2025

CS 4402

Comparative Programming Languages

LEARNING JOURNAL 2

SANA UR REHMAN

INSTRUCTOR: JIM CASALE

INTRODUCTION

This week focused on the fundamental elements of programming languages. I explored syntax, semantics, grammars, and parsing, while also examining compilation steps and how virtual machines work. My activities included completing the discussion post on Wirth's compilation model, studying the provided readings, and taking self-quizzes to test my understanding.

ACTIVITIES PERFORMED

I began by reviewing syntax and semantics to understand how programming languages are structured and interpreted. I studied Backus–Naur Form (BNF) and Extended Backus–Naur Form (EBNF) to see how grammars define valid language constructs. My main task was writing a discussion post on the compilation process based on Niklaus Wirth's six phases: lexical analysis, syntax analysis, semantic analysis, intermediate code generation, optimization, and code generation (Wirth, 2005). I highlighted the role of Context-Free Grammar (CFG) in ensuring proper parsing and syntax checking (Aho et al., 2007). I also explored the differences between compilers and interpreters and how virtual machines like Java Virtual Machine execute code across platforms.

REACTIONS AND FEELINGS

At first, abstract concepts of grammars and parsing made no sense, but breaking them up into steps helped visualize how compilers translate human-readable code into machine code. It was relieving to know that I had described Wirth's process of compilation properly after attempting. It was pleasant to connect CFG to real-world language design, and it made the built-in logic of programming languages even more understandable to me.

FEEDBACK AND INTERACTIONS

My peers indicated that my discussion post explained the compilation stages well and made it easy to understand CFG's position. The instructor noted that the examples of the lexical tokens and the syntax trees helped clarify. This confirmation made me certain that I can explain technical topics well, motivating me to remain at the same level of explanation in future assignments.

CHALLENGES AND SURPRISES

One of the challenging aspects was thoroughly grasping where type checking fits into semantic analysis. It required a revision of the symbol tables idea and how compilers handle variables and their scopes (Aho et al., 2007). I was surprised at how optional optimization can so profoundly affect performance, especially in computationally heavy programs.

SKILLS AND KNOWLEDGE GAINED

I crystallized my understanding of programming language design, e.g., syntax-semantics relation and parsing process. I also understood more clearly the difference between compilers and interpreters and how virtual machines provide platform independence. Writing the discussion post improved my ability to explain technical steps in an orderly fashion.

PERSONAL INSIGHTS

I realized that I learn best by connecting theory to examples from real life. Through the comparison of grammar rules with common programming principles, I was able to comprehend abstract ideas more readily. I also enjoyed how careful planning was necessary while explaining multi-step procedures such as compilation. These lessons will allow me to tackle future topics such as type systems and runtime environments.

CONCLUSION

The assignment for this week gave me a clearer picture of programming language parts, from syntax and grammar to compilation and virtual machines. I can now describe how compilers transform code, the role played by CFG towards parsing, and the difference between compilation and interpretation. These skills will facilitate my future growth in computer science and my ability to reason for language design.

REFERENCES

Aho, A. V., Lam, M. S., Sethi, R., & Ullman, J. D. (2007). *Compilers: Principles, techniques, and tools* (2nd ed.). Addison-Wesley.

Wirth, N. (2005). *Compiler construction*. ETH Zürich. https://people.inf.ethz.ch/wirth/CompilerConstruction/CompilerConstruction1.pdf

Wordcount: 547