WWTF Grant Proposal
Leveraging Resolution for Verified Program Synthesis

March 21, 2022

1 Project summary

Program synthesis is a long-standing problem crossing many areas of computer science and mathematics including programming languages, artificial intelligence, formal methods and proof theory. There have been major advances in NLP (natural language processing) based program synthesis recently, most prominently DeeepMind's AlphaCode [11], OpenAI's Codex [2] and Github's Copilot [4].

However a major drawback of such an approach is that there is little control over the generated code leaving us with no guarantees for reliability or security of the resulting programs.

Another route to program synthesis is offered through the Curry-Howard correspondence. Exploiting the equivalence of proofs in intuitionistic logic and functional programs allows us to reduce the problem of synthesising a program satisfying a specification into proving its validity. The added benefit of this approach is that a program generated from such a proof is guaranteed to be correct. From the perspective of formal methods there are benefits as well. At the moment creating a verified program requires both writing specification and code. Being able to automate the latter step would save considerable overhead.

However automated theorem proving in intuitionistic logic is notoriously difficult in particular due to the absence of convenient normal forms. Most state of the art theorem provers like Vampire [10] and E [13] use resolution which is a great technique for automation but inherently non-intuitionistic in that it requires the formulas to be transformed into conjunctive normal form and works through refutation of the negated formula, which is not intuitionistically sufficient for a proof of validity.

The goal of this project is to establish when and to what degree theorem provers based on classical logic can still be utilized, allowing us to leverage the simplicity of resolution to synthesize programs, and to then implement a functionality to this end on top of the Vampire theorem prover.

2 Introduction, background and state of the art

Program synthesis in the sense of this project is concerned with generating a program that satisfies a specification given in FOL (first order logic). Instead of directly generating a program however we obtain it via the Curry-Howard correspondence from a proof of the validity of the specification. Such a program will always be valid.

Cutting edge theorem provers like Vampire [10] and E [13] have advanced to a point where they can already fully automatically prove many specifications [1] and are ready to be used in practical systems, in particular when provided with hints. With recent development to inductive reasoning [8][7] we can expect that more and more specifications will be provable in the future.

However for the Curry-Howard correspondence to apply the initial proof has to be valid in intuitionistic logic but state-of-the-art theorem provers utilize resolution which is an inherently classical techniques. There are 2 approaches present in literature that can help to remedy this fact.

On the one hand the Curry-Howard correspondence can be extended to classical logic using control operators [5], Parigot's $\lambda\mu$-Calculus [12] or CPS-translation [6]. However in this process we must in general give up on some desirable properties, at the very least realizability of $\vee$ and $\exists$, i.e. a proof of $\exists x P(x)$ will not necessarily give us an x such that $P(x)$ and a proof of $A \vee B$ might give us neither a proof of A nor a proof of B.

On the other hand in certain regimes it is possible to transform a classical proof into an intuitionistic one, e.g. proofs of Π_2 sentences in Peano Arithmetic [3], via Friedman's Translation or similar methods. [9] This approach has been examined in different small-scale settings and would have to be adapted for a general setting.

Finally we suggest a third approach which has not been covered in literature yet, that is to pass a modified formula to the classical prover which can give us additional information about what an intuitionistic proof can look like.

By combining the above approaches we hope to examine to what extent we can extract intuitionistic information from a specification using a classical theorem prover and implement a real-world system for this on top of the vampire theorem prover.

3 Research questions, objectives and hypotheses

The final goal of our project is to leverage state-of-the-art first-order automated theorem provers, in particular Vampire, for program synthesis. This is motivated by the recent and prospective advancements in their capabilities.

Before beginning an implementation a lot of theoretical groundwork will have to be laid, this will be (WP1). An optimal procedure for synthesis would consist of

- transforming the specification into an alternate form that would guarantee the use of as few non-intuitionistic rules as possible and applicability of Friedman translation in as many places as possible
- applying the prover to this modified specification obtaining a classical proof
- apply Friedman translation or similar translation procedures in as many places as possible
- extracting a program via Curry-Howard correspondence, applying control operators to non-intuitionistic remnants as necessary.

All of these elements except the initial modification of the specification already exist separately and in model settings such as arithmetic. Extending these results and combining them will be the first major undertaking of this project. This is also a necessary precondition for determining what initial modifications should be undertaken. We expect that for most real-world synthesis problems our procedure will be sufficient to transform classical proofs into usable and verified programs.

Finally we want to implement a real system on top of Vampire realizing the synthesis problem (WP2) which will target a suitable functional programming language.

4 Expected results, novelty, and relevance

As outlined we expect to obtain the necessary theoretical foundations as well as a practical system to leverage state-of-the-art classical theorem provers for program extraction. At the current time no such system exists. Much of the theoretical groundwork for implementing such a system is already present, but is scattered and mostly applied in theoretical model settings.

A successful completion of this project would lay the foundation for verified program synthesis via automated theorem provers.

5 Methods and feasibility

6 Potential to span across disciplines and to build bridges to application fields

References

[1] The CADE ATP System Competition. http://www.tptp.org/CASC/.

[2] Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374* (2021).

[3] Friedman, H. Classically and intuitionistically provably recursive functions. In *Higher Set Theory* (Berlin, Heidelberg, 1978), G. H. Müller and D. S. Scott, Eds., Springer Berlin Heidelberg, pp. 21–27.

[4] GitHub. Your AI pair programmer, 2021. https://copilot.github.com/ accessed on 20.03.2022.

[5] Griffin, T. G. A formulae-as-type notion of control. In *Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages* (1989), pp. 47–58.

[6] Groote, P. d. A cps-translation of the $\lambda\mu$-calculus. In *Colloquium on Trees in Algebra and Programming* (1994), Springer, pp. 85–99.

[7] Hajdú, M., Hozzová, P., Kovács, L., Schoisswohl, J., and Voronkov, A. Induction with generalization in superposition reasoning. In *International Conference on Intelligent Computer Mathematics* (2020), Springer, pp. 123–137.

[8] Hozzová, P., Kovács, L., and Voronkov, A. Integer induction in saturation. In *Automated Deduction – CADE 28* (Cham, 2021), A. Platzer and G. Sutcliffe, Eds., Springer International Publishing, pp. 361–377.

[9] Kohlenbach, U. The Friedman A-translation. *Applied Proof Theory: Proof Interpretations and Their Use in Mathematics* (2008), 273–277.

[10] Kovács, L., and Voronkov, A. First-Order Theorem Proving and Vampire. In *Computer Aided Verification*. Springer Berlin Heidelberg, 2013, pp. 1–35.

[11] Li, Y., Choi, D., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., Eccles, T., Keeling, J., Gimeno, F., Lago, A. D., et al. Competition-Level Code Generation with AlphaCode. *arXiv preprint arXiv:2203.07814* (2022).

[12] Parigot, M. $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In *International Conference on Logic for Programming Artificial Intelligence and Reasoning* (1992), Springer, pp. 190–201.

[13] Schulz, S. E–a brainiac theorem prover. *Ai Communications 15*, 2-3 (2002), 111–126.