Towards Evolutionary Theorem Proving for Isabelle/HOL

Yutaka Nagashima[*]
University of Innsbruck, Czech Technical University

ABSTRACT

Mechanized theorem proving is becoming the basis of reliable systems programming and rigorous mathematics. Despite decades of progress in proof automation, writing mechanized proofs still requires engineers' expertise and remains labor intensive. Recently, researchers have extracted heuristics of interactive proof development from existing large proof corpora using supervised learning. However, such existing proof corpora present only one way of proving conjectures, while there are often multiple equivalently effective ways to prove one conjecture. In this abstract, we identify challenges in discovering heuristics for automatic proof search and propose our novel approach to improve heuristics of automatic proof search in Isabelle/HOL using evolutionary computation.

CCS CONCEPTS

• **Software and its engineering → Search-based software engineering**; *Formal software verification*;

KEYWORDS

Theorem Proving, Isabelle/HOL, Genetic Algorithm

ACM Reference Format:
Yutaka Nagashima. 2019. Towards Evolutionary Theorem Proving for Isabelle/HOL. In *Genetic and Evolutionary Computation Conference Companion (GECCO '19 Companion), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3319619.3321921

1 BACKGROUND

1.1 Interactive Theorem Proving

Interactive theorem provers (ITPs) are forming the basis of reliable software engineering. Klein *et al.* proved the correctness of the seL4 micro-kernel using Isabelle/HOL [3]. Leroy developed a verified opimizing C compiler, CompCert, in Coq [6]. Kumar *et al.* built a verified compiler for a functional programming language, CakeML, in HOL4 [5]. In mathematics, mathematicians are substituting their pen-and-paper proofs with mechanized proofs to avoid human-errors in their proofs: Hales *et al.* mechanically proved the Kepler conjecture using HOL-light and Isabelle/HOL [2], whereas Gonthier *et al.* proved of the four colour theorem in Coq [1]. In theoretical computer science, Paulson proved Gödel's incompleteness theorems using Nominal Isabelle [11].

[*]Supported by the European Regional Development Fund under the project AI & Reasoning (reg. no.CZ.02.1.01/0.0/0.0/15_003/0000466)

1.2 Meta-Tool Approach for Proof Automation

To facilitate efficient proof developments in large scale verification projects, modern ITPs are equipped with many sub-tools, such as proof methods and tactics. For example, Isabelle/HOL comes with 160 proof methods defined in its standard library. These sub-tools provide useful automation for interactive proof development.

`PSL`. Nagashima *et al.* presented `PSL`, a proof strategy language [9], for Isabelle/HOL. `PSL` is a programmable, extensible, meta-tool based framework, which allows Isabelle users to encode abstract descriptions of how to attack proof obligations.

Given a `PSL` strategy and proof obligation, `PSL`'s runtime system first creates various versions of proof methods specified by the strategy, each of which tailored out for the proof obligation, and combine them both sequentially and non-deterministically, while exploring search space by applying these created proof methods.

The default strategy, `try_hard`, outperformed, `sledgehammer`, the state-of-the-art proof automation for Isabelle/HOL, by 16 percentage points when tested against 1,526 proof obligations for 300 seconds of timeout; However, the dependence on the fixed default strategy impairs `PSL`'s runtime system: `try_hard` sometimes produces proof methods that are, for human engineers, obviously inappropriate to the given proof obligations.

`PaMpeR`. Nagashima *et al.* developed `PaMpeR` [8], a proof method recommendation tool, trying to further automate proof development in Isabell/HOL. `PaMpeR` learns when to use which proof methods from human-written large proof corpora called the Archive of Formal Proofs (AFP)[4]. The AFP is an online journal that hosts various formalization projects and mechanized proof scripts. Currently, the AFP consists of 460 articles with 126,100 lemmas written by 303 authors in total.

`PaMpeR` first preprocess this data base: it applies 108 assertions to each (possibly intermediate) proof obligation appearing in the AFP and converts each of them into a vector of boolean values. This way, `PaMpeR` creates 425,334 data points, each of which is tagged with the name of proof method chosen by a human engineer to attack the obligation represented by the corresponding vector. Then, `PaMpeR` applies a multi-output regression tree construction algorithm to the database. This process builds a regression tree for each proof method. For instance, `PaMpeR` builds the following tree for the `induct` method:

```
(1, (10, expectation 0.0110944442872,
         expectation 0.00345987448177),
    (10, expectation 0.0510162518838,
         expectation 0.0102138733024))
```

where each of `1` and `10` in the first elements of the pairs represent the number of the corresponding assertion. For example, this tree tells that for proof obligations to which the assertion 1 returns false but the assertion 10 returns true, the chance of an experienced proof engineer using the `induct` method is about 5.1%.

When a user of PaMpeR seeks for a recommendation, PaMpeR transforms the proof obligation at hand into a vector of boolean values and looks up the trees and presents its recommendations.

PaMpeR's regression tree construction is based on a problem transformation method, which handles a multi-output problem as a set of independent single-output problems: For each obligation, PaMpeR attempts to provide multiple promising proof methods to attack the obligation, by computing how likely each proof method is useful to the obligation one by one.

PaMpeR *is not optimal to guide* PSL. One would imagine that it is natural step forward to improve PSL's default strategy by allowing PaMpeR to choose the most promising strategy for a given problem instead of always applying the fixed strategy, `try_hard`, naively.

Despite the positive results of cross-validation reported by Nagashima *et al.*, PaMpeR's recommendation is not necessarily optimal to guide an automatic meta-tool based proof search for two reasons. First, PaMpeR recommends only one step of proof method application, even though many proof methods, such as `induction`, can discharge proof obligations only when followed by appropriate proof methods, such as `auto`, which is a general purpose proof method in Isabelle/HOL. Second, when PaMpeR transforms a multi-output problem to a set of single-output problems, PaMpeR pre-process the database introducing a conservative estimate of the correct choice of proof methods. In the above example, PaMpeR's pre-processor produces the following data point for all databases corresponding to proof methods that are not `induction`.

```
not, [1,0,0,1,0,0,0,0,1,0,0,1,0,...]
```

We know that this conservative estimate wrongfully lowers the expectation for other proof methods for this case. For example, Isabelle/HOL has multiple proof methods for induction, such as `induct` and `induct_tac`. Experienced engineers know `induction` is a valid choice for most proof obligations where `induct` is used. Unfortunately, it is not computationally plausible to find out all alternative proofs for a proof obligation, since many proof methods return intermediate proof obligations that have to be discharged by other methods and even equivalently effective methods for the same obligation may return distinct intermediate proof obligations. In the above example, even though both `induct` and `induction` are the right choice for many proof obligations, they return slightly different intermediate proof goals for most of the cases, making it difficult to decide systematically if `induct` was also the right method where human engineers used `induction` method.

2 EVOLUTIONARY PROVER IN ISABELLE/HOL

We propose a novel approach based on evolutionary computation to overcome the aforementioned limitations of method recommendation based on supervised learning. Our objective is to discover heuristics to choose the most promising PSL strategy out of many hand written default strategies when applied to a given proof goal, so that PSL can exploit computational resources more effectively.

We represent programs as a sequence of floating point numbers, each of which corresponds to a combinations of results of applying assertions to a proof obligation. PaMpeR leaned 239 proof methods from the AFP and built a tree of height of two for each of them; Therefore, we represent a program as a sequence of floating number of length 956, which is the total number of leaf nodes in all regression trees. Then, we assign such sequence to each default proof strategy. Our prover first applies assertions to categorize a proof goal, then applies the most promising strategy for that goal.

As a training data set, we randomly picks up a set of proof obligations from large proof corpora. And we measure how many obligations in this data set each version of our prover can discharge given a fixed timeout for each obligation. The more proof goals in the data set a prover can discharge, the better the prover is.

After each iteration, we mutate the program, which is a mapping function from a combination of results of assertions to the likelihood of each strategy being promising to the corresponding proof obligations. After each evaluation, we select provers with higher success rates and leave them for the next iteration, while discarding those with lower success rates.

We are still designing the details of the aforementioned experiment. We expect that when combined with the goal-oriented conjecturing mechanism [10] this project leads to the meta-tool based smart proof search in Isabelle/HOL initially proposed in 2017 [7].

REFERENCES

[1] Georges Gonthier. 2007. The Four Colour Theorem: Engineering of a Formal Proof. In *Computer Mathematics, 8th Asian Symposium, ASCM 2007, Singapore, December 15-17, 2007. Revised and Invited Papers (Lecture Notes in Computer Science)*, Deepak Kapur (Ed.), Vol. 5081. Springer, Berlin, Heidelberg, 333. https://doi.org/10.1007/978-3-540-87827-8_28

[2] Thomas C. Hales, Mark Adams, Gertrud Bauer, Dat Tat Dang, John Harrison, Truong Le Hoang, Cezary Kaliszyk, Victor Magron, Sean McLaughlin, Thang Tat Nguyen, Truong Quang Nguyen, Tobias Nipkow, Steven Obua, Joseph Pleso, Jason M. Rute, Alexey Solovyev, An Hoai Thi Ta, Trung Nam Tran, Diep Thi Trieu, Josef Urban, Ky Khac Vu, and Roland Zumkeller. 2015. A formal proof of the Kepler conjecture. *CoRR* abs/1501.02155 (2015). arXiv:1501.02155 http://arxiv.org/abs/1501.02155

[3] Gerwin Klein, June Andronick, Kevin Elphinstone, Gernot Heiser, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, Thomas Sewell, Harvey Tuch, and Simon Winwood. 2010. seL4: formal verification of an operating-system kernel. *Commun. ACM* 53, 6 (2010), 107–115. https://doi.org/10.1145/1743546.1743574

[4] Gerwin Klein, Tobias Nipkow, Larry Paulson, and Rene Thiemann. 2004. . https://www.isa-afp.org/

[5] Ramana Kumar, Magnus O. Myreen, Michael Norrish, and Scott Owens. 2014. CakeML: a verified implementation of ML. In *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*, Suresh Jagannathan and Peter Sewell (Eds.). ACM, New York, NY, USA, 179–192. https://doi.org/10.1145/2535838.2535841

[6] Xavier Leroy. 2009. Formal verification of a realistic compiler. *Commun. ACM* 52, 7 (2009), 107–115. https://doi.org/10.1145/1538788.1538814

[7] Yutaka Nagashima. 2017. Towards Smart Proof Search for Isabelle. *CoRR* abs/1701.03037 (2017). arXiv:1701.03037 http://arxiv.org/abs/1701.03037

[8] Yutaka Nagashima and Yilun He. 2018. PaMpeR: Proof Method Recommendation System for Isabelle/HOL. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE 2018)*. ACM, New York, NY, USA, 362–372. https://doi.org/10.1145/3238147.3238210

[9] Yutaka Nagashima and Ramana Kumar. 2017. A Proof Strategy Language and Proof Script Generation for Isabelle/HOL. In *Automated Deduction - CADE 26 - 26th International Conference on Automated Deduction, Gothenburg, Sweden, August 6-11, 2017, Proceedings (Lecture Notes in Computer Science)*, Leonardo de Moura (Ed.), Vol. 10395. Springer, Cham, 528–545. https://doi.org/10.1007/978-3-319-63046-5_32

[10] Yutaka Nagashima and Julian Parsert. 2018. Goal-Oriented Conjecturing for Isabelle/HOL. In *Intelligent Computer Mathematics - 11th International Conference, CICM 2018, Hagenberg, Austria, August 13-17, 2018, Proceedings (Lecture Notes in Computer Science)*, Florian Rabe, William M. Farmer, Grant O. Passmore, and Abdou Youssef (Eds.), Vol. 11006. Springer, 225–231. https://doi.org/10.1007/978-3-319-96812-4_19

[11] Lawrence C. Paulson. 2015. A Mechanised Proof of Gödel's Incompleteness Theorems Using Nominal Isabelle. *J. Autom. Reasoning* 55, 1 (2015), 1–37. https://doi.org/10.1007/s10817-015-9322-8