End-to-End Compiler Verification

Motivation
- Many life-critical systems rely on the correct operation of software.
- Examples include autonomous vehicles, avionics systems, medical systems such as robotic surgery machines, nuclear reactor control systems, etc.
- End-to-End Compiler Verification ensures that the software executable generated by the compiler is a correct translation of the source code.

Compiler Process
- A compiler first translates the source program into another language called the intermediate representation (LLVM code).
- Next it optimizes the LLVM code.
- Finally it translates the LLVM code into the target language (the executable file).

Isabelle
- We model our LLVM code with the Isabelle Proof Assistant.
- Proof assistants lend the capabilities of computers to solving proofs.
- The only feasible way to write proofs about complex systems is to utilize automated theorem provers.

Proving De Morgan’s Laws in Isabelle

Select Instruction

Future Work
The goal of the VeriF-OPT project is to develop a framework of tools that compiler designers can use to verify their work.

Future additions to the project include extending the operational semantics of the LLVM model to include more instructions such as ‘getelementptr’.

Optimizations of concurrent programs can also be verified by the addition of a memory model.

Acknowledgements
I would like to give a huge thanks to my mentor, William Mansky, for all his instruction and guidance throughout this project.

Many thanks to the PURE committee for organizing this unique opportunity to become involved with the fascinating research being conducted on campus.

And thanks to Rockwell Collins for making this opportunity possible.

If you want to learn more about the VeriF-OPT project, you can visit their website at [VeriF-OPT](https://www.example.com/verif-opt).

University of Illinois at Urbana-Champaign

Promoting Undergraduate Research in Engineering