Machine Learner for Automated Reasoning 0.4 and 0.5

Cezary Kaliszyk[1], Josef Urban[2], and Jiří Vyskočil[3]

[1] University of Innsbruck, Austria
[2] Radboud University Nijmegen
[3] Czech Technical University

Abstract

Machine Learner for Automated Reasoning (MaLARea) is a learning and reasoning system for proving in large formal libraries where thousands of theorems are available when attacking a new conjecture, and a large number of related problems and proofs can be used to learn specific theorem-proving knowledge. The last version of the system has by a large margin won the 2013 CASC LTB competition. This paper describes the motivation behind the methods used in MaLARea, discusses the general approach and the issues arising in evaluation of such system, and describes the Mizar@Turing100 and CASC'24 versions of MaLARea.

1 Introduction: MaLARea as an Experiment with Data-Driven AI/ATP Methods

Machine Learner for Automated Reasoning (MaLARea) is a (meta-)system for automated theorem proving (ATP) in large theories consisting of thousands of formulas, symbols and proofs. The main motivation behind the system has been to develop, employ, and evaluate various kinds of heuristic guiding methods that are useful for reasoning in such large (especially complex mathematical) theories. A particular goal was to start exploring analogies with how trained mathematicians work, i.e., how they develop and accumulate problem-solving knowledge, learn it from others and re-use it for attacking more and more difficult problems. Providing at least one underlying database for developing such methods has also been one of the main motivations for translating the Mizar library to ATP formats. MaLARea started to be developed in 2007, when the translation (MPTP [15, 16]) of the whole Mizar library to first-order logic (FOF TPTP) was largely finished. An AI/ATP competition on a smaller set of related large-theory MPTP problems – the MPTP Challenge[1] – was designed in 2006 to measure the strength of various large-theory AI/ATP systems and techniques limited to CASC-scale resources, and to encourage further development of such systems. Our hope has been that the more successful AI/ATP techniques developed experimentally for MaLARea on the smaller benchmarks will eventually be deployed as systems that help with mathematics done over large formal libraries.

The system design has been influenced by earlier experiments done over the whole translated Mizar library and its parts. The main guiding idea tried already in [15] was to use the previous Mizar library proofs to learn which of the thousands of available theorems in the library could be relevant for a new conjecture. In [16], this was for some experiments complemented by learning also from the ATP proofs of related Mizar problems. The heuristic justification behind this overall approach to large-theory ATP was that predicting relevance is in general very hard, i.e., it is intractable or undecidable (depending on one's exact assumptions) to estimate how a theorem will be proved and what previous knowledge will best serve that. So once such knowledge has been (often expensively) discovered for some problems, it should be re-used as much as possible,

[1] `http://www.cs.miami.edu/~tptp/MPTPChallenge/`

becoming an integral part of the ATP algorithms, instead of disregarding it and relying just on simpler (e.g., symbol-based) pre-programmed criteria that (at least currently) do not seem to be quite successful in emulating the reasoning processes leading to the human discovery of complicated mathematical proofs. Such approach to algorithm design has been recently called *data-driven* [13]. This paradigm does not seem yet entirely established in the domain of Automated Reasoning, despite its generally acknowledged crucial role in recent major AI (and also AR) achievements such as the IBM Watson system. We hope that the techniques and results described here will motivate further interest in such ATP methods. The paper is organized as follows. Section 2 explains the main techniques implemented in the early versions of MaLARea. Section 3 briefly discusses the various concerns and issues related to evaluating such AI/ATP systems. Section4 describes the version of MaLARea used at the Mizar@Turing100 AI/ATP competition, and Section 5 describes the version used at the CASC'24 competition.

2 Basic Ideas and Techniques

The basic techniques used already by early versions of MaLARea are described in detail in [17, 20]. The metasystem relies on one or several core ATPs (originally SPASS [21] and E [12]) and one or several machine learners (originally the naive Bayes learner provided by the SNoW system [2]). The goal in benchmarks like the MPTP Challenge is to solve as many large-theory problems as possible within a global time limit. This allows systems to spend time on analyzing and solving different problems, and to transfer the knowledge obtained while solving one problem to other problems. The main way how MaLARea does this is by incremental exploration and learning of the *relevance relation* saying which (sets of) existing theorems are likely to be useful for proving a given conjecture. Making decisions about which of the problems to attack next, which axioms to use, and how it is going to improve the overall knowledge about the relevance relation is an instance of the general *exploration vs. exploitation* framework studied, e.g., on multi-armed bandits [3] in the domain of reinforcement learning [14].

Preprocessing, Features: The system starts by several pre-processing steps done on the input set of all problems, during which the formulas in the problems are syntactically normalized (using the `tptp4X` tool from the TPTP distribution) and checked for duplicate names. Different names for the same formula are merged into one unique name. This typically later benefits the learning of the relevance relation, which targets the formula names. More elaborate schemes for normalization in large libraries are possible (and could benefit the knowledge re-use even more): for example, splitting of all conjunctions (and naming the conjuncts) and normalizing also the symbol names using recursive content-based hashing has been recently introduced for the whole Flyspeck [7]. The normalized formulas are then subjected to the initial extraction of suitable features that approximate the formulas for the learning systems. In the first version, only the set of (non-variable) symbols was used. In the later versions, the set of all terms in all formulas was enumerated using a tool based on E prover's shared term banks, and the serial numbers of the shared terms (or just their internal string representation) were added to the set of formula features. Overlap (measured, e.g., using the Jaccard index [5]) on such feature sets then provides a much more detailed notion of similarity between two formulas. Since this notion of similarity can be too fine, various further modification of this idea have been considered. One that was also introduced early is the use of just one generic variable in all formulas, so that terms that differ only in variables can still give rise to similarity between formulas. Again, this idea has been in recent related systems extended in various ways [6, 9]. The number of such features can be quite high (thousands to millions, depending on the size of the data set), however for relatively simple sparse learning methods such as naive Bayes this is not an efficiency problem when compared to the times used by the ATPs. In addition

to such purely syntactic features, much more semantic formula features were added in [20] (no longer during the preprocessing phase), based on the validity of the formulas in a large pool of finite models that are typically created dynamically as counter-examples during the main MaLARea loop execution. Unlike the syntactic proximity, which will make the formulas ϕ and $\neg\phi$ very close, the model-based proximity should much better correspond to the "true semantic relation" of the formulas in the Lindenbaum algebra, providing also basis for using (and learning) straightforward (counter-)model-based criteria for premise selection [20].

Main Loop: After the preprocessing steps, the system proceeds by running the main loop which interleaves the work of the ATP systems with the work of the learning (and other premise-selection) systems. Based on the conjecture features and previous proofs, the learning systems try to select the most relevant premises for the conjectures that are still unproved. The ATP systems in turn try to (dis)prove the problems with various numbers of the most relevant premises, adding more proofs or counter-models to the common (initially empty) knowledge base. There are various options that govern this main loop, controlling mainly the premise numbers, time limits, and the speed of re-learning on the knowledge base when new proof (or counter-model) data become available. The loop ends either by solving all problems, timing out, or after trying all allowed combinations of premise numbers and time limits for the remaining conjectures without obtaining any new information (proof or counter-model) that could update the relevance relation. The main purpose behind this loop is to efficiently combine educated guessing (induction) based on the current knowledge of the world with deductive confirmation (or disproof) of such guesses, which produces further knowledge. The early versions of MaLARea outperformed other systems on (the large-theory division of) the MPTP Challenge and also on the Mizar category of the first (2008) CASC LTB (large-theory batch) competition.

3 Large-Theory Competitions and Evaluating MaLARea

A major issue triggered by MaLARea turned out to be the evaluation of such AI systems. Until the introduction of the LTB division in 2008 (consisting then of the Mizar, SUMO, and Cyc problems), the CASC competition largely prohibited knowledge re-use between different problems and systems that would come with pre-recorded information about the solutions of other (typically TPTP) problems.[2] Good knowledge extraction and re-use is however the main research topic of data-driven AI methods. The MPTP Challenge addressed this dilemma by (i) keeping the CASC rules preventing pre-recording, (ii) allowing arbitrary recording and re-use between the problems within the competition time, (iii) providing sufficiently many – 252 – related problems of varied difficulty, and (iv) allowing further knowledge re-use by letting systems return to unsolved problems. An even better solution (similar, e.g., to machine learning competitions such as the Netflix Challenge) would have been to use a public dataset for system preparation and an unknown similar dataset for evaluation. To some extent this was realized in the first (2008) CASC LTB competition, when the MPTP Challenge benchmark could be used for preparing systems, but the competition data for the Mizar category (following the same design as the MPTP Challenge) were unknown prior to the competition. The next (2009) CASC LTB however significantly decreased the number of Mizar competition problems (to 40), and focused on a query answering mode, in which systems are additionally not allowed not to return to unsolved problems (point (iv) above). This LTB setup was kept until 2012. A real-time query answering mode is in principle a valid scenario in large-theory formal mathematics [7, 19, 9], but the low number of preceding competition problems results in only a few

[2] The fine point of such rules is of course construction of targetted ATP strategies and their use based on problem classification methods. The finer such methods are, the closer they are to pre-recording information about the problem solutions.

proofs to learn from, which mostly does not correspond to reality in (formal) mathematics. Such simplification (denial of existence of great amount of useful data) obviously largely diminishes the usefulness of data-driven methods, whose development was the primary motivation behind MaLARea, MPTP, and the first large-theory benchmarks. Consequently, MaLARea did not compete in CASC LTB from 2009 to 2011.

4 MaLARea 0.4 at Mizar@Turing100

In 2012, the Mizar@Turing100 competition brought back the possibility of learning from many related proofs. It was based on the larger MPTP2078 benchmark [1] and followed the MPTP Challenge rules (using 400 competition problems), modified by additionally pre-releasing 1000 training problems together with a large number (13455) of their (ATP or Mizar) proofs. The systems were allowed to preprocess these problems and proofs in arbitrary way before the competition, and use any knowledge thus obtained in the competition.[3] Three main additions were made to MaLARea, accommodating the new rules and taking into account recent large-theory research.

Use of the training proofs to initialize the learning systems: The 13455 training proofs were analyzed in the same way as the proofs obtained during the competition (extracting the premises used), and the resulting table became part of the competition system and used for the first learning at the start of the competition. The main interesting AI issue was how to best learn the relevance relation from many different proofs (on average 13) of the same problem. The experimentally obtained best answer to this was (at that time, for Mizar and available learners) to use the shortest proof available [11, 10].

Use of non-learning predictors and their combining with learners: The research done in [11] also motivated the second addition: using E's version of SInE [4] to produce its own relevance ordering of premises, which may then be also linearly combined with the ordering produced by the machine learner. In [11] such combination of rankers improved the final ATP performance by 10%.

Use of ATP strategies automatically constructed on the training data: The newly developed Blind Strategymaker (BliStr) strategy evolving system [18] was used to automatically construct new ATP strategies on the training problems. Building of Blind Strategymaker itself was a response to the preliminary measurements showing that E 1.6 (used as the only ATP in MaLARea for the competition) could in 300s prove only 518 of the training problems, compared to 691 proved by Vampire 1.8.[4] After 6 runs (30 hours of real time on 12-core Xeon 2.67GHz server), BliStr developed a set of E strategies that (using altogether 300s) raised the performance of E to about 650 training problems. This 25% improvement turned out to carry over also to the 400 competition problems, probably thanks to the relatively strong guards against overfitting (versatility criterion) used in the BliStr strategy-evolution loop [18].

The overall MaLARea parameters have then been (manually) tuned on a random subset of 100 problems from the training dataset, restricting the training data to the remaining 900 problems. To speed up the expensive testing runs, a file-based (content-indexed) cache of solutions was added, growing in the end to about 2.5M unique solutions. The final MaLARea version solved 257 of the 400 competition problems, but lost 17 due to a bug in proof delivery and thus came second. The version without the new E strategies solved only 214 problems [18].

[3] This was broadly motivated by the presence of a large number of proofs in the ITP systems' libraries, and the possibility to often obtain different (often shorter) ATP proofs for the problems exported from such libraries.

[4] Measured on pruned problems containing only the facts needed in the Mizar proofs.

5 MaLARea 0.5 at CADE'24

In 2013, CASC LTB (running for HOL, Isabelle and Mizar categories) also provided pre-training problems and solutions, however they had to be processed withing the overall competition time. The competition problems (250 for Mizar) themselves had to be processed in the *ordered* mode, i.e, without returning to previous unsolved problems. This made the "explore vs. exploit" heuristics in previous versions of MaLARea largely redundant, resulting in a 2013 re-implementation done from scratch which contains several new techniques and so far omits some of the (more complicated) old ones.

Main loop: Instead of deciding which problem to attack next, the system only has to decide how much time it spends in learning/predicting and in proving for each problem. Solving previously unsolved problems might still be beneficial for the learning [6], but with 250 competition problems this did not seem sufficiently rewarding yet. The main loop thus consists only of (parallelized) running of predictors, followed by (parallelized) proving, which is (if successful) followed by proof pseudo-minimization [6], usually producing better data for the learning.

Predictors, features: A family of distance weighted k-nearest neighbor (k-NN) learners complemented by the IDF (inverse document frequency) feature weighting has recently shown very good prediction properties on the Flyspeck dataset [8]. Additionally, a lot of work (measuring distances and finding the nearest neighbors) can be simply shared in such families for different values of k. The final family thus consists of 4 subfamilies run in parallel with different IDF weightings and feature normalization, each of them using 8 values of k, producing 32 k-NN rankings very fast. These rankings can then be differently sliced, easily yielding over 100 premise selections for a given conjecture. An interesting alternative to the IDF feature preprocessing is the latent semantic analysis (LSA), for which the gensim toolkit[5] was used, instructed to produce 400 topics (new features). Such features were used as an input to some k-NNs.

Strategies: The BliStr system has been extended to produce also good SInE strategies for E, and run several times on the Mizar@Turing100 training problems and the published Flyspeck data, producing further overall strengthenings [8].

Global optimization: Several test runs were done on the 2012 Mizar@Turing100 competition data. The large number of premise selections produced during these runs were collected, and used as a pool of problems for finding the strongest combinations of the BliStr strategies with the predictors. The final ensemble consists of 40 combinations characterized by the selection of features, k-NN version, feature weighting, premise count, and ATP strategy.

Evaluation and Competition Performance: The final system solved 260 of the 400 Mizar@Turing100 competition problems (used for optimization here). On the unknown CASC'24 LTB competition data the system solved 239 problems out of 750. The second best system (E 1.8-LTB) solved 135 problems. This is the largest relative distance (77% more) between the first and second system in CASC since Waldmeister's victory in the UEQ division in 2000.

6 Acknowledgements

It has been a great pleasure to use Stephan Schulz's open-source E prover. We now use E as the main ground ATP, E's implementation of SInE, its API for strategy specification, and its code base for our fast shared-term enumerator.

[5] `http://radimrehurek.com/gensim/`

References

[1] Jesse Alama, Tom Heskes, Daniel Kühlwein, Evgeni Tsivtsivadze, and Josef Urban. Premise selection for mathematics by corpus analysis and kernel methods. *J. Autom. Reasoning*, 52(2):191–213, 2014.

[2] Andy Carlson, Chad Cumby, Jeff Rosen, and Dan Roth. The SNoW Learning Architecture. Technical Report UIUCDCS-R-99-2101, UIUC Comp. Sci. Department, 5 1999.

[3] John C Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 148–177, 1979.

[4] Krystof Hoder and Andrei Voronkov. Sine qua non for large theory reasoning. In Nikolaj Bjørner and Viorica Sofronie-Stokkermans, editors, *CADE*, volume 6803 of *LNCS*, pages 299–314. Springer, 2011.

[5] Paul Jaccard. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bull. del la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901.

[6] Cezary Kaliszyk and Josef Urban. Learning-assisted automated reasoning with Flyspeck. *CoRR*, abs/1211.7012, 2012.

[7] Cezary Kaliszyk and Josef Urban. HOL(y)Hammer: Online ATP service for HOL Light. *CoRR*, abs/1309.4962, 2013.

[8] Cezary Kaliszyk and Josef Urban. Stronger automation for Flyspeck by feature weighting and strategy evolution. In Jasmin Christian Blanchette and Josef Urban, editors, *PxTP 2013*, volume 14 of *EPiC Series*, pages 87–95. EasyChair, 2013.

[9] Daniel Kühlwein, Jasmin Christian Blanchette, Cezary Kaliszyk, and Josef Urban. MaSh: Machine learning for Sledgehammer. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *ITP*, volume 7998 of *Lecture Notes in Computer Science*, pages 35–50. Springer, 2013.

[10] Daniel Kühlwein and Josef Urban. Learning from multiple proofs: First experiments. In Pascal Fontaine, Renate A. Schmidt, and Stephan Schulz, editors, *PAAR@IJCAR*, volume 21 of *EPiC Series*, pages 82–94. EasyChair, 2012.

[11] Daniel Kühlwein, Twan van Laarhoven, Evgeni Tsivtsivadze, Josef Urban, and Tom Heskes. Overview and evaluation of premise selection techniques for large theory mathematics. In *IJCAR*, pages 378–392, 2012.

[12] Stephan Schulz. E - A Brainiac Theorem Prover. *AI Commun.*, 15(2-3):111–126, 2002.

[13] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.

[14] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. Cambridge Univ Press, 1998.

[15] Josef Urban. MPTP - Motivation, Implementation, First Experiments. *Journal of Automated Reasoning*, 33(3-4):319–339, 2004.

[16] Josef Urban. MPTP 0.2: Design, implementation, and initial experiments. *J. Autom. Reasoning*, 37(1-2):21–43, 2006.

[17] Josef Urban. MaLARea: a metasystem for automated reasoning in large theories. In Geoff Sutcliffe, Josef Urban, and Stephan Schulz, editors, *ESARLT*, volume 257 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.

[18] Josef Urban. BliStr: The Blind Strategymaker. *CoRR*, abs/1301.2683, 2013.

[19] Josef Urban, Piotr Rudnicki, and Geoff Sutcliffe. ATP and presentation service for Mizar formalizations. *J. Autom. Reasoning*, 50(2):229–241, 2013.

[20] Josef Urban, Geoff Sutcliffe, Petr Pudlák, and Jiří Vyskočil. MaLARea SG1 - Machine Learner for Automated Reasoning with Semantic Guidance. In *IJCAR*, pages 441–456, 2008.

[21] Christoph Weidenbach, Dilyana Dimova, Arnaud Fietzke, Rohit Kumar, Martin Suda, and Patrick Wischnewski. SPASS Version 3.5. In Renate A. Schmidt, editor, *CADE*, volume 5663 of *LNCS*, pages 140–145. Springer, 2009.