Statement of Interest

Anthony Hunt

May 13, 2025

Statement of Interest

Intuitive, simple, and powerful languages have fundamentally transformed our approach to programming and continue to define the future of software engineering. From 19th century mechanical calculators to high-level compilers, abstract computational interfaces have exponentially increased the expressiveness and efficiency of program execution. Within the vast field of computer science, my quest for ergonomic and performant programming workflows has paved the way for a Master's in programming languages.

Throughout my undergraduate degree, I was privileged to work with several McMaster professors on a variety of topics. My first-year research assistantship with Dr. Jacques Carette and Dr. Spencer Smith served as the initial spark for my interest in languages, where I was introduced to code generation techniques for several programming paradigms. At the end of the term, I presented an overview of software information encapsulation at McMaster's undergraduate poster event [4], solidifying my passion for research and technical communication.

Later on, a 16-month position at Intel presented several opportunities to experiment with the effects of usability-focused domain-specific languages. Under the tutelage of my co-op supervisor Dr. Hamid Hosseiny, I authored an internally published peer-reviewed paper on interface usability and parallelism in the context of test automation [5]. Based on our findings, I worked with the team to create a parallel-oriented DSL with a flow-based visual programming GUI, streamlining the process of creating and modifying automation tests.

Recent coursework has continued this trend towards programming language projects. My capstone project, for example, explored visual methods of programming language interaction, using diagrams to gain a deeper understanding of information flow. Simultaneously, a one-on-one directed readings course with Dr. Emil Sekerinski allowed me to explore recent literature on languages, modelling methods [1], and rewriting [2]. I spent several weeks researching formal mathematics in ten diverse languages, making note of strengths, weaknesses, and possible efficiency improvements. Then, I completed a survey on historical and recent rewriting techniques, from strategies [9] to equational saturation [8]. In the final weeks of my undergrad, I built a prototypical optimizing compiler for set-based operations.

Continuing work on the optimizing compiler in the summer months initiates the first phase of my Master's project: generating efficient implementations of abstract data types [6]. Oftentimes, formal specifications for real-world modelling only make use of boolean logic, number theory, and collection types foundational to discrete mathematics (as seen in the birthday book problem [7]). Sets, sequences, and relations may very well be powerful enough to accomplish a large subset of programming tasks that otherwise demand complex, concrete data types. Further, collection types come with a rich set of universally agreed-upon semantics [3] that allow extensive compilation time optimizations. Through my research, I seek to answer the following: To what extent can abstract data types be used to model real world phenomena? Then, using semantics as a foundation for optimization, are abstract data type operations more efficient than similar tasks in conventional languages?

McMaster's depth and breadth in formal methods, programming language theory, and compilation align closely with my goal of building a more usable interface for reasoning and communicating. By continuing my education, I plan to learn about type theory, compiler optimization, and influential approaches to modeling computation. Further, I am excited about advancing the core areas of modern languages, especially data structure optimization, formal computation techniques, and industry-standard code generation. Researching under this program and experimenting with compilation techniques will grant me the skills needed to back my ideals for future language development.

References

[1] Jean-Raymond Abrial. *Modeling in Event-B : system and software engineering*. eng. Cambridge: Cambridge University Press, 2010. ISBN: 9781139195881.

[2] Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998. DOI: 10.1017/CBO9781139172752.

[3] D. Gries and F.B. Schneider. *A Logical Approach to Discrete Math*. Monographs in Computer Science. Springer New York, 1993. DOI: 10.1007/978-1-4757-3837-7.

[4] Anthony Hunt, Jacques Carette, and Spencer Smith. *Information Encoding and Traceability in Software. Drasil - Generate All the Things!* Annual Undergraduate Student Research Poster Showcase at McMaster University. Aug. 2021.

[5] Anthony Hunt and Hamid Hosseiny. "MemAutoGUI: An Enhanced Automation Approach for Accelerated Validation Focused on Usability". In: *Design and Test Technology Conference*. Pages: 10. Intel, May 2024.

[6] Anthony Hunt and Emil Sekerinski. "Generating Implementations of Data Type Operations". Workshop on Synthesis (co-located with the International Conference on Computer Aided Verification). Pages: 3. May 2025.

[7] J. M. Spivey. "An introduction to Z and formal specifications". In: *Softw. Eng. J.* 4.1 (Jan. 1989), pp. 40–50. DOI: 10.1049/sej.1989.0006.

[8] Ross Tate, Michael Stepp, Zachary Tatlock, and Sorin Lerner. "Equality saturation: a new approach to optimization". In: *Proceedings of the 36th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL '09. Savannah, GA, USA: Association for Computing Machinery, 2009, pp. 264–276. DOI: 10.1145/1480881.1480915.

[9] Eelco Visser, Zine-el-Abidine Benaissa, and Andrew Tolmach. "Building program optimizers with rewriting strategies". In: *SIGPLAN Not.* 34.1 (1998), pp. 13–26. DOI: 10.1145/291251.289425.