Chapter 11: Computations in a functor context III. Monad transformers

Sergei Winitzki

Academy by the Bay

2019-01-05

Computations within a functor context: Combining monads

- Programs need to combine monadic effects
- “Effect” $\equiv$ what else happens in $A \Rightarrow M^B$ besides computing B from A
- Examples of effects for some standard monads:
  - `Option` – computation will have no result or a single result
  - `List` – computation will have zero, one, or multiple results
  - `Either` – computation may fail to obtain its result
  - `Reader` – computation needs to read an external context value
  - `Writer` – some value will be appended to a (monoidal) log
  - `Future` – computation will be scheduled to run later
- How to combine several effects in the same functor block (`for`/`yield`)?

```
// This is not valid Scala!
val result = for { i ← 1 to n
   j ← Future { q(i) }
   k ← maybeError(j)
} yield f(k)
// What should be the type of result??
```

```
// This is not valid Scala!
(1 to n).flatMap { i ⇒
   Future(q(i)).flatMap { j ⇒
     maybeError(j).map { k ⇒
       f(k)
         }}}
```

- The code will work if we “unify” all effects in a new, larger monad
- Need to compute the type of new monad that contains all given effects

How to combine monadic effects

There are several ways of combining two monads into a new monad

- If M_1^A and M_2^A are monads then $M_1^A \times M_2^A$ is also a monad
    - But $M_1^A \times M_2^A$ describes two separate values with two separate effects
- If M_1^A and M_2^A are monads then $M_1^A + M_2^A$ is usually not a monad
    - If it worked, it would be a choice between two different values / effects
- If M_1^A and M_2^A are monads then one of $M_1^{M_2^A}$ or $M_2^{M_1^A}$ is often a monad
- Examples and counterexamples:
    - Combine `Future[A]` and `Option[A]` as `Future[Option[A]]`
    - Combine $Z \Rightarrow A$ and List^A as $Z \Rightarrow \text{List}^A$
    - But `Either[Z, Future[A]]` and `Option[Z ⇒ A]` are not monads
    - Neither `Future[State[A]]` nor `State[Future[A]]` are monads
- The order of effects matters when composition works both ways:
    - Combine `Either` ($M_1^A = Z + A$) and `Writer` ($M_2^A = W \times A$)
        - as $Z + W \times A$ – either compute result and write a message, or all fails
        - as $(Z + A) \times W$ – message is always written, but computation may fail
- Find a general way of defining a new monad with combined effects
- Derive properties required for the new monad

Exercises

1. For a given