Brian Christian

Mrs. John

English 12

23 May 2019

Annotated Bibliography

Graham, Paul. "The Roots of Lisp." 18 Jan. 2002. *Paul Graham*, www.paulgraham.com/rootsoflisp.html. Working paper.

This paper provides a concise reexamination and explanation of John McCarthy's original Lisp paper based on on how Lisp has most commonly been used and restates some concepts in a more understandable manner. Author Paul Graham has copious amounts of computer science experience and has published two Lisp books, certifying that the information is accurate and thorough. This paper serves a similar use to McCarthy's paper while, by simplifying many concepts in the context of Lisp's modern use, reducing Lisp to its most important components and thereby simplifying the tasks of understanding and explaining it to general audiences.

McCarthy, John. "Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I." *Communications of the ACM*, vol. 3, no. 4, Apr. 1960, pp. 184-95.

This paper provides the original definition of the Lisp programming language and family. It defines the properties of the language that underlie every subsequent Lisp language, including their syntax, recursion-based models, higher-order functions, and reading and evaluation models. This source is arguably the most accurate source on the Lisp language and family because it provides the original definition of the language from its original author, John McCarthy. As an explanation of the Lisp language, this source is very useful for understanding it and its history, which aids in explaining Lisp, using a Lisp language, or designing an interpreter for one.

Mudawar, Muhamed. "Context-Free Grammar." *Compiler Design*, American U in Cairo, faculty.kfupm.edu.sa/COE/mudawar/csci447/slides/Grammar.pdf.

This presentation explains context-free grammars, the grammars that include most programming language syntaxes. The slides define these grammars and their written notation, show how text can be broken into components based on a grammar, and briefly overview other types of grammars. As a long-time computer science and compiler design professor at the American University in Cairo, Dr. Muhamed Mudawar is well-equipped to teach about the theoretical linguistics surrounding programming languages. The slides help to build up a comprehensive model of programming language design and parsing, which is useful for accurately explaining the topic in full to any audience.

Nystrom, Robert. *Crafting Interpreters*. craftinginterpreters.com.

This book teaches readers to implement a fully-featured language interpreter. In addition to explaining the entire process, it describes theoretical aspects of programming language design and compilation, including the phases of language compilers. Author Robert Nystrom has created six programming languages and works at Google on the Dart language, making him well-qualified to write a book about programming language design. This book is immensely useful for attaining a holistic view of programming language design and compilation, encompassing both theoretical and practical aspects. With this information, one can create a language interpreter and explain its functionality at various levels of complexity.

Pattis, Richard. "EBNF: A Notation to Describe Syntax." *Richard E. Pattis*, U of California, Irvine, www.ics.uci.edu/~pattis/misc/ebnf2.pdf.

This article explains Extended Backus-Naur Form, a notation for describing programming language syntaxes. Core topics include EBNF's history and fundamentals, methodologies for determining a symbol's legality according to a grammar, the distinction between syntax and semantics, EBNF's visual representations, and EBNF's support for languages with arbitrarily-deep nesting. Author Richard Pattis' experience with language design and as a computer science professor ensures the article's reliability. The article is useful for understanding the relationship between linguistics and programming language design by illustrating grammar definitions. Comparing processes related to EBNF grammars to human language processing methodologies clarifies this relationship for general audiences.

Vigo, Eugenio M. "On The Need of A Linguistic Theory of Programming Languages." Pompeu Fabra U, June 2015. *ResearchGate*, www.researchgate.net/publication/283016463_On_the_need_of_a_linguistic_theory_on_programming_languages. Typescript.

This paper draws parallels between the characteristics of programming languages and natural languages. Topics include the use of programming languages to communicate algorithms to humans and computers, their specialized syntaxes based on their use cases, their metalinguistic capabilities to extend a language within itself, and the question of language fluency. As a linguistics professor and researcher as well as a writer in the Free Software movement, author Dr. Eugenio Vigo's ideas are certifiably legitimate. The article is useful for understanding the similarities between programming languages and natural languages and assisting unfamiliar audiences with comprehending programming language structure, parsing, and analysis.