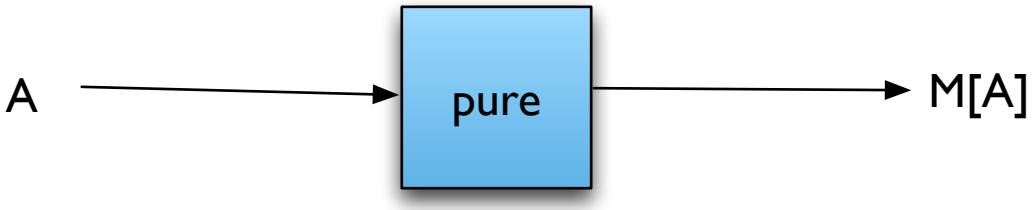
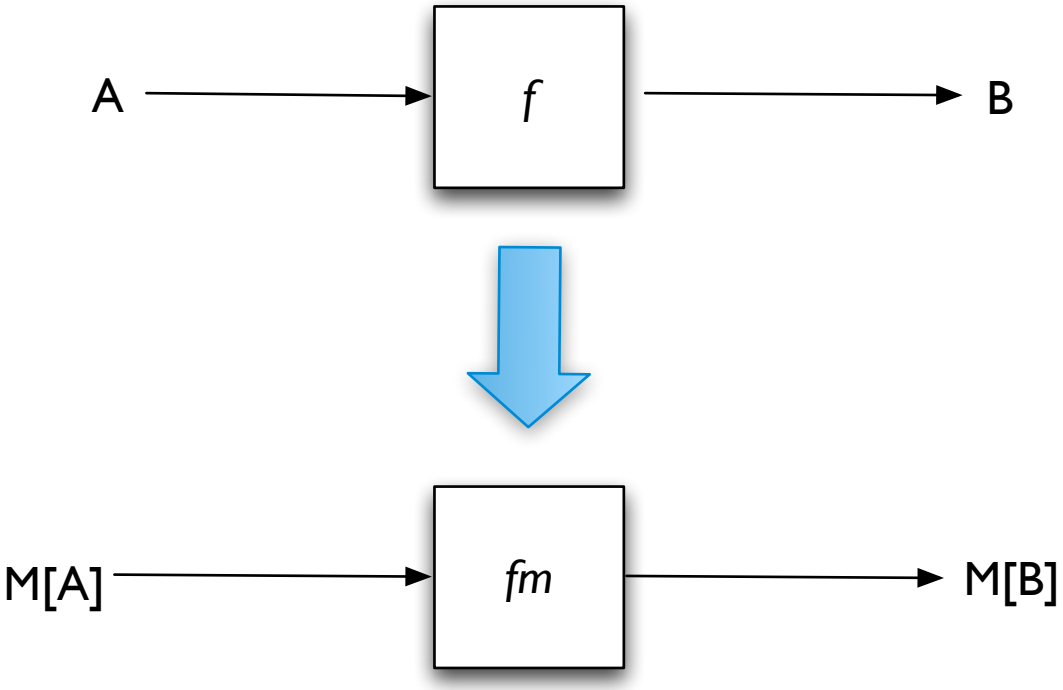
# Pure



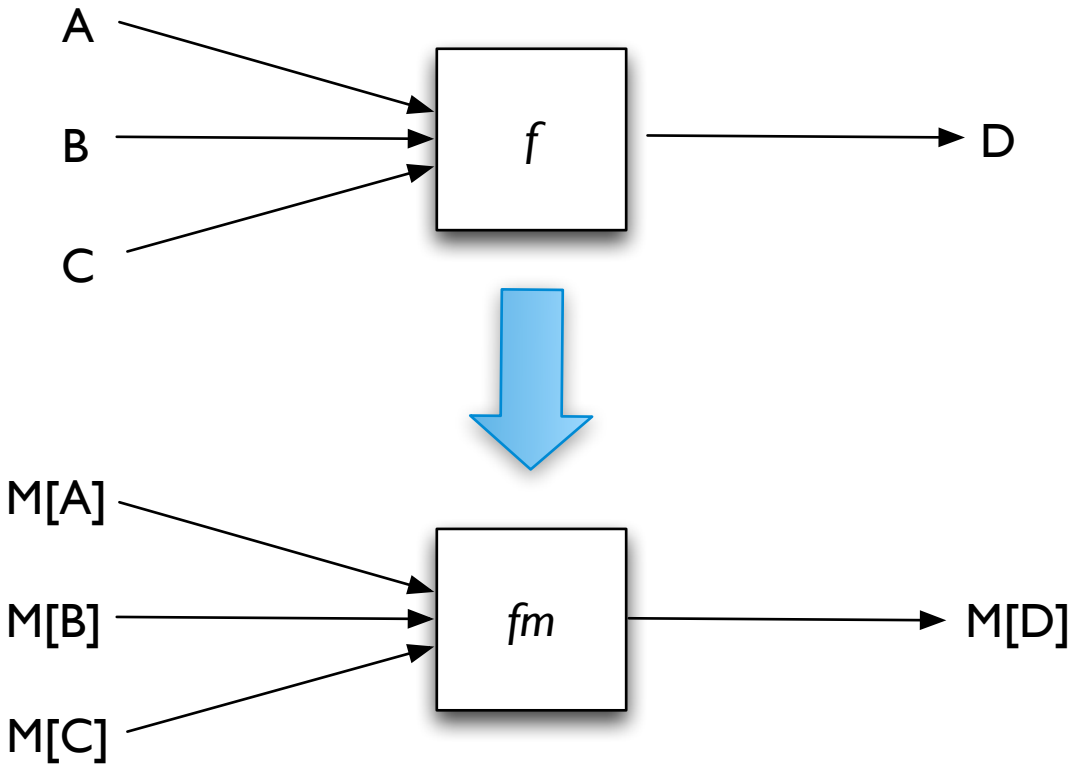

```
def wrap(a: A): M[A] = a.pure[M]
```

# Functor



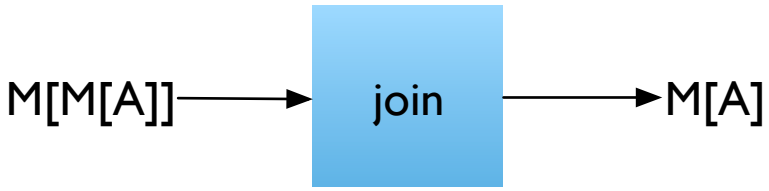

```
def fm(ma: M[A]) = ma map f
```

# Applicative Functor

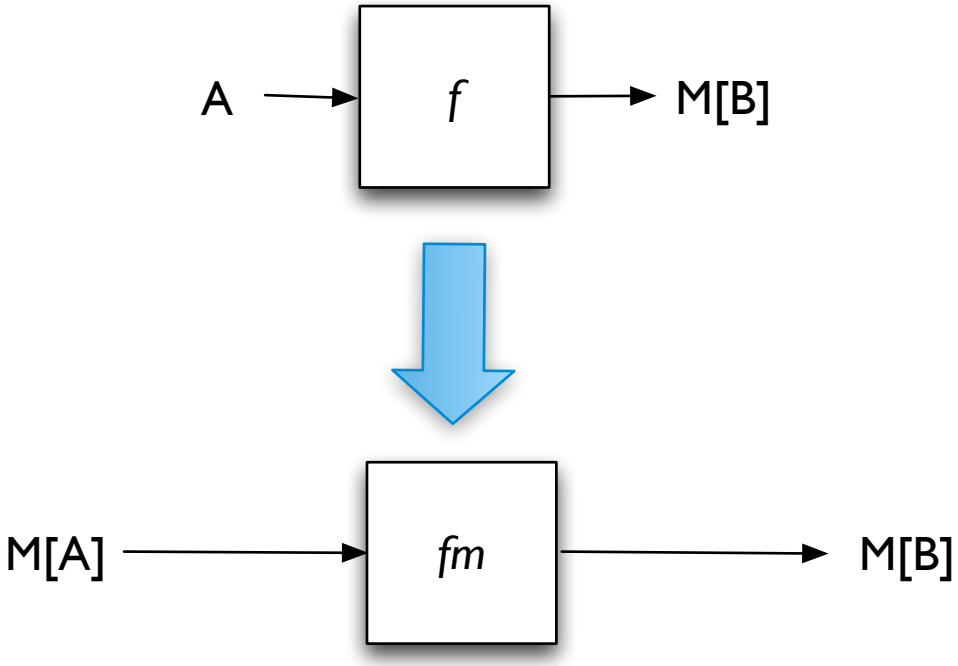



```
def fm(ma: M[A], mb: M[B], mc: M[C]): M[D]
          = (ma |@| mb |@| mc)(f)
```

# Monad

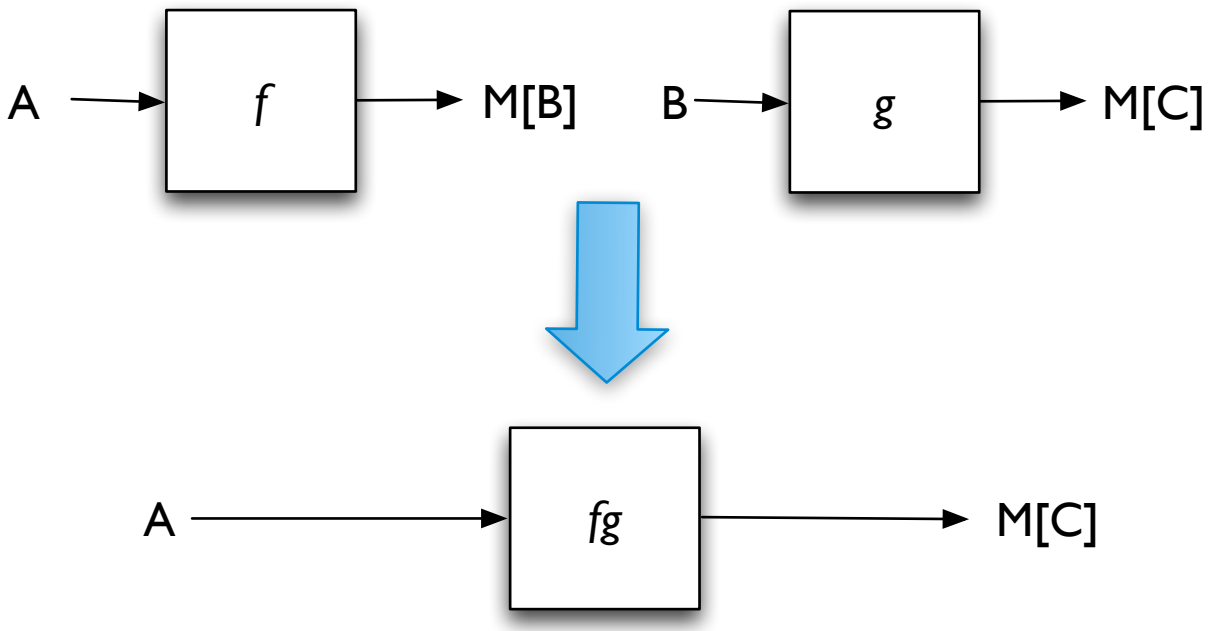


# Monad




```
def fm(ma: M[A]): M[B] = for {
  b <- f(a)
} yield b
```

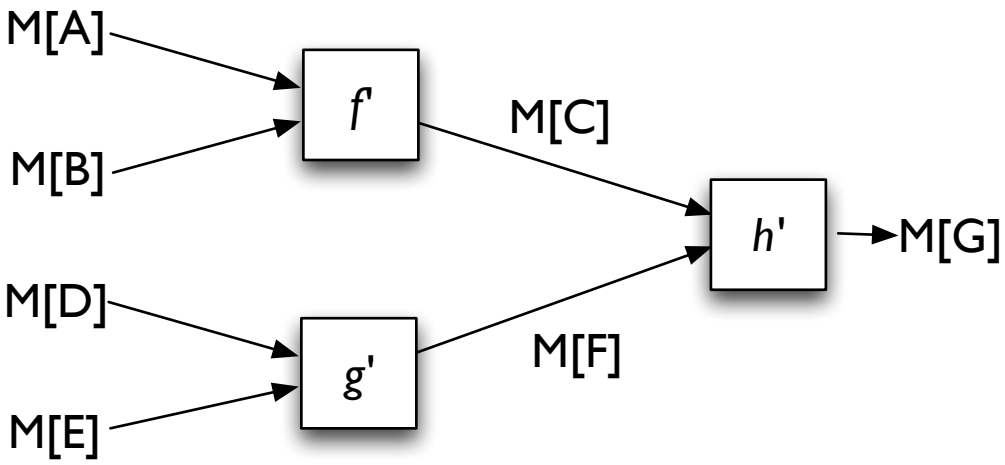
# Monad




```
def fg(a: A): M[C] = for {
  b <- f(a)
  c <- g(b)
} yield c

val fg = f >=> g // alternative
```

# Applicative Functor




```
val mc = (ma |@| mb)(f)
val mf = (me |@| md)(g)
val mg = (mc |@| mf)(h)
```