TYPE INFERENCING FOR FUNCTIONAL PROGRAMMING LANGUAGES

EMILY ASHWORTH, TANYA BOUMAN, TONYE FIBERESIMA
ASHWOREL, BOUMANTE, FIBERET

1. High Level Description

This project is about compilers and compiler construction for functional programming languages, written in a functional programming language.

2. Area of Focus

In particular, we are focusing on type inferencing for a toy language. This toy language features Haskell-like syntax, but has a simpler type system, featuring only Ints, Bools, Strings and Functions.

3. Implementation

Similar to Martin Grabmüller's paper [1], we would like to explain type inferencing with an example algorithm. Since Grabmüller uses Algorithm W to explain step-by-step, we would like to use Algorithm M, and perform a similar functionality. In addition to the content explained in his paper, we also need to explain the background on type inferencing and would include some of the formal specifications of the Hindley-Milner type inferencing algorithms.

4. Relevance to the Field

This subject is important because while in many languages, such as C and Java, all types must be explicitly declared, this is not the case for all languages. In many functional languages, such as Haskell, it is not necessary to specify all types, since the compiler can infer them. A common system to accomplish this is the Hindley-Milner type inference [2], which has two main algorithms, Algorithm W and Algorithm M [3].

References

[1] M. Grabmüller, "Algorithm w step by step,"

[2] L. Damas and R. Milner, "Principal type-schemes for functional programs," in *Proceedings of the 9th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '82, (New York, NY, USA), pp. 207–212, ACM, 1982.

[3] O. Lee and K. Yi, "Proofs about a folklore let-polymorphic type inference algorithm," *ACM Trans. Program. Lang. Syst.*, vol. 20, pp. 707–723, July 1998.