Funktor <*> Applicative

1. fmap id = id

= (pure id) <*>

= id // Identitätsgesetze der Abbildung

fmap id x = id x // id an x == id

= (pure id) <*> x

= x

2. fmap f . fmap g = fmap (f . g)

=> fmap f . fmap g x = fmap (f . g) x

= pure (f . g) <*> x

= pure ((.) f g) <*> x

= pure (.) <*> pure f <*> pure g <*> x

= (pure f) <*> ((pure . g) <*> x)

= pure f <*> . (pure g) <*> x

= pure f <*> . fmap g x

= fmap f . fmap g x

Monade Applicative

1.) pure id <*> v == v

= (return id) <*> v

= (return id) >>= (\x1 -> v >>= (\x2 -> return (x1 x2)

= v >>= (\x2 -> return (id x2))

= v // Eigenschaft Identitätsabbildung

2.) pure (.) <*> u <*> v <*> w

= pure (u.v) <*> w = fmap (u.v) w

= w >>= return . (u.v) = (w >>= (.) return (u.v)) >>= return

= w >>= (\x -> ((.) return (u.v)) >>= return)

= w >>= (\x -> ((.) return (u.v)) x)

= w >>= (\x -> return (u (v x))

= w >>= return . (\x -> (u (v x)))

= fmap (\x -> (u (v x))) w

= pure (\x -> (u (v x))) <*> w

= u <*> (v <*> w)

3.) pure f <*> pure x

= return f <*> return x

= (return f) >>= (\x1 -> return x >>= (\x2 -> return (x1 x2)

= return x >>= (\x2 -> return (f x2))

= return (f x)

= pure (f x)

4.) u <*> pure y

= u <*> return y

= u >>= (\x1 -> return y >>= (\x2 -> return (x1 x2)))

= u >>= (\x1 -> return (x1 y))

= u >>= (\x1 -> (\x2 -> return (x2 y)) x1 >>= return)

= (u >>= (\x2 -> return (x2 y))) >>= return

= u >>= (\x2 -> return (x2 y)))

= u >>= return . (\x1 -> (\x2 -> (x2 x1)) y

= fmap (\x2 -> (x2 y)) u

= pure (\x2 -> (x2 y)) <*> u

= pure ($ y) <*> u