# Towards Machine Learning Induction in Poly/ML

Yutaka Nagashima[1,2]

[1] CIIRC, Czech Technical University in Prague,
Prague, Czech Republic
[2] Department of Computer Science, University of Innsbruck,
Innsbruck, Tyrol, Austria

**Abstract**

Induction lies at the heart of mathematics and computer science. However, automated theorem proving of inductive problems is still limited in its power. In this abstract, we first summarize our progress in automating inductive theorem proving for Isabelle/HOL. Then, we present `MeLoId`, our approach to suggesting promising applications of induction without completing a proof search. Our tools are mostly written in Poly/ML, Isabelle's implementation language.

## 1 PSL and Goal-Oriented Conjecturing for Isabelle/HOL

Previously, we developed `PSL` [4] for Isabelle/HOL [6] and its extension to conjecturing mechanism [5] as initial steps towards the development of a smart proof search in Isabelle [2]. With `PSL` one can write the following strategy for induction:

```
strategy DInd = Thens [Dynamic (Induct), Auto, IsSolved]
```

`PSL`'s `Dynamic` keyword creates variations of the `induct` method by specifying different combinations of promising arguments found in the proof goal and its background proof context. Then, `DInd` combines these induction methods with the general purpose proof method, `auto`, and `is_solved`, which checks if there is any proof goal left after applying `auto`. `PSL` keeps applying the combination of a specialization of `induct` method and `auto`, until either `auto` discharges all remaining sub-goals or `DInd` runs out of the variations of `induct` methods.

Sometimes it is necessary for human-engineers to come up with auxiliary lemmas, from which they can derive the original goal. To automate this process, we developed a new atomic strategy, `Conjecture`, as an extension to `PSL`. Given a proof goal, `Conjecture` first produces various conjectures that might be useful to discharge the original proof goal, then inserts these conjectures as the premise of the original goal. Thus, for each conjecture, `PSL` produces two sub-goals: the first sub-goal states that the conjecture implies the original goal, and the second sub-goal states that the conjecture indeed holds. With `Conjecture` integrated into `PSL`, one can write the following strategy:

```
strategy CDInd = Thens [Conjecture, Fastforce, Quickcheck, DInd]
```

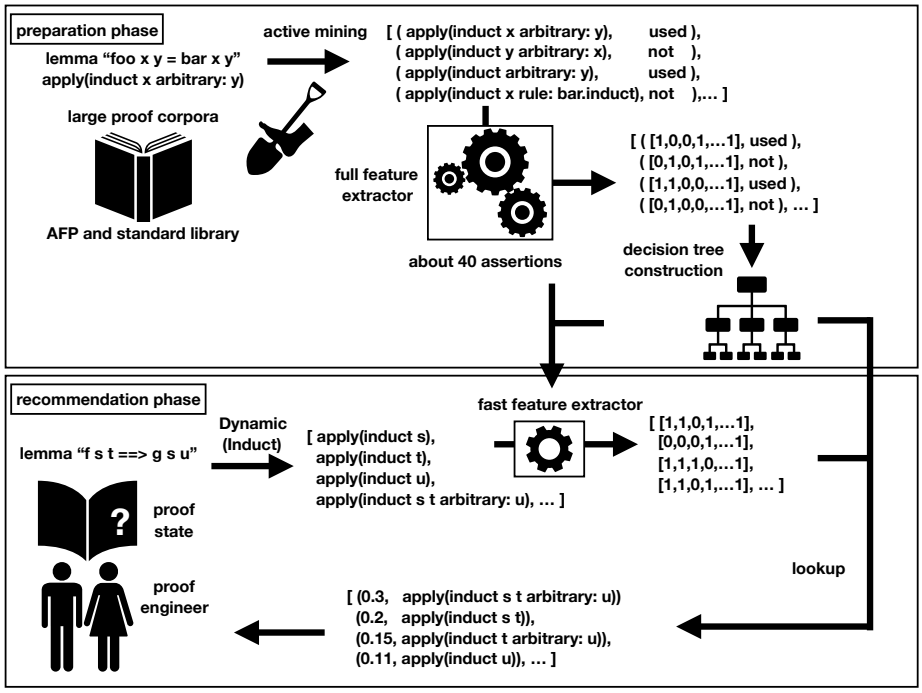
The sequential application of `Fastforce` prunes conjectures that are not strong enough to prove the original goal, whereas the application of `Quickcheck` attempts to prune conjectures that are equivalent to `False`. This way, we can narrow the search space by focusing on promising conjectures; however, when proof goals require many applications of inductions and multiple conjecturing steps, the search space blows up rapidly due to the various `induct` methods produced by the `Dynamic` keyword. Since the `induct` method usually preserves the provability of proof goal, even when the `induct` method has arguments that are inappropriate to discharge the proof goal, counter-example finders, such as `Quickcheck`, cannot discard them. To address this problem, we are developing `MeLoId` to suggest how to apply induction without completing a proof.

## 2 MeLoId: Machine Learning Induction

The figure below illustrates the overall architecture of MeLoId. Similarly to PaMpeR [3], which suggests promising proof methods for a given proof goal and its underlying context, MeLoId tries to learn how to apply induction effectively using human-written proof corpora as training data. In the preparation phase, MeLoId collects invocations of the induct method appearing in the proof corpora and converts each of them into a simpler format, a vector of booleans using an assertion-based feature extractor. Then, MeLoId constructs a regression tree [1], which describes not only which variations of the induct method are promising but also which assertions are useful to make such recommendations in the recommendation phase.

The mechanism of MeLoId differs from that of PaMpeR in multiple ways. First of all, MeLoId analyzes proof corpora via what we call *active mining*: MeLoId first creates various induct methods with distinct combinations of arguments, applies each of them to the goal, and compares their results. Secondly, the input to MeLoId's assertions are the triples of a goal with its context, the arguments to the induct method, and the sub-goal appearing after applying induct, whereas PaMpeR's assertions consider only the first two as input. MeLoId takes the emerging sub-goals into considerations: Since the application of the induct method alone is not time-consuming, we expect that it is desirable to improve the accuracy of recommendation using the emerging sub-goals even at the cost of the extra time spent by the induct method. Third, MeLoId assertions tend to analyze the structures of the triples, while PaMpeR's assertions tend to focus on the names of constants and types appearing in the proof goal at hand.

We have implemented the active mining mechanism and around 40 assertions. Our preliminary experiment suggests that the feature extractor successfully distills the essence of some undesirable combinations of arguments of induct. However, more comprehensive evaluation and further engineering efforts remain as our future work.



# 3 Acknowledgments

This work was 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).

# References

[1] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[2] Yutaka Nagashima. Towards smart proof search for isabelle, 2017.

[3] Yutaka Nagashima and Yilun He. PaMpeR: Proof method recommendation system for Isabelle/HOL. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ASE 2018, pages 362–372, New York, NY, USA, 2018. ACM.

[4] 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.

[5] Yutaka Nagashima and Julian Parsert. Goal-oriented conjecturing for Isabelle/HOL. In Florian Rabe, William M. Farmer, Grant O. Passmore, and Abdou Youssef, editors, *Intelligent Computer Mathematics*, pages 225–231, Cham, 2018. Springer International Publishing.

[6] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL - a proof assistant for higher-order logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer, 2002.