Research Assessment #12

Date: 28 February, 2020

Subject: Syntax

APA citation:

Turbak , F., Gifford , D., & Sheldon, M. S. A. (n.d.). Design Concepts in Programming Languages. Retrieved February 21, 2020, from https://www.e-reading.club/bookreader.php/134602/Gifford,_Turbak_-_Design_Concepts_in_Programming_Languages.pdf

Assessment:

This week, I further researched the design of a Syntax of a programming language. Through my research last week I got to learn about the 3 principal elements of language design - Syntax (the form of the language), Semantics (the meaning of the language), and Pragmatics (the implementation of the language). Amongst these, the pragmatics of a language is not essential for me as my language would function as a husk over a pre-existing language. This means that I don't need to handle the inner level optimisations and interpretations, which exist to give a language its contextual meaning, such as being low-level and allowing editing of very specific details in files or being optimised to run very fast due to its processing algorithm. Amongst the remaining two, Syntax deals more with how the user interacts with language and writes programs.

First, I read about Abstract Syntax, the viewing of syntax from a more universal and theoretical manner. This lens or perspective involves recognising different aspects of syntax and how they may interact with each other. The abstract syntax behind a language is important as a review of the process of coding. This determines the interactions between elements and the ways and degree to which the programmer can control these interactions.

I also read about Abstract Syntax Trees, tree structures to represent the categorisation, interaction and relationships between the individual elements and keywords of a code sample. Although such structures can be implemented in the interpretation algorithm to make them a more practical than theoretical aspect of the language, for now, I will limit their use to the brainstorming aspect of designing my syntax. By viewing my brainstormed ideas from such a lens, I enable myself to make them more efficient also make myself comprehend the syntax on a deeper level, thus making it easier for me to code the interpretation.

Next, I read about Concrete Syntax, the practical aspects of a language, such as operation order and keywords. The concrete aspect of the syntax deals more directly with readability and ease of use. This, however, does not impact the interpretation of the code drastically. This is the part that I will focus on the most since the readability of the code is my primary focus in the design of this programming language. I decided to go with infix operation order, so the operand always goes between operations. This is the most natural order of precedence since most sentences in English have subject then verb then object.

My research from the week also allowed me to focus on brainstorming potential program structure for my language. I had to think over the efficiency of some keywords over others, the debate over more naturalised language versus more concise language, dynamic versus static types, etc. Based on my progress, I will continue working on the syntax, and with my research over semantics from next week, I will make changes as necessary according to new research, since any difficulty or ease in the interpretation of a language can impact how it is structured.