Week3 Paper Summary——Monad
By Yi Zhou

Haskell and some other pure functional languages don't support some impure features such as global variables which could save the state. However, such features could be quite important under some circumstance. Sometimes a small change may require a pure language program to be extensively restructured. This week's paper provides an elegant way called monad for us to solve these problems in pure functional languages.

According to the paper, a monad is a triple(M, unitM, bindM) consisting of a type constructor M, which we normally call constructor in OOP, and a pair of polymorphic functions:

unitM :: a->M a

bindM :: M a -> (a -> Mb) -> M b

M is a type constructor that specifies how the monadic type is build up from other preexisting type. unitM is used to wrap underlying variable in monads and bindM could be used a combinator, which could unwrap the variable from the monad-typed variable and then inserts it into a function. Then the paper uses some examples to show how these three things work together to implement some features in a much more clear and easy way. Besides, monads could also be used to realize continuation-passing style. The paper also shows how could monads be applied in a compiler for Haskell.

From my perspective, monad is quite an awesome method to implement some impure language features by using pure functional languages. As I know, some languages, such as Haskell, as embedded monad into its standard library. However, it' s quite a difficult concept for beginners to understand. As a programming novice, maybe OOP language could be a better choice. But, there's no denying that, mathematicians will enjoy using pure functional languages with such feature.