Statement of Purpose

Chenhui Yuan

November 12, 2018

A famous diagram, in P. H. Winston’s 1977 volume on artificial intelligence, boldly states that AI can be achieved through merely eight steps. From below, transistors, gates, registers, and instructions engineer silicon into machinery. From above, intelligent programs sit atop a pattern matcher, emulating human reasoning. The middle steps sit indisputably within computer science: the language Lisp, and its compiler and interpreter, bridge between human and machine. In the realm of languages, of Lisp, ML, and Algol, innovation has shaped computer science as it became ubiquitous. It is in this field of programming languages that I wish to pursue my postgraduate studies, applying the theory of PL to the design and implementation of robust software systems.

Research

My current research interest is Bayesian inference on probabilistic programming languages (PPLs), an example of the interaction between PL and current challenges in other fields. Such languages allow programmers to leverage nondeterministic constructs, compactly expressing infinitely many possible outcomes. They are powerful enough to model problems in fields like machine learning, statistics, and dynamical systems. For example, popular graphical models like Bayesian networks may be directly embedded into PPLs.

Inference, the prediction of the output distributions of probabilistic programs, solves important questions like conditional probabilities on significant states or the expected convergence time of an algorithm. Rapid inference is valuable for practical experimentation and deployment of PPLs, but it is also difficult since the state space is exponential in the number of variables. Dynamic techniques for inference generally rely on sampling which is slow and unsound. By contrast, we consider static analyses that are exact and efficient, and guarantee soundness. For example, Claret et al. use dataflow analysis to achieve exact inference, and Batz et al. use program analysis to compute sampling time, demonstrating the possibility of fast and trustworthy approaches.

To that effect, under the tutelage of Prof. Jan Hoffmann and his student Di Wang, I built an inference system for PPLs with Bernoulli variables and loops, via a compiler into linear constraint systems. We interpret each statement as a stochastic matrix on program states, and transformations combine to inductively yield meaning for subprograms. Loops present a challenge due to self-reference in the transformations, which we treat as a matrix system converted into a linear program. Our result enables efficient inference on expressive programs. Experiments using real-world Bayesian networks indicate competitiveness with existing approaches for inference, with greater generality than commercially available solvers.
This year, I am continuing the work as part of a senior research thesis. We are developing a theoretical framework to prove soundness for the analysis through denotational semantics that provide a mathematical interpretation of each syntactical construct, related to the matrix formulation. On the implementation side, we are integrating novel dataflow analysis to reduce the complexity of inference in the presence of new language features. We diminish the sizes of the intermediate systems by restricting subprogram analyses to only relevant states. An important challenge is preserving non-termination behavior, which is achieved by least-fixed-point reasoning. The relevance of states is formally defined as dataflow equations and implemented using a sparse data structure for transformations. Since many programs are decomposable into independent substructures, we may thus accelerate inference and consume less memory.

Academic Interests

The opportunity to conduct research in cutting-edge computer science compels me to pursue postgraduate studies. I entered undergrad at Carnegie Mellon having been previously exposed to research in computational biology, leading me to explore the productivity of computational tools. My undergraduate career was marked by a dual fascination for the theory of programming languages and the implementation of computer systems. On the former front, I have been a teaching assistant for our PL course for two semesters, covering induction, abstraction, and recursion within the typed lambda calculus, along with parallelism and concurrency. As a course project, I implemented a type-directed compiler for Standard ML, and I also participated in a graduate seminar on computational higher type theory. Classes, research, and the opportunities to attend summer schools and conference workshops, notably PLMW at PLDI’18, all bolstered my academic experience in programming languages.

At the same time, I gained substantial experience in systems implementation, including a multithreaded Unix-like operating system kernel, an optimizing compiler for a C dialect, a bidirectional ray-tracing renderer, and a GPU-accelerated video processing pipeline. I am particularly interested in the interaction between languages and systems, and how they enable efficient and robust software construction. Many topics at the forefront of computer science research lie in the intersection—modern memory-safe systems programming as in Rust, advanced garbage collectors for parallel programs, compilation of domain-specific languages into efficient native code, and countless more. I am motivated by the vast potential in the collaboration of these two fields, as advances in language design and type systems will enable programmers to leverage hardware and embrace functional, parallel, and concurrent paradigms for building large-scale applications.

Finally, I have a passion for teaching. I have been a TA at CMU continuously since freshman year, and in addition to the PL theory course, I served as a head TA for our course on parallel algorithms. This course, part of the CS core sequence and taken by more than two hundred students each semester, gave me the opportunity to lead a sizable teaching staff, contribute to assignment, exam, and recitation material, and directly interact with students in large lecture settings. Also, through CMU’s Student College program, I helped develop a fully student-taught course with more than thirty enrollments—“Hype for Types”, a class on topics such as dependent types, pure type systems, and category theory. Teaching has been immensely fruitful for me personally, im-
proving my communication and leadership skills while introducing me to the academic world. I hope to continue having the privilege of leaving an impact on students and working alongside professors in graduate school. In all, attending a Ph.D. program will enable me to make my own contribution to computer science and to continue my academic career as a professor. Being able to share knowledge with others and make contributions to computer science is the most rewarding career path I can picture.

At CMU, I would be honored to continue developing my understanding of the theory of programming languages, while participating in world-class research and exploring the interplay between PL and systems. My interests including verification, static analysis, and probabilistic programming are well-represented at CMU, such as by Prof. Hoffmann, while Prof. Harper leads in type systems, semantics, and logic. I am also interested in cross-field opportunities, for example working with Profs. Blelloch, Acar, and others on practical parallel computing, as well as the excellent systems faculty on the implementation of these ideas. The quality of my undergraduate education here convinces me that CMU will remain at the vanguard of computer science.