Youngjae Moon

Professor Johnson

CS 6315 Automated Verification

30 April, 2023

Project Final Report

For CS 6315 final project, I decided to explore the correctness of Exocompiler. To do this, I have first used Exocompiler to compile Exo functions embedded in Python into C (most of the functions from the GitHub repository of Exo language). Next, I have encoded the original Python and transformed C functions in Python. Then, I have encoded them again in SMT. Finally, I have used an SMT solver to prove that the original and transformed programs are equivalent.

Here is the GitHub repository of my final project: https://github.com/Pingumaniac/Verification_of_Exocompilation. Note that I have made separate folders for different Exo functions that are embedded in Python, and have been compiled into C, and then checked for equivalence of original and transformed programs using PySMT.

There are four main ways to improve the project in the future. First, I can represent Python and C functions directly in SMT instead of first representing them in Python and then SMT. have initially tried this using PySMT expressions such as ForAll, Implies, And, Equals, Plus, Times, etc. For instance, ForAll and Implies can be used to represent the for loop structure in Python instead. Nonetheless, this gave me endless errors for debugging. Hence I have stopped approaching in this way and changed the approach.

Second, I can perhaps write more complex Exo functions for this project. I have noted that the Exo language lacks documentation, and hence when I have tried out implementing a simple ReLU function and an artificial neural network, they weren't able to be compiled. All Exo functions except conv2d (convolution for 2D) and gaussian blur in this file have been obtained from analyzing the GitHub repository of the Exo language and extracting specific functions mostly from their testing files, other remaining files, and from their web page.

Third, I can use PySMT to help in determining the optimal set of transformations and optimizations for the Exocompiler to generate optimised code for the target hardware accelerators.

Finally, I can use PySMT to synthesize code fragments that meet specific requirements: e.g., hardware compatibility or performance constraints. This can then be integrated into the generated low-level code.

To publish a paper about formal verification of Exo language, I should be able to expand the project based on the third and fourth idea. But I do not have the ability to do this yet.

On a side note, I am not sure about the commercial value of Exo, because I think Python will one day die, as Python is made by adding a transpiler on top of C. I think it is better to create another embedded DSL on top of Rust or Zig for (1) optimizing numerical programs, (2) targeting uncommon accelerator hardware or even developing your own, and (3) getting as close as possible to the physical limits of the target hardware.

My current plan over this summer break is to increase my skills in programming language. I will first complete all the exercises in Scheme from "Structure and Interpretation of Computer Programs" by Gerald Jay Sussman, Hal Abelson, and Julie Sussman. Then, I will implement a functional programming language based on "Functional Programming Application and Implementation" by Peter Henderson.

Further Notes

1. About Lisp and recursion in functional programming languages

I think Lisp does not have problems in terms of stack overflow when using recursion. From my limited knowledge Lisp compiler uses tail call optimization. This allows the Lisp compiler to not create new stack frames when calling a function recursively in a tail position. In other words, the current stack frame is reused instead of creating new stack frames. This, in turn, reduces the amount of memory needed and thus reduces the risk of stack overflow.

Thus, the Lisp compiler automatically allocate and deallocate memory as needed during program execution. This makes it possible to allocate large data structures and to use recursion without worrying about running out of memory.

I remember from taking an "Artificial Intelligence" course while undergrad that I was asked to code all the assignments in Lisp. Hence I complained to the associate Head of Computer Science department for my undergrad about this and why not ask in e.g. Python (or other language) instead. Thus, I asked how functional programming language may not be good due to the problem about recursion. I remember how he told me that how Lisp compiler is well-designed to overcome this problem. I have not studied deeply about this yet.

2. WebAssembly and Game Engines

I think we can create various new programming language in which AI will write codes from now-on, thus as new domains will come out intermittently, we can perhaps create domain specific languages and/or programming languages based on this.

I was thinking about creating a programming language on top of WebAssembly Text Format (WAT), which somewhat looks like a Racket. Perhaps we can also have something similar like WAT for other assembly languages such as RISC-V.

There is a new programming language recently released from Epic Games: https://dev.epicgames.com/documentation/en-us/uefn/verse-language-reference. This new language is based on functional programming paradigm, and I will have a look later once there are more adequate materials.

I don't think Exo DSL is good b/c it is based on Python. I think C++ will be replaced by Rust except for developing games using Unreal (unless this will be also replaced which is one of my future goals), Python will be replaced by other programming language (as it is too slow and it transpiles to C, perhaps we can create a new transpiler on top of Rust to create a Python-like language for Rust), JavaScript and TypeScript will be replaced by new programming language based on WASM, most JVM based language will be replaced with something else (e.g. Golang, Rust) due to their slow performance and thus things are now being converged to Web and there is less need to create VM based programming language such as Java and C#.

I think ReactJS will die as well, considering how Meta is a declining company. Children tend to not use SNS that their parents do mainly as it is a human nature for them to not be overlooked by their parents, and hence I think new SNS will come out intermittently. Facebook started to decline, Instagram is slowly declining, and I am sure Tik-Tok will one day decline as well.

I am thinking about creating a new programming language for game engines and cloud games. I think coding for computer graphics can be mostly done by AI models in the future, and the key skill for developing game engines will lie mostly in the understanding of programming language design, and compiler design and optimization.

The problem with creating a game engine based on WASM is that I think it will become hard for using GPUs and NPUs for heavy computation. I think Rust's ownership model is not good for creating game engines.

3. Further notes about AI weakness

https://goattack.far.ai/pdfs/go_attack_paper.pdf

https://arxiv.org/abs/2211.00241

In addition to energy efficiency, I think AI programs have limitations in abstract and logical thinking skills, as machine learning algorithms are based on mathematics, probability, and statistics. This is evident by how the Chinese AI program – KataGo lost against human amateur (Chinese) Go player (though the tactic that the human player used is developed by a machine learning program as well).

https://www.youtube.com/watch?v=iyLwdQg_wDI

One of Korean pro Go players also used this tactic to try against KataGo afterwards and has won several games afterwards. The Korean Go player shown in the YouTube video above, has more detailed analysis about the weakness of AI programs for Go in his other videos. On a side note, I always watch all his videos regarding AI programs for Go since my undergrad. One of the critical points that he said about the weakness of AI programs for Go is that since they are based on probabilistic algorithms, they show off critical weakness when they must do extremely deep logical thinking. Thus, he pointed out that AI programs tend to not know any deep concepts about Go that humans have when playing Go, and hence make silly mistakes that no human pro player will play.

On a side note, there are some other points that he made as well but they require some knowledge about Go and thus it is very hard for me to translate. Thus, some of the points that he mentioned have been resolved now as AI programs got better over time.

Nonetheless, I know that there are separate AI programs that are good in mathematics. This, there was a time where people assumed that AI programs could not do art and design well, but this have been proven to be wrong. So, I should still be careful.

4. Database Management Systems (DBMS)

I think new DBMS will come out more frequently, as new domains will keep coming out. Thus, I think things like ORM, ODM, OGM will become anti-patterns as I am sure AI models can write complex queries effectively (like your point about software verification).

I wonder how it would be like if I create a new multi-modal DBMS (so that it is complex enough) that stores data e.g., in all relational, document, and graph way. And we can let AI to write complex queries for handling this complex DBMS. How useful will this be? Perhaps some of the new domains created in the future, will benefit from using multi-modal DBMS.

5. About Computer Architecture Industry

I think the field of computer architecture will slowly decline as well (though there are ups and downs in this field periodically) for two main reasons. First, too many semi-conductor factories are being built. For instance, TSMC and Samsung are building a lot of factories not only in Taiwan and South Korea, but globally such as America, Europe, and Japan. In addition, the Chinese government wants to have its own supply chain for semi-conductors and hence a lot of semi-conductor factories are being built by Chinese companies at China as well. Once they all get constructed and start to get operate, I think the price of semiconductor will drop significantly.

Second, since there are so many people working in this field, it is attractive for people in AI disciplines to try out automating them. If large people work on single task, there will likely be more data to train AI models to automate. Moreover, it is a good target for AI models to automate larger number of people as then they can generate higher profit compared to automating smaller number of people.

Thus, semiconductor business seems in general to generate high profits, there are critical problems for businessmen to do business in this field. First, the total amount of investment needed is huge, especially for R&D and building and modifying factories. Hence in terms of return of rate of investment (ROI), it is not high enough and hence it is hard for semiconductor companies to give extremely high salaries to its workers.

Second, new semiconductors always come out and whenever new semiconductors come out, the value of existing semiconductors depreciates significantly. Hence these companies must sell their semiconductors as soon as possible before new semiconductors come out.

Third, although it is possible for semiconductor companies to reduce its amount of production, it cannot stop its production after factories have been built. Because it costed too much money for them to build their factories and the only way to get the return of investment is to sell their semiconductor produced from their factories.

Fourth, it is hard to predict which computer architecture technology will come out and dominate as time passes. Hence the entry barrier for this business seems very high for most people, but it isn't as if new innovative technology comes out and commercialized the existing production lines become useless.

Finally, in terms of cash flow perspective, although it seems that semiconductor companies generate high sales and profits in general, they must periodically spend extremely large amount of money for R&D and building new factories. Otherwise, they will fall behind and lose in the competition. While this can be good from a country economy perspective, it is not good for semiconductor companies to figure out whether they are making profits or losing their money in the long run.

For all these reasons, I do not think diving into computer architecture field is good. I think this field will slowly decline over time in the long run. Nonetheless, I do not mean that concepts from computer architecture are meaningless at all. I guess they should be learnt and rather applied for energy efficient computing as AI programs consume too much electricity. Thus, I am not an EE major and hence I cannot be very good enough at this field in the first place.

6. About PhD admission

I will not apply for Yale PhD in CS. Thus, I will not do a PhD in Software Verification. Even without your thoughts about AI models going to replace Software Verification Engineers' jobs, I think the software verification part is too narrow but deep and therefore will not be good for me. It will require me to heavily study in this field if I go for a PhD here, but once I graduate, it will be difficult for me to found or co-found a company or at least be an excellent generalist.

I will try to stick with Professor Yu Huang for Ph.D. But I think the probability for this is about 1/3. Therefore, I should have to apply various Ph.D. programs in the USA and perhaps some top Canadian universities as well if I could not do a Ph.D. under her guidance. For Ph.D., unless I do not get an internal or external fellowships or be a teaching assistant, I should work as a research assistant. Although there is no guarantee she will have obtained new research funds to support me from January 2025, I will do my best to first acquire enough skills to do research about Software Engineering.

I do not want to go to California, New York and perhaps Seattle as well. I think they will decline considering how generating AI models will replace a bunch of white-collar jobs. This is contrasting to Texas and Nashville where many companies are moving their headquarters to here and thus the population will grow.

I would like to end this note by pointing out how thankful and blessed I am for me to hear your thoughts about various matters.