Chapter 8: Applicative and traversable functors

Part 2: Their laws and structure

Sergei Winitzki

Academy by the Bay

2018-06-07

Motivation for applicative laws

- Derive applicative laws from monad laws

Monads perform effects *sequentially* even if effects are independent:

```
x ← Future { c1 }
y ← Future { c2 }
z ← Future { c3 }
```

```
Future { c1 }.flatMap { x ⇒
  Future { c2 }.flatMap { y ⇒
    Future { c3 }.map { z ⇒ ...}
} }
```

- We would like to parallelize independent computations
- We would like to accumulate *all* errors, rather than stop at the first one

Changing the order of monad's effects will (generally) change the result:

```
for {
  x ← List(1, 2)
  y ← List(10, 20)
} yield f(x, y)
// f(1, 10), f(1, 20), f(2, 10), f(2, 20)
```

```
for {
  y ← List(10, 20)
  x ← List(1, 2)
} yield f(x, y)
// f(1, 10), f(2, 10), f(1, 20), f(2, 20)
```

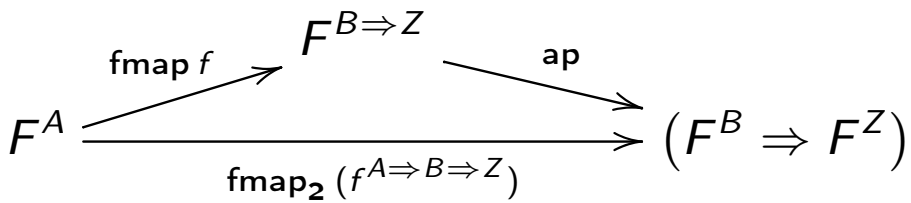
- We would like to express a computation where effects are unordered
  - This can be done using a method `map2`, *not* defined via `flatMap`: the desired type signature is map2 : $F^A \times F^B \Rightarrow (A \times B \Rightarrow C) \Rightarrow F^C$
  - An **applicative functor** has `map2` but is not necessarily a monad

Deriving the `ap` operation from `map2`

Can we avoid having to define map_n separately for each n?

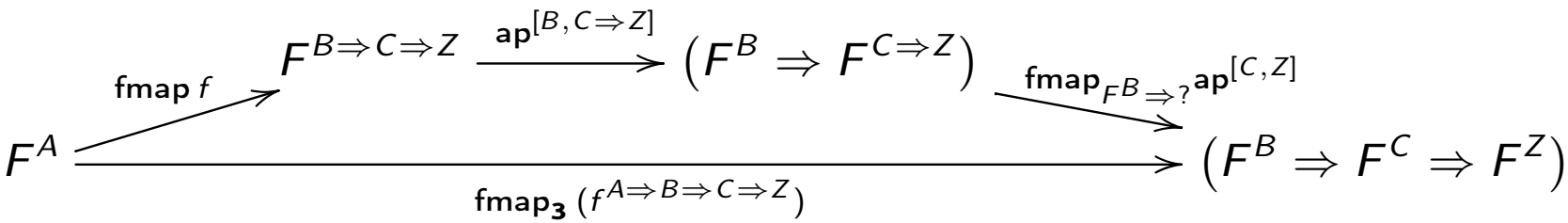
- Use curried arguments, $\text{fmap}_2 : (A \Rightarrow B \Rightarrow Z) \Rightarrow F^A \Rightarrow F^B \Rightarrow F^Z$
- Set $A = B \Rightarrow Z$ and apply fmap_2 to the identity $\text{id}^{(B \Rightarrow Z) \Rightarrow (B \Rightarrow Z)}$: obtain $\text{ap}^{[B,Z]} : F^{B \Rightarrow Z} \Rightarrow F^B \Rightarrow F^Z \equiv \text{fmap}_2\,(\text{id})$
- The functions `fmap2` and `ap` are computationally equivalent:

$$\text{fmap}_2\, f^{A \Rightarrow B \Rightarrow Z} = \text{fmap}\, f \circ \text{ap}$$



- The functions `fmap3`, `fmap4` etc. can be defined similarly:

$$\text{fmap}_3\, f^{A \Rightarrow B \Rightarrow C \Rightarrow Z} = \text{fmap}\, f \circ \text{ap} \circ \text{fmap}_{F^B \Rightarrow ?}\text{ap}$$



Intuition: the `zip` operation on lists

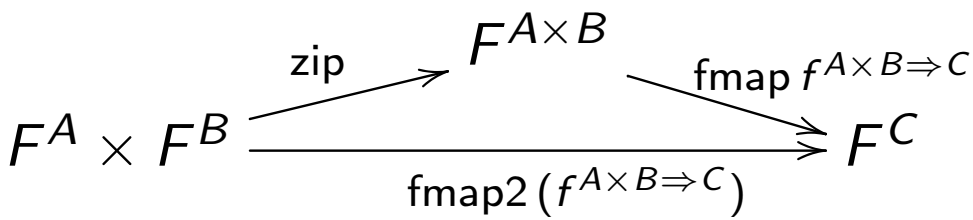
- Note: Function types $A \Rightarrow B \Rightarrow C$ and $A \times B \Rightarrow C$ are equivalent
- Uncurry fmap_2 to $\text{fmap2} : (A \times B \Rightarrow C) \Rightarrow F^A \times F^B \Rightarrow F^C$
- Compute $\text{fmap2}(f)$ with $f = \text{id}^{A \times B \Rightarrow A \times B}$, expecting to obtain a simpler natural transformation:

$$\text{zip} : F^A \times F^B \Rightarrow F^{A \times B}$$

- This is quite similar to `zip` for lists:
  `List(1, 2).zip(List(10, 20)) = List((1, 10), (2, 20))`
- The functions `zip` and `fmap2` are computationally equivalent:

$$\text{zip} = \text{fmap2}(\text{id})$$

$$\text{fmap2}(f^{A \times B \Rightarrow C}) = \text{zip} \circ \text{fmap}\, f$$



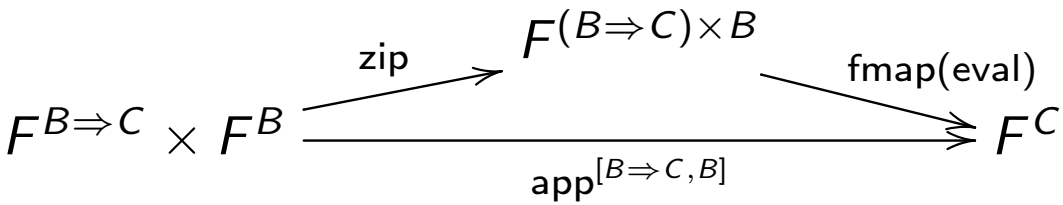
- The functor F is "zippable" if such a `zip` exists

Deriving the `ap` operation from `zip`

- Set $A \equiv B \Rightarrow C$, get $\text{zip}^{[B \Rightarrow C, B]} : F^{B \Rightarrow C} \times F^B \Rightarrow F^{(B \Rightarrow C) \times B}$
- Use `eval` : $(B \Rightarrow C) \times B \Rightarrow C$ and $\text{fmap}\,(\text{eval}) : F^{(B \Rightarrow C) \times B} \Rightarrow F^C$
- Define $\text{app}^{[B,C]} : F^{B \Rightarrow C} \times F^B \Rightarrow F^C \equiv \text{zip} \circ \text{fmap}\,(\text{eval})$
- The functions `zip` and `app` are computationally equivalent:
  - use $\text{pair} : (A \Rightarrow B \Rightarrow A \times B) = a^A \Rightarrow b^B \Rightarrow a \times b$
  - use $\text{fmap}\,(\text{pair}) \equiv \text{pair}^\uparrow$ on an fa^{F^A}, get $(\text{pair}^\uparrow fa) : F^{B \Rightarrow A \times B}$; then

$$\text{zip}\,(fa \times fb) = \text{app}\left((\text{pair}^\uparrow fa) \times fb\right)$$

$$\text{app}^{[B \Rightarrow C, B]} = \text{zip}^{[B \Rightarrow C, B]} \circ \text{fmap}\,(\text{eval})$$



- Rewrite this using curried arguments: $\text{fzip}^{[A,B]} : F^A \Rightarrow F^B \Rightarrow F^{A \times B}$; $\text{ap}^{[B,C]} : F^{B \Rightarrow C} \Rightarrow F^B \Rightarrow F^C$; then $\text{ap}\, f = \text{fzip}\, f \circ \text{fmap}\,(\text{eval})$.
- Now $\text{fzip}\, p^{F^A} q^{F^B} = \text{ap}\left(\text{pair}^\uparrow p\right) q$, hence we can write as point-free: $\text{fzip} = \text{pair}^\uparrow \circ \text{ap}$. With explicit types: $\text{fzip}^{[A,B]} = \text{pair}^\uparrow \circ \text{ap}^{[B, A \Rightarrow B]}$.