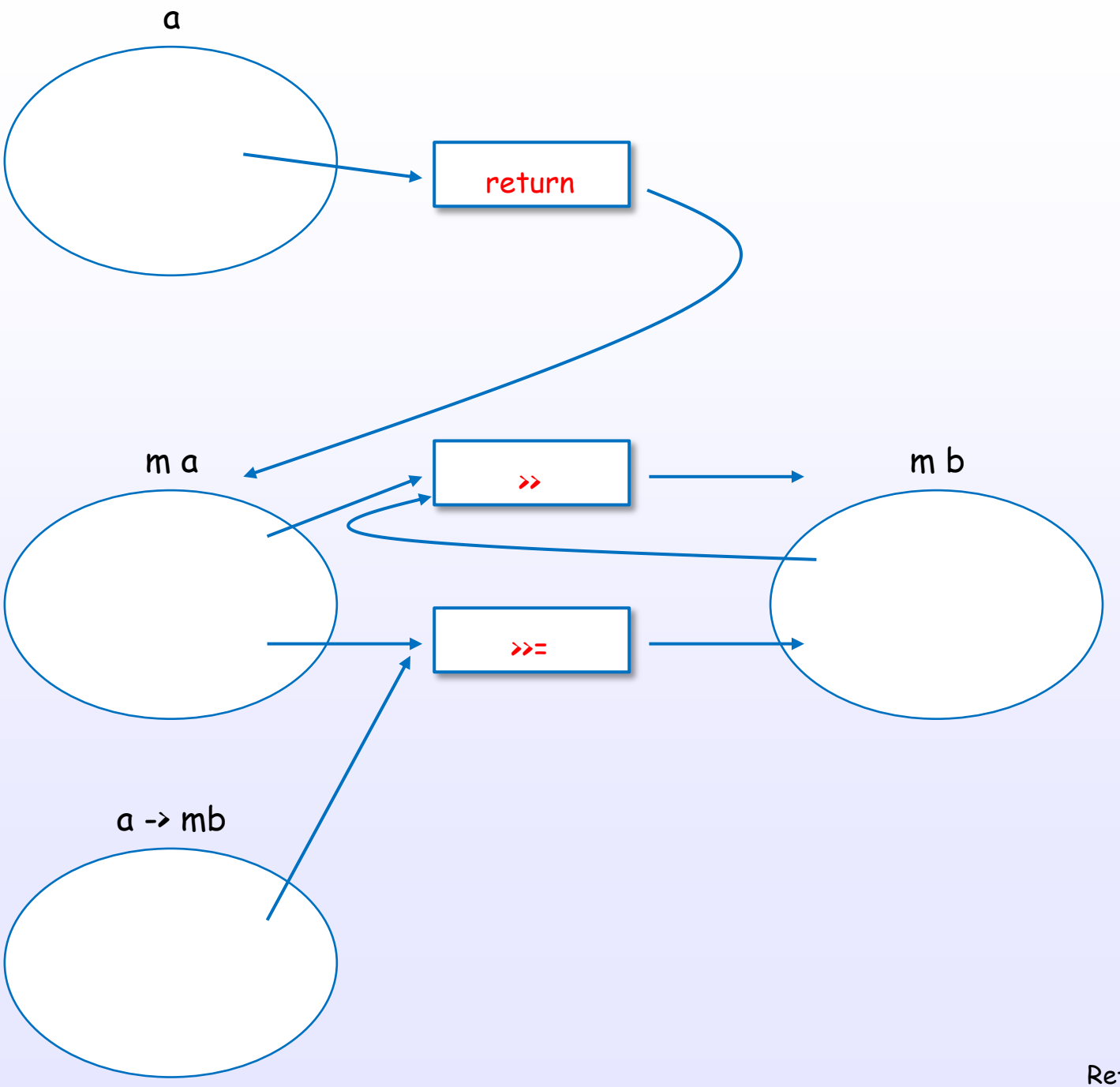
References : @@@

Applicative class



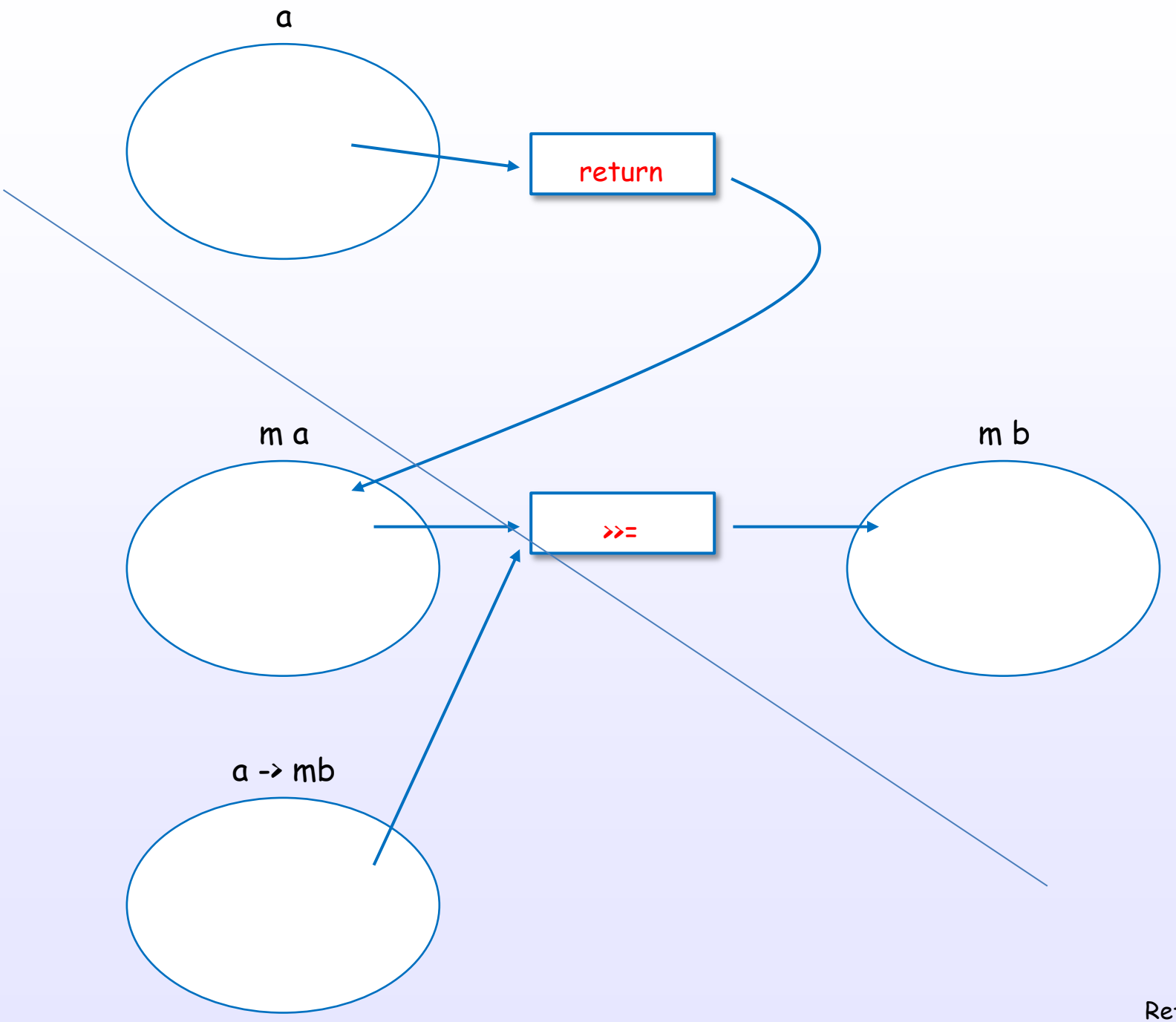
References : @@@

Monad class



References : @@@

Monad class



References : @@@

References

[B1] Learn You a Haskell for Great Good!
http://learnyouahaskell.com/

[B2] Thinking Functionally with Haskell (IFPH 3rd edition)
http://www.cs.ox.ac.uk/publications/books/functional/

[B3] Programming in Haskell
https://www.cs.nott.ac.uk/~gmh/book.html

[B4] Types and Programming Languages (TAPL)
https://mitpress.mit.edu/books/types-and-programming-languages

References

[D1] CIS 194: Introduction to Haskell
http://www.seas.upenn.edu/~cis194/lectures.html

[D2] Type Systems
http://dev.stephendiehl.com/fun/004_type_systems.html

[D3] Typeclassopedia
http://www.cs.tufts.edu/comp/150FP/archive/brent-yorgey/tc.pdf
https://wiki.haskell.org/Typeclassopedia

References

[S1] Hoogle
https://www.haskell.org/hoogle

References

[H1] Haskell 2010 Language Report
https://www.haskell.org/definition/haskell2010.pdf

[H2] The Glorious Glasgow Haskell Compilation System (GHC user's guide)
https://downloads.haskell.org/~ghc/latest/docs/users_guide.pdf