Week3 Summary

General idea of this paper:

This paper generally introduced the use of monads to structure functional programs. Monads make modifying the program easier and also provide effects not easily achieved with impure features including exceptions, state and continuations. The article first gave an extended example of the use of monads and then describes the relations between the monads and continuation-passing styles. In the last part, the author illustrates how monads are applied in writing a complier for Haskell that is written in Haskell.

Details about the paper:

In the beginning, the author introduced the monads by comparing the characteristics of pure and impure functional languages, monads can avoid the side effects of a pure or impure programming language.

In the second part, the author demonstrates that monads enhance modularity by presenting several variations of an interpreter for lambda calculus. The author first gave examples of monads and the methods for defining a monad. Philip talks about variations and new characteristics introduced for the interpreter with monads, including the standard interpreter, error messages, error messages with positions, states, output, non-deterministic choice, backward state, call-by-name interpreter and monad laws.

In the last part of the article, the author compared the monadic style produced by section 2 with continuation-passing style (CPS). The author introduced the differences between the CPS interpreter and the monads and the illustrated the method to transform them into the other's style. The author also raised some questions for the future based on his research in the article.

My Personal Thoughts:

This paper gave a comprehensive explanation of using monads to structure functional programs. Haskell is popular functional programming language and monads are well realized in this language. We can use many monads to create complicated functions by assemble them together. I think the concepts of monads are very abstract, however, learning something about it will improve my understanding of functional programming and lambda expressions. This concept is also applied in JavaScript and some other language features and I think I can take advantage of this chance to read more materials about monads.