Towards A New Type of Prover: On the Benefits of Discovering Sequences of “Related” Proofs

David M. Cerna[1][2]

[1] Research Institute for Symbolic Computation, Johannes Kepler University Linz, Austria
[2] Institute for Formal Models and Verification, Johannes Kepler University Linz, Austria
david.cerna@risc.jku.at

Abstract

This extended abstract has been written with two goals in mind: 1) to motivate the need for a prover which derives sequences of proofs rather than a single proof, and 2) to present this problem to an audience of experts who can critique this approach and who may be interested in the further development of this project. A *uniform sequence prover* can address issues arising in the area of inductive theorem proving, in particular automated discovery of induction invariants by instance analysis. We illustrate this approach using a novel tree grammar based method introduced by S. Eberhard and S. Hetzl (implemented as *Viper* within the GAPT system). This method necessitates that first-order theorem prover produces sequences of instance proofs which are “related”. This notion of relatedness has so far remained extra-logical and any precision has come from empirical analysis of numerous examples.

While automated theorem proving in the presence of induction is in general undecidable (concerning both the validity and unsatisfiability problems), there are fragments of first-order logic extended by inductive definitions which have a semi-decidable unsatisfiability problem [6]. Unfortunately, these fragments are mathematically quite weak. Even the most powerful theorem provers, based on the techniques outlined in [6][1], such as the superposition prover of V. Aravantinos *et al.* [1], fail to find invariants for even the simplest mathematically interesting problems; for an example see the analysis of the *Eventually Constant Assertion* [5]. Even statements as simple as $x + (x + x) = (x + x) + x$ are problematic for loop discovery methods.

Given the apparent limitations of loop discovery methods, S. Eberhard & S. Hetzl [7] took a different approach to the invariant discovery problem by extracting information from instance proofs. Consider your run-of-the-mill theorem prover such as SPASS, prover9, etc. When such a theorem prover finds a proof, this proof will be cut-free[2]. *Herbrand's Theorem* (also known as the Mid-Sequent Theorem) [9] tells us that we can find and extract a set of quantifier-free instances of the proven statement which together provide a proof of validity.

Without going into too much detail consider a valid statement of first-order logic extended by inductive definitions $\forall x P(x)$ provable from a set of formula Δ where x is a variable over the natural numbers[3]. If $\forall x P(x)$ is provable from Δ using a single induction, then $P(\alpha)$ ought to be provable from Δ without induction (α being a natural number). Thus, using a theorem prover we can get an instance proof Π_α from which we can extract a *Herbrand sequent* (mid-sequent) denoted by $\mathcal{H}_\alpha$. In [7], they produce a special type of tree grammar (based on Herbrand's Theorem) from $\Pi_0, \cdots, \Pi_\alpha$ which induces a quantifier-free second-order unification problem. Solving this problem results is the discovery of an invariant.

Unlike loop discovery methods which have an exceedingly hard time with $x + (x + x) = (x + x) + x$, the tree grammar approach is able to find the invariant $(x + x) + y = x + (x + y)$

[1] What we will refer to as *loop discovery methods*.
[2] For those unfamiliar with the concept of *cut*, this essentially means *analytic*.
[3] In [7] one is not limited to numerals.

and show that $x + (x + x) = (x + x) + x$ is provable by induction from the axioms of addition and successor [8]. While this is a significant step toward more general methods of automated inductive theorem proving, solving the second-order unification problem requires the discovery of uniform constructs within the instance proofs $\Pi_0, \cdots, \Pi_\alpha$. Consider, instead of $x+(x+x) = (x + x) + x$ the statement $x * (x * x) = (x * x) * x$. Note that multiplication is typically defined in terms of addition, and thus it is not entirely clear if the invariant ought to be multiplication based or an addition based, or a mix of both. Each of these possibilities can be abstracted from a particular sequence of instance proofs. A theorem prover may instead of producing instance proofs from one of these sequences unfruitfully zig-zag between various sequences. This makes solving the second-order unification problem intractable.

However, this unfortunate behavior does not mean that the prover is incapable of handling harder problems. Consider the following sentence:

$$F = \forall n(\forall x(E(g(x), n) \vee L(x, n) \ \wedge \forall x(E(x, n) \vee L(x, n)) \ \wedge \hat{Q}(n))$$

where $\hat{Q}$ is defined as follows:

$$\begin{array}{rcl} \hat{Q}(0) & \Rightarrow & \neg L(a, 0) \ \wedge \ \forall x(\neg E(x, 0) \vee \neg E(g(x), 0)) \\ \forall n(\hat{Q}(s(n)) & \Rightarrow & \forall x(\neg E(x, s(n)) \vee \neg E(g(x), s(n))) \ \forall x(\neg L(x, s(n)) \vee E(x, n) \vee L(x, n)) \\ & & \wedge \ \forall x(\neg L(g(x), s(n)) \vee E(g(x), n) \vee L(x, n)) \ \wedge \hat{Q}(n)) \end{array}$$

This sentence is unsatisfiable for all values of n, though it is not completely obvious how one can prove this by resolution. To provide a more intuitive understanding of what the sentence states, one ought to consider it as an invariant of the Infinitary Pigeonhole Principle. Encoded in the predicates E and L is a total function f from the natural numbers (encoded by g and a) to a finite set of natural numbers (encoded by s and 0). The sentence F is the negation of the statement that there exists a value α in the domain of f such that $f(\alpha) = f(g(\alpha))$. Viper found the following invariant after roughly 5 hours of search.

$$(F\{n \leftarrow x\} \rightarrow (E(0, g(a)) \vee E(0, a) \vee \hat{Q}(0))) \wedge \neg(\hat{Q}(s(x)) \wedge \hat{Q}(x) \wedge F\{n \leftarrow s(x)\})$$

Unlike $x * (x * x) = (x * x) * x$, it is not the difficulty of sifting through the various instance proofs which makes the problem difficult because there is (modulo structural variation) only one way to prove each instance, and furthermore each instance is relatively straight forward to prove. Thus, a significant portion of the 5 hours was spend on finding the invariant.

What this example highlights is how difficult constructing the invariant is even when we are provided with a uniform sequence of instance proofs. Instead of producing an arbitrary sequence of instance proofs $\Pi_0, \cdots, \Pi_{s(\alpha)}$, the prover ought to relate $\Pi_{s(\alpha)}$, to the proofs $\Pi_0, \cdots, \Pi_\alpha$. These observations leave a few open questions:

- What does it mean for two proofs Π_1 and Π_2 to be "related"?
- How does one compute whether or not two proofs are "related"?
- Is "related" enough to guarantee discovery of the uniform structures hidden within the instance proofs?

The first question is currently under investigation and is limited to the class of primitive recursive sentences F (see above) inhabits [2]. In [2], we consider a notion of relatedness based on a intrinsic clausal representation of the instances of a primitive recursive formula definition. The clauses of this intrinsic representations can be related through anti-unification techniques [4, 3].

Thus, we consider two clauses arising from the intrinsic clausal representation of two different instance formula related if they have the same anti-unifier. This is a great simplification of the idea and for more details please see [2].

Concerning the other two questions, these are open research question which have so far been approach empirically. Our conjecture is that modern artificial intelligence techniques, especially those arising from the sub-field of machine learning, can be of benefit to computation of "relatedness", i.e. how to choose which clauses which have the same anti-unifier ought to be used in the instance proofs. How one would precisely integrate such methods is still an open question and begs further investigation.

References

[1] Vincent Aravantinos, Mnacho Echenim, and Nicolas Peltier. A resolution calculus for first-order schemata. *Fundamenta Informaticae*, pages 101–133, 2013.

[2] David M. Cerna. On the complexity of unsatisfiable primitive recursively defined $\sum_1$-sentences. Technical report, RISC, 2019.

[3] David M. Cerna and Temur Kutsia. Higher-Order Equational Pattern Anti-Unification. In Helene Kirchner, editor, *3rd International Conference on Formal Structures for Computation and Deduction (FSCD 2018)*, volume 108 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:17, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. Peer Reviewed.

[4] David M. Cerna and Temur Kutsia. Idempotent Anti-unification. Technical report, RISC, January 2018. In Review.

[5] David M. Cerna and Alexander Leitsch. Schematic cut elimination and the ordered pigeonhole principle. In *Automated Reasoning - 8th International Joint Conference, IJCAR 2016, Coimbra, Portugal, June 27 - July 2, 2016, Proceedings*, pages 241–256, 2016. Peer Reviewed.

[6] Hubert Comon. Inductionless induction. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning (in 2 volumes)*, pages 913–962. Elsevier and MIT Press, 2001.

[7] Sebastian Eberhard and Stefan Hetzl. Inductive theorem proving based on tree grammars. *Ann. Pure Appl. Logic*, 166(6):665–700, 2015.

[8] Gabriel Ebner and Stefan Hetzl. Tree grammars for induction on inductive data types modulo equational theories. Technical report, Technical University of Vienna, 2018. https://gebner.org/pdfs/2018-01-29_indmodth.pdf.

[9] Gaisi Takeuti. *Proof Theory*, volume 81 of *Studies in logic and the foundations of mathematics*. American Elsevier Pub., 1975.