Elements of Reinforcement Learning in Saturation-based Theorem Proving*

Martin Suda

Czech Technical University in Prague, Czech Republic

The Promise and the Hype

Reinforcement learning (RL) [18], especially its *deep* variant relying on modern neural networks, is probably the most fashionable method for attacking problems in our machine learning (ML) era. The impressive successes in board games [13] or on the ATARI benchmark [3] justify the excitement. Moreover, it is very appealing to have the machine look for a solution unbiased by our preconceptions, since this intuitively increases the chances of discovering brand new strategies. However, we should also be aware of the various shortcomings of the approach [4].

In automatic theorem proving (ATP), we have seen the Monte-Carlo tree search paradigm [8] extend a connection tableaux prover [7] or, more recently, a saturation-based setup called TRAIL [1], featuring an interesting idea of multiplicative attention for expressing a dependence on prover's state. Despite the partial successes, we still seem to be far from getting a system that could challenge a state-of-the-art prover in a real-time evaluation (Kaliszyk et al. [7] use abstract time, TRAIL falls short of improving over plain E [11]), let alone on a versatile benchmark such as the TPTP library [17] (both mentioned works target a more uniform Mizar benchmark).

Ancient Lore and its Contemporary Extensions

It is instructive to recall the basic RL ingredients and project them to the state-of-the-art (SotA) saturation-based ATP technology and its recent improvements by ML. In this light, we can think of a prover as being guided by an *agent*, who monitors the prover's *state* and chooses appropriate *actions* to reach the goal of deriving the empty clause, ideally in the shortest time possible. A learning feedback for the agent should come in the form of a *reward*, received after executing each individual action or at the end of a proof attempt.

In saturation-based ATP (setting aside the role of proving strategies) the guiding agent is most fittingly identified with the clause selection heuristic [see, e.g., 12]. A proving state naturally decomposes into two conceptual parts: a *static* one, the formula subject to proving, and an *evolving* one, any information influencing what should be done next in order to prove it. Finally, the available actions correspond simply to the passive (unprocessed) clauses.

The author finds it noteworthy, that SotA provers, backed by decades of research in the field, mostly ignore the state for clause selection. Except possibly for a few bits to remember which queue to select the next clause from, the effective state is blank[1] and each selection aims greedily at the best available clause. Could this indicate there is actually little hope for meaningful proof planning in general purpose ATP?

The situation is different with the recent improvements by ML. Information about the conjecture (i.e., a static state) has been included since the second version of ENIGMA [5]

*Supported by the Czech Science Foundation project 20-06390Y and the project RICAIP no. 857306 under the EU-H2020 programme.

[1] Conjecture clauses sometimes get a different status for some heuristics [e.g., 14], but only uniformly, not depending on what the conjecture actually is.

and, before that, by the work of Loos et al. [10]. While the latter paper does not perform a corresponding ablation, ENIGMA is reported to moderately improve thanks to the conjecture features.[2] As mentioned, an evolving state is proudly included in TRAIL [1] and also in, e.g., ENIGMAWatch [2]. In both cases, the papers report on an improvement thanks to the evolving state feature. Although this is only shown for Mizar, maybe there is hope after all!

Let us close this section by returning to the concept of reward. It seems unrealistic to ever learn useful guidance for ATP by only rewarding the final proving step.[3] All the mentioned systems agree and retrospectively reward (or mark as positive) not just the final, but all the actions that contributed to the found proof. An ambiguity in the terminology seems to arise: can we have RL without an (explicit) reward? In the light of the just explained, does TRAIL really differ that much from looping in ENIGMA [6], which also iteratively improves the learned knowledge, generating training data for the next iteration using the current knowledge?

Back to the Drawing Board

In this project, we want to attack the ATP+RL target from a new angle. Rather than immediately aiming at designing an (end-to-end trainable) agent with access to the complete state (that could, in principle, solve the whole formula before the search even begins and would, effectively, only use the prover as a verifier), we want to start as close as possible to the SotA design and use RL as a research tool to further our understanding of proof search dynamics.

One possible setup, which is—at first sight—so glaringly impractical that it probably has not been tried yet, is training an agent *on a single problem only*. Yes, with a complete state description the agent can just memorize a proof (once it finds one, maybe after a long initial search) and then just keep replaying it afterwards. However, there are at least two aspects which make already this simple setup interesting.

First, in a typical proof search a complete state description very soon becomes intractably large to be processed by the agent efficiently (we talk about thousands of clauses generated in a few seconds) and thus *cheaply computable abstractions* have to come to rescue. Going back to the SotA agent, we often find it happy with representing each clause by just two numbers, its age and weight. Guiding towards a previously seen proof becomes an interesting challenge for an agent when "partially blindfolded" by simple abstractions.

The second aspect is the inherent *fragility of proof search*, on which the author recently shed light using randomization [16]. It turns out that even very small changes in a concrete run, introduced at the level of "don't care non-determinism" such as the exact order of literals in a newly generated clause, can have a tremendous impact on how long it takes to find a proof. Further investigation is needed to pinpoint what exactly causes so much chaos in our provers.[4]

In this project, we plan to undertake such investigation with the tools of RL, making use of the randomization code from our previous work [16] to turn theorem proving into a stochastic environment. This will create a second challenge for our agent, forcing it to seek robust strategies. Ultimately, we would like the agent to be able to recognize situations that are particularly unstable, so that it could respond particularly carefully. By examining the used features, we, as prover developers, will then hopefully learn how to build more robust provers ourselves.

[2] There is, however, also a meta-point: Deepire's guidance [15] does not depend on the conjecture in this sense, yet the system achieves a comparable, if not better, performance to ENIGMA on Mizar [6].

[3] And letting the prover figure out which actions were actually useful for the success by trial and error.

[4] Although a major part is probably caused by the eager simplifications and their interactions with clause selection (generating inferences on their own would stay nicely confluent), there is also the possibility that a sudden selection of what we could call a "highly explosive clause" dramatically changes the content of the weight-sorted queue, rendering the previously observed proof out of reach.

References

[1] M. Crouse, I. Abdelaziz, B. Makni, S. Whitehead, C. Cornelio, P. Kapanipathi, K. Srinivas, V. Thost, M. Witbrock, and A. Fokoue. A deep reinforcement learning approach to first-order logic theorem proving. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pp. 6279–6287. AAAI Press, 2021.

[2] Z. A. Goertzel, J. Jakubuv, and J. Urban. ENIGMAWatch: ProofWatch meets ENIGMA. In *Automated Reasoning with Analytic Tableaux and Related Methods - 28th International Conference, TABLEAUX 2019, London, UK, September 3-5, 2019, Proceedings*, vol. 11714 of *Lecture Notes in Computer Science*, pp. 374–388. Springer, 2019.

[3] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. G. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 3215–3222. AAAI Press, 2018.

[4] A. Irpan. Deep reinforcement learning doesn't work yet. https://www.alexirpan.com/2018/02/14/rl-hard.html, 2018.

[5] J. Jakubuv and J. Urban. Enhancing ENIGMA given clause guidance. In *Intelligent Computer Mathematics - 11th International Conference, CICM 2018, Hagenberg, Austria, August 13-17, 2018, Proceedings*, vol. 11006 of *Lecture Notes in Computer Science*, pp. 118–124. Springer, 2018.

[6] J. Jakubuv and J. Urban. Hammering mizar by learning clause guidance (short paper). In *10th International Conference on Interactive Theorem Proving, ITP 2019, September 9-12, 2019, Portland, OR, USA*, vol. 141 of *LIPIcs*, pp. 34:1–34:8. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[7] C. Kaliszyk, J. Urban, H. Michalewski, and M. Olšák. Reinforcement learning of theorem proving. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 8836–8847, 2018.

[8] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In *ECML 2006, Proceedings*, vol. 4212 of *Lecture Notes in Computer Science*, pp. 282–293. Springer, 2006.

[9] L. Kovács and A. Voronkov. First-order theorem proving and vampire. In *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, vol. 8044 of *Lecture Notes in Computer Science*, pp. 1–35. Springer, 2013.

[10] S. M. Loos, G. Irving, C. Szegedy, and C. Kaliszyk. Deep network guided proof search. In *LPAR-21, 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Maun, Botswana, May 7-12, 2017*, vol. 46 of *EPiC Series in Computing*, pp. 85–105. EasyChair, 2017.

[11] S. Schulz, S. Cruanes, and P. Vukmirović. Faster, higher, stronger: E 2.3. In *Proc. of the 27th CADE, Natal, Brasil*, number 11716 in LNAI, pp. 495–507. Springer, 2019.

[12] S. Schulz and M. Möhrmann. Performance of clause selection heuristics for saturation-based theorem proving. In *Automated Reasoning - 8th International Joint Conference, IJCAR 2016, Coimbra, Portugal, June 27 - July 2, 2016, Proceedings*, vol. 9706 of *Lecture Notes in Computer Science*, pp. 330–345. Springer, 2016.

[13] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.

[14] M. Suda. Aiming for the goal with sine. In *Vampire 2018 and Vampire 2019. The 5th and 6th Vampire Workshops*, vol. 71 of *EPiC Series in Computing*, pp. 38–44. EasyChair, 2019.

[15] M. Suda. Vampire with a brain is a good ITP hammer. In *Frontiers of Combining Systems - 13th International Symposium, FroCoS 2021, Birmingham, UK, September 8-10, 2021, Proceedings*, vol. 12941 of *Lecture Notes in Computer Science*, pp. 192–209. Springer, 2021.

[16] M. Suda. Vampire getting noisy: Will random bits help conquer chaos? (system description). EasyChair Preprint no. 7719, EasyChair, 2022.

[17] G. Sutcliffe. The TPTP Problem Library and Associated Infrastructure. From CNF to TH0, TPTP v6.4.0. *Journal of Automated Reasoning*, 59(4):483–502, 2017.

[18] R. S. Sutton and A. G. Barto. *Reinforcement learning - an introduction.* Adaptive computation and machine learning. MIT Press, 1998.