Faster Smarter Proof by Induction in Isabelle/HOL with Definitional Quantifiers

Yutaka Nagashima[1 2]

[1] Department of Computer Science, University of Innsbruck, Austria
[2] Yale-NUS College, National University of Singapore, Singapore
yutaka@yale-nus.edu.sg

Abstract

Proof by induction plays a critical role in formal verification and mathematics at large. However, its automation remains as one of the long-standing challenges in Computer Science. To address this problem, we developed `sem_ind`. Given inductive problem, `sem_ind` recommends what arguments to pass to the `induct` tactic. To improve the accuracy of `sem_ind`, we introduced *definitional quantifiers*, a new kind of quantifiers that allow us to investigate not only the syntactic structures of inductive problems but also the definitions of relevant constants in a domain-agnostic style. Our evaluation shows that compared to its predecessor `sem_ind` improves the accuracy of recommendation from 20.1% to 38.2% for the most promising candidates within 5.0 seconds of timeout while decreasing the median value of execution time from 2.79 seconds to 1.06 seconds.

1 Proof by Induction in Isabelle/HOL

The automation of proof by induction is a long-standing challenge in Computer Science. To handle inductive problems, Isabelle [7] offers the `induct` tactics. When using the `induct` tactic, however, users have to manually specify its arguments by answering the following three questions:

- On which terms do they apply induction?
- Which variables do they pass to the `arbitrary` field for variable generalisations?
- Which induction rule do they pass to the `rule` field?

Unfortunately, answering these questions requires users to investigate problems at hand. To automate this process, we previously developed `smart_induct` [4] and `PSL` [6]. `PSL` is a domain-specific language, which allows users to describe proof search strategies. Based on such strategies, `PSL`'s interpreter tries to identify good arguments for the `induct` tactic by executing a possibly expensive proof search. The drawback of this approach is that `PSL` cannot make any recommendations at all if the interpreter fails to complete a proof search. `smart_induct` complements `PSL`'s limitation by suggesting promising arguments for the `induct` tactic without relying on a proof search but based on heuristics encoded in a language called `LiFtEr` [1]. Our previous evaluations, however, identifies two problems of `smart_induct`:

- `smart_induct` tends to be unreliable when variable generalisation is essential.
- `smart_induct` can be quite slow for some inductive problems.

2 Faster Smarter Induction with Definitional Quantifiers

To overcome these limitations, we developed `sem_ind`. Figure 1 presents the overall architecture of `sem_ind`: `sem_ind` firstly produces a small number of induction candidates, using the syntactic structure of problems as a hint. After filtering out candidates that do not even produce sub-goals, `sem_ind` ranks remaining candidates, using induction heuristics encoded in a domain-specific language called `SeLFiE`. Then, out of the five most promising candidates, `sem_ind` produces candidates including generalisation and ranks them using generalisation heuristics written in `SeLFiE`.

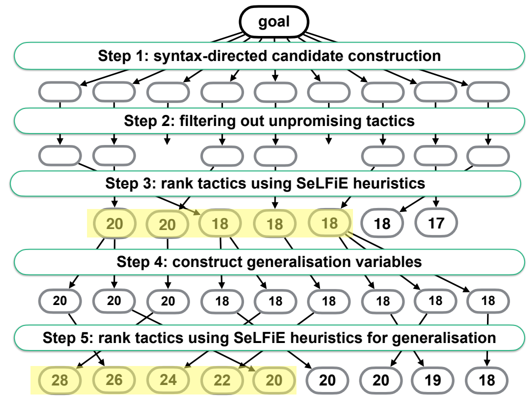

Figure 1: Overview of `sem_ind`.

Table 1 shows how often `sem_ind` produces recommendations within each timeout when applied to 1,095 inductive problems from 22 Isabelle theory files. The first row labelled as "new" shows the results of `sem_ind`, while the second row labelled as "old" shows those of `smart_induct`. This table makes it clear that `sem_ind` performs *faster* than `smart_induct`. This improvement is achieved mainly by the aforementioned architecture, which separates two problems: on what term we should apply induction, and which variables we should generalise while applying induction. This separation allows for the aggressive pruning of less promising candidates for each step, leading to a fewer number of candidates that `sem_ind` has to analyse using `SeLFiE` heuristics.

Table 2, on the other hand, shows how often the recommendations of each tool coincide with the choices of human engineers for the same problem set. For example, Table 2 shows 59.3% in the first row for "top 3". This means that for 59.3% problems the choices of human engineers appear among the three most promising candidates suggested by `sem_ind`. Thus, this table corroborates that `sem_ind` is *smarter* than `smart_induct`, producing more accurate suggestions. The main reason for this improved accuracy is its implementation language, `SeLFiE`. `SeLFiE` provides *definitional quantifier*s, $\exists_{def}$ and $\forall_{def}$, which allow us to encode heuristics that analyse relevant definitions in a domain-agnostic style. Conceptually, a definitional quantifier checks if certain properties hold for all or some of the clauses defining a given constant. For instance, $\exists_{def}(constant, heuristic, arguments)$, checks if there exists a clause defining *constant*, for which *heuristic* holds when applied to *arguments*.

3 Conclusion

We presented `sem_ind` and its implementation language `SeLFiE`. More comprehensive explanations are provided in our drafts [2, 3]. `sem_ind` is fully integrated into the Isabelle ecosystem and freely available at our GitHub repository [5].

tool	0.2s	0.5s	1.0s	5.0s
new	8.8%	24.7%	47.8%	86.8%
old	0.0%	3.5%	16.9%	70.2%

Table 1: Return Rates for Five Timeouts.

tool	top 1	top 3	top 5	top 10
new	38.2%	59.3%	64.5%	72.7%
old	20.1%	42.8%	48.5%	55.3%

Table 2: Coincidence Rates

Acknowledgments

This work has been supported by the following grants:

- the grant of Singapore NRF National Satellite of Excellence in Trustworthy Software Systems (NSoE-TSS),
- the European Regional Development Fund under the project AI & Reasoning (reg.no.CZ.02.1.01/0.0/0.0/15_003/0000466), and
- NII under NII-Internship Program 2019-2nd call.

References

[1] Yutaka Nagashima. LiFtEr: Language to encode induction heuristics for Isabelle/HOL. In *Programming Languages and Systems - 17th Asian Symposium, APLAS 2019, Nusa Dua, Bali, Indonesia, December 1-4, 2019, Proceedings*, pages 266–287, 2019.

[2] Yutaka Nagashima. Faster smarter induction in Isabelle/HOL. *CoRR*, abs/2009.09215, 2020.

[3] Yutaka Nagashima. Selfie: Modular semantic reasoning for induction in isabelle/hol. *CoRR*, abs/2010.10296, 2020.

[4] Yutaka Nagashima. Smart induction for Isabelle/HOL (tool paper). In *2020 Formal Methods in Computer Aided Design, FMCAD 2020, Haifa, Israel, September 21-24, 2020*, pages 245–254. IEEE, 2020.

[5] Yutaka Nagashima et al. data61/PSL. https://github.com/data61/PSL, 2021.

[6] Yutaka Nagashima and Ramana Kumar. A proof strategy language and proof script generation for Isabelle/HOL. In *International Conference on Automated Deduction CADE 2017*, 2017.

[7] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL - a proof assistant for higher-order logic*. Springer, 2002.