Research Statement

Ben Little

Sep 20, 2023

Humanity has long had the desire to build some grand machination that will save us from our troubles. In recent times this desire is seemingly manifest in the form of global scale computer systems. These systems are so vast and complex that it easy to allow ourselves to believe that they are actually magic. All we must do is hand over our data and do what the system tells us; surely this will lead us to salvation! But one truth seems to cut through this zealotry: that the solutions to human problems necessarily originate in human minds.

If computers are to play a part in these solutions, their operation must be clear and correct. We cannot afford to muddy our perception and risk fooling ourselves back into a state of magical thinking. Verifying the correctness of the physical systems that we use to automate our reasoning is a monumental endeavor, but it is the only path forward. There are yet many who would prefer the magical thinking. It is easier and more comfortable to put our faith behind unfathomable complexity than it is to work in the mud, building the foundations of truly verified systems.

There are existing tools for verifying the correctness of software and hardware. Tools like Coq, Agda, and Lean, allow users to both specify the *meaning* of their programs and prove that an implementation satisfies this specification. Each of them has failed to garner a wide audience, however. Researchers, scholars, and a few brave and highly skilled software engineers, remain their primary users. If we are to break free of our dependence on unverified systems, we must find a way to make formal verification accessible.

The accessiblity of a programming paradigm is most visible in its presentation. Programming languages based on human language are sequential, while the structure of most mathematical and scientific problems are highly parallel. One tool that has made parallelism more accessible is dataflow programming and the tightly-related concept of Visual Programming Languages (VPLs). VPLs present programs as dataflow graphs. One large computational process as the composition of smaller computational processes. Processes can begin as soon as all of their dependencies are ready, making parrallelism automatic in the sense that independent computations can be performed simultaneously.

Proofs are similar in this regard: One proof can be verified as soon as all of

its dependencies are verified. If VPLs can express the parrallism of dataflow programming languages, this leads to the natural question, "Is there a useful and concise means to visually represent proofs?" Furthermore, would this Visual *Proving* Language be an effective means to make formally verified computation more accessible?

My research is to better understand these questions. Dependent type theory is the mathematical basis on which proof-capable programming languages are built. A dependently-type VPL is my first research goal. To reach this goal, I plan to use the existing body of research in the field of Applied Category Theory. String diagrams, a form a visual reasoning for category theory, are an inroads to designing a dependently-type VPL for many processes of human interest, including both classical and quantum computation, computer hardware, and non-traditional computing media like biological systems. It is my hope that this line of research will yield an artifact that is both mathematically beautiful and practically useful in the effort to include more minds in the quest to solve humanity's biggest problems.