# Project Proposal: Forward Reasoning in Hindsight

Michael Rawson[1], Zsolt Zombori[2,3],
Maximilian Doré[4], and Christoph Wernhard[5]

[1] TU Wien, Austria michael@rawsons.uk
[2] Alfréd Rényi Institute of Mathematics, Hungary zombori@renyi.hu
[3] Eötvös Loránd University, Budapest, Hungary
[4] University of Oxford, United Kingdom maximilian.dore@cs.ox.ac.uk
[5] University of Potsdam, Germany info@christophwernhard.com

**Abstract**

Hindsight Experience Replay is a promising technique in reinforcement learning. However, we argue that its interpretation in *refutational* theorem proving is somewhat indirect, and instead propose its application in reasoning settings where consequences are derived from axioms alone until a goal is reached. Such settings include many sequent-like calculi, condensed detachment, non-trivial fragments of dependently-typed languages such as Agda, and we conjecture that unit equational reasoning is also suitable.

## 1 Hindsight Experience Replay and Reasoning

Applied to automated theorem proving, Hindsight Experience Replay (HER) [1] could be summarised as follows. Suppose a goal $G$ should be reached from some premises $P$ via some calculus that generates consequences $C$ of $P$. A model $\mathcal{M}(C \mid G, \theta)$ is employed as a heuristic for the enumeration of consequences based on the goal and learned parameters $\theta$. If a proof is not found within some resource limit, training data can be obtained by taking some derived consequence $\hat{C}$, pretending *in hindsight* that this was the desired goal, and inspecting its derivation. Positive examples are then those other consequences used in the derivation of $C$, and negative examples those that are not in the derivation of $C$. These are then used to update $\theta$.

This procedure yields labelled tuples $\langle C, \hat{C}, l\rangle$, where $l$ is a Boolean label. These tuples represent a kind of selection problem: is $C$ useful for deriving $\hat{C}$? Implicit in this régime is that the model is capable of *generalising*, such that training on the proxy $\langle C, \hat{C}, k\rangle$ task yields an improvement on the real task $\langle C, G, ?\rangle$. This deviates from the norm in machine-learning-for-theorem-proving: conventionally, a set of goal-premises pairs ("problems") are given such that an untrained system can solve some, but not all, of the problems. It is assumed that training on data obtained from the solved pairs helps the trained system to solve the unsolved pairs. A MaLARea-style loop [12] is often used. By contrast, HER does not need to solve a single problem, and can even be trained on only one problem in principle. It is not clear whether it is necessary or even desirable to have a set of solved pairs in the training set.

The seminal work of Aygün et al. [2] shows great promise applying HER to automated theorem proving. Aygün et al. employ a *refutational* calculus, so the "true goal" $G$ is always $\perp$, a problem which is ingeniously worked around: the original goal is taken to be the input clause set $S$, including the negated conjecture, and when some clause $D$ is derived it can be used as training data in hindsight for the goal $S, \neg D$. The first author believes that even more could be achieved with HER-style learning in theorem proving with a non-refutational calculus, where the notion of true and hindsight goals are perhaps more natural.

## 2 Proposal

We therefore propose employing HER with a non-refutational, forward-chaining calculus. Proofs in such calculi proceed from premises only, deriving new conclusions until the goal (or some generalisation thereof) is reached. The goal, or its negation, is *not* involved in search, and is only used in our proposed setup as a parameter to $\mathcal{M}$. This does unfortunately rule out popular calculi such as superposition. However, our project could profitably include:

**Condensed Detachment.** Carew Meredith's *condensed detachment* (in essence modus ponens with unification) provides an ideal setting for us, in which there are also hard problems of interest [14]. `LCL073-1` in the TPTP library [11] is particularly hard for automatic systems, with a fully-automatic proof obtained only recently [10]. It is both complete and efficient to apply the condensed detachment rule only, and the goal is considered proved when a consequence is derived that subsumes it. Despite its simplicity, condensed detachment, extended with a second proof constructor for quantification, is sufficient as sole basis [6] for the formalisation of mathematics with Metamath [7].

**Dependent Type Theory, Proof Schemas and Combinators.** In dependently-typed interactive theorem provers such as Agda [8], a goal type is dispatched by constructing a term of that type. Many terms can be constructed using only function application ($s\ t$), where $s$ and $t$ are themselves applications or names of previous constructions. Proof search hence typically proceeds by (partially) applying functions and type-checking the refined proof goals. By also using explicit elimination principles for inductive types (defined uniformly), it is moreover possible to carry out proofs by induction using only function applications. Agda therefore fits very naturally in our approach of deriving consequences from already known results, and its type-checker quickly computes the ostensible proof goal. We are also struck by the similarity to *compressed combinatory proof structures*, implemented (so far only for backwards search) within the CCS prover [13].

**Unit Equational Reasoning.** We avoid the term *rewriting* as what we propose does not fit into the usual mould of rewrite systems [3], but the problem is the same: given a set of universally-quantified equations, determine whether a goal $s = t$ is implied by the set. This is usually done by negating the goal and rewriting it, possibly *overlapping* equations along the way. In order to fit this into our setting it seems almost sufficient to only overlap equations (without employing term orderings) until the goal is reached. However, checking that the goal has been reached is not straightforward — consider that we have derived $a = b$ and $c = d$ but the goal is $f(a, c) = f(b, d)$ — but perhaps a congruence rule or some generalised notion of reaching the goal fixes this problem. We would be especially pleased to hear from the AITP community about this. Note that some condensed detachment problems in the TPTP concern axiomatisations of equality, and *vice versa* some unit-equational problems encode condensed detachment.

**Sequent Calculi.** Many sequent-like calculi fit naturally. However, we suspect this is rarely useful, as such calculi are not designed for automated deduction and are rarely efficient without substantial modification.

**Other.** We would be delighted to hear about other calculi that we have not encountered but that fit into this setting, or ideas about how to adapt existing refutational calculi such as superposition or resolution. These calculi have already been used for finding consequences of axioms [5], with applications to software verification [4].

## 3 Initial Results

Previous work by some of the authors [9, 10] employed a hindsight mechanism for selecting useful lemmas in the context of condensed detachment. Here we wish to push this idea further, and to include other proof calculi. A prototype system using HER to guide condensed detachment search has been developed by the first author[1]. The system hard-codes the challenge problem `LCL073-1` and then applies saturation-style search with the condensed detachment rule, starting from the single axiom and limited to 100 activations. Condensed detachment has a habit of producing exponentially large terms if unsupervised, so for the prototype a limit on the term size is enforced. After 100 activations, the search space is sampled to produce positive and negative training data for a sampled hindsight goal. These data are used periodically to train a neural network, which selects the next activated consequence in $\epsilon$-greedy fashion.

Fixing a single hard problem has several advantages, as well as the significant reduction in system complexity. One is that we can test the hypothesis that training on a single problem is sufficient. Another is that we can *monitor the progress* of the system by recording which intermediate results in a known proof are derived in a particular saturation run. This is a stochastic measure by design due to the $\epsilon$-greedy policy, and somewhat flawed in that alternative proofs are not recognised. However, the system rapidly and clearly improves over a random baseline, and the current record run derived 21 results from the reference 46-step proof[2].

## 4 Outlook

From this prototype we conclude that our approach shows promise, but has room for improvement both in terms of proving power (the other half of `LCL073-1` remains to be shown!) and reusability. We intend several future directions. Manual inspection of the system's behaviour may reveal some deficiencies that prevent it progressing further, although we suspect that the challenge problem is simply very hard and finding a 46-step proof in 100 activations requires very precise selection. A greater degree of compute (currently the first author's trusty desktop) and a different experimental setup including easier problems may prove profitable for this reason. We also intend to begin investigations outside of condensed detachment. Recent work has shown that applicative terms may be a viable route to the integration of Agda, when combined with explicit induction schemes, hand-selected functions such as dependently-typed SKI combinators, and Agda's support for type inference and implicit arguments.

## References

[1] Marcin Andrychowicz, Dwight Crow, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5048–5058, 2017.

[2] Eser Aygün, Ankit Anand, Laurent Orseau, Xavier Glorot, Stephen Marcus McAleer, Vlad Firoiu, Lei M. Zhang, Doina Precup, and Shibl Mourad. Proving theorems using incremental learning and hindsight experience replay. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022,*

[1] https://github.com/MichaelRawson/hercd

[2] Note that the reference proof is used only for monitoring. Care is taken to ensure it does not affect training.

*17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 1198–1210. PMLR, 2022.

[3] Franz Baader and Tobias Nipkow. *Term rewriting and all that.* Cambridge University Press, 1998.

[4] Laura Kovács and Andrei Voronkov. Finding loop invariants for programs over arrays using a theorem prover. In Marsha Chechik and Martin Wirsing, editors, *Fundamental Approaches to Software Engineering, 12th International Conference, FASE 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings*, volume 5503 of *Lecture Notes in Computer Science*, pages 470–485. Springer, 2009.

[5] Richard Char-Tung Lee. *A completeness theorem and computer program for finding theorems derivable from given axioms.* PhD thesis, University of California, Berkeley, CA, 1967.

[6] Norman D. Megill. A finitely axiomatized formalization of predicate calculus with equality. 36(3):435–453, 1995.

[7] Norman D. Megill. Metamath. In Freek Wiedijk, editor, *The Seventeen Provers of the World, Foreword by Dana S. Scott*, volume 3600 of *Lecture Notes in Computer Science*, pages 88–95. Springer, 2006.

[8] Ulf Norell. Dependently typed programming in Agda. In Andrew Kennedy and Amal Ahmed, editors, *Proceedings of TLDI'09: 2009 ACM SIGPLAN International Workshop on Types in Languages Design and Implementation, Savannah, GA, USA, January 24, 2009*, pages 1–2. ACM, 2009.

[9] Michael Rawson, Christoph Wernhard, and Zsolt Zombori. Learning to identify useful lemmas from failure (extended abstract). In Michael R. Douglas, Thomas C. Hales, Cezary Kaliszyk, Stephan Schulz, and Josef Urban, editors, *AITP 2023 (Informal Book of Abstracts)*, 2023.

[10] Michael Rawson, Christoph Wernhard, Zsolt Zombori, and Wolfgang Bibel. Lemmas: Generation, selection, application. In Revantha Ramanayake and Josef Urban, editors, *Automated Reasoning with Analytic Tableaux and Related Methods — 32nd International Conference, TABLEAUX 2023, Prague, Czech Republic, September 18–21, 2023, Proceedings*, volume 14278 of *Lecture Notes in Computer Science*, pages 153–174. Springer, 2023.

[11] Geoff Sutcliffe. The TPTP problem library and associated infrastructure — from CNF to TH0, TPTP v6.4.0. *J. Autom. Reason.*, 59(4):483–502, 2017.

[12] Josef Urban, Geoff Sutcliffe, Petr Pudlák, and Jiří Vyskocil. MaLARea SG1 — machine learner for automated reasoning with semantic guidance. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 12-15, 2008, Proceedings*, volume 5195 of *Lecture Notes in Computer Science*, pages 441–456. Springer, 2008.

[13] Christoph Wernhard. Generating compressed combinatory proof structures — an approach to automated first-order theorem proving. In Boris Konev, Claudia Schon, and Alexander Steen, editors, *PAAR 2022*, volume 3201. CEUR-WS.org, 2022. https://arxiv.org/abs/2209.12592.

[14] Christoph Wernhard and Wolfgang Bibel. Learning from Łukasiewicz and Meredith: Investigations into proof structures. In André Platzer and Geoff Sutcliffe, editors, *Automated Deduction - CADE 28 - 28th International Conference on Automated Deduction, Virtual Event, July 12-15, 2021, Proceedings*, volume 12699 of *Lecture Notes in Computer Science*, pages 58–75. Springer, 2021.