Maedmax at School: Learning Selection in Equational Reasoning

Sarah Winkler*

University of Innsbruck
sarah.winkler@uibk.ac.at

Abstract

MædMax is an equational theorem prover based on maximal ordered completion. Like in many automated deduction tools, the selection of equations and inequalities constitutes a critical choice point in the search for a proof. Here we describe the use of random forests to guide selection in two ways: (a) to learn which equations are useful, and (b) to learn a measure for proof progress, which in turn triggers the selection of additional equations.

1 Introduction

The tool MædMax performs equational reasoning by implementing *maximal ordered completion* [8]. As in the given-clause algorithm, selection of facts to process next is a crucial choice point. Here we outline two experiments exploiting machine learning techniques to improve MædMax' selection heuristic. First, random forests were used to learn a measure for the usefulness of equations and inequalities. Second, an estimate of proof progress was learned. Before giving details about these experiments we summarize the main control loop of MædMax.

Maximal completion maintains a pool of equations and inequalities $\mathcal{E}$, split into active and passive items $\mathcal{E}_A$ and $\mathcal{E}_P$. A reduction order giving rise to a terminating rewrite system $\mathcal{R}$ is determined from $\mathcal{E}_A$ by employing a maxSMT call (for instance, by orienting as many equations as possible). If thereby a goal becomes joinable or the system gets ground confluent the procedure succeeds. Otherwise the extended critical pairs of $\mathcal{R}$ and $\mathcal{E}_A$ are added to $\mathcal{E}_P$, a small subset of $\mathcal{E}_P$ is selected into $\mathcal{E}_A$, and the procedure gets reiterated. In MædMax, this selection was so far guided by a straightforward size-age ratio. In addition, a heuristic is used to estimate proof progress turned out to be useful: if progress is assumed to be small, additional old nodes are fed into the active set. For further details the reader is referred to [8].

2 Selection of Equations

We instrumented MædMax to keep track of selections. When a proof is found, it outputs for all selected items a feature vector and a classification as positive or negative, depending on whether it contributed to the proof or not. This vector comprises hand-crafted properties of both the current proof state and the equation itself along with features describing the term structure. For an equation e and a current set of active (in)equalities $\mathcal{E}_A$, the former include the size of $\mathcal{E}_A$, the iteration count, the size, size balance, and age of e, properties of e related to linearity and orientability, plus the number of matches and critical pairs of e on $\mathcal{E}_A$. To capture the term structure pq-grams [1] get computed (where $p = 1$ and $q = 2$). Function symbols were renamed uniformly according to their arity, and we counted occurrences for all 12-grams up to arity 3 (105 features per term).

*Supported by Austrian Science Fund project T789.

In a first experiment, MædMax was run in random selection mode, recording selections as described above. This set was balanced and classified with a random forest of maximum depth 14 and 100 trees, using scikit-learn.[1] Using 5-fold cross-validation, this resulted in a precision of 0.86 and a recall of 0.94 (especially the latter is relevant, since it gives the ratio of useful facts that get positively classified). Term and hand-crafted features contributed 60% and 40% of importance, respectively. Among the latter, fact and state size, number of matches and critical pairs turned out to be most relevant.[2]

We continued this experiment using a reinforcement loop to emulate the way a human might optimize a selection heuristic: we used the obtained classifier as a filter, picking randomly selected facts only if they are positively classified with probability > 0.4. After adding the new proofs' selections to the data set, the classification was repeated and the procedure reiterated. In that way, after four iterations 433 problems get solved within 60s, opposed to 206 beforehand, applying classification to 352000 selections.

Finally, when combining the previously used size-age ratio with the obtained classifier, MædMax solves 613 instead of 606 problems within 60s, with the maximal number of equalities dropping from over 440 to 220 and the maximal number of goals from 1800 to 800. The time spent on selection rises by 1-2% of the overall proof time.

3 Estimating Proof Progress

In MædMax an estimate of the proof progress is used to select additional old facts. The heuristic used so far simply checks whether the cost of the maxSMT call remained unchanged for some iterations. To gain data on proof progress, we implemented a proof track mode: Taking a TSTP proof P as additional input, the tool keeps track of its progress with respect to P, by recording in every iteration features of the prover's current state along with the ratio of facts in P that are already present in $\mathcal{E}_A$ and/or $\mathcal{E}_P$. As features we used 10 properties including iteration count, the size of $\mathcal{E}$, memory used, number of SMT checks, cost of the last maxSMT check, the numbers of facts in $\mathcal{E}_A$ reducible by the last rewrite system $\mathcal{R}$, and critical pairs between $\mathcal{R}$ and $\mathcal{E}_A$.

We ran the tool in proof track mode on all proofs obtained with E and Vampire. This resulted in about 20000 data records of MædMax iterations. We computed the differences of consecutive iterations to learn about changes in the proof state. Random forest classification with 100 trees of a maximum depth 10 resulted in a cross-validated precision and recall of both 0.72. Despite this moderate accuracy, incorporating a decision tree based on the most influential features into MædMax increased the number of solved problems by about 1.5%.

4 Related Work

Although the first efforts in this direction date more than 20 years back, learning from previous proof experience in ATPs is still a field offering many challenges [7]. Here we focus on work about guiding fact selection in ATPs by learning from earlier proofs.

Fuchs [3] employs learning heuristics in the CoDe system to select the clauses. To that end, preference is given to *focus facts* which contributed to earlier proofs, as well as their descendants. He also already aggregated similarity based on some syntactic features to assess new clauses.

[1] See scikit-learn.org.

[2] All experiments use the 897 unsatisfiable TPTP 6.4.0 UEQ problems and were run on Starexec with a timeout of 60s, see http://cl-informatik.uibk.ac.at/users/swinkler/maedmax_at_school for details.

Specifically for the case of purely equational theorem proving, Denzinger and Schulz evaluated two learning-based heuristics to improve selection of equations in the Discount system [2]. First, equations were compiled into patterns to abstract from the signature, and their usefulness recorded for later proof attempts. Second, the recorded equation patterns were arranged in a tree from which based on similarity a measure for the usefulness of all terms could be derived. However, similarity was based on a single string representation.

This approach was carried over for the superposition prover E [6]. Clauses (also abstracted to signature-independent patterns) were recorded with the number of proofs they participates in and their distance to the proof. Problems were classified, by some simple features, and given an input problem the relevant pattern set to guide selection was retrieved based on similarity. In this approach only exactly matching patterns were taken into account.

More recently, Jakubův and Urban proposed feature-based classification of clauses to improve the selection heuristics in saturation-based theorem proving [4]. As features the occurrence counts of *term walks* are used. Term walks resemble *pq*-grams, though the former do not abstract from the signature. The classification model is built with LIBLINEAR. An evaluation on E showed a large increase of performance.

Also a line of experiments with the tableau prover leanCoP investigated guidance of inference algorithms by machine learning techniques; it turned out to significantly improve the relevance heuristics. For instance, in [5] the selection of a clause for the tableau extension is guided by a naive Bayes classification based on features of the current proof state. To this end, the proof state is characterized by the frequency of terms on the active path.

5 Conclusion

In summary, the conducted experiments helped to improve the heuristics of MædMax such that about 2.5% additional problems can be solved, and delivered insights about relevant features. In the future, we plan more thorough reinforcement learning experiments to obtain further data, and will investigate alternative features such as term walks [4].

References

[1] N. Augsten, M. Böhlen, and J. Gamper. The *pq*-gram distance between ordered labeled trees. *ACM Transactions on Database Systems*, 35(1):1–36, 2010.

[2] J. Denzinger and S. Schulz. Automatic acquisition of search control knowledge from multiple proof attempts. *IC*, 162:59–79, 2000.

[3] M. Fuchs. Experiments in the heuristic use of past proof experience. In *Proc. 13th CADE*, volume 1104 of *LNCS*, pages 523–537, 1996.

[4] J. Jakubův and J. Urban. ENIGMA: efficient learning-based inference guiding machine. In *Proc. CICM 2017*, volume 10383 of *LNCS*, pages 292–302, 2017.

[5] C. Kaliszyk and J. Urban. FEMaLeCoP: Fairly efficient machine learning connection prover. In *Proc. LPAR 2015*, volume 9450 of *LNCS*, pages 88–96, 2015.

[6] S. Schulz. Learning search control knowledge for equational theorem proving. In *Proc. KI 2001*, volume 2174 of *LNCS*, pages 320–334, 2001.

[7] S. Schulz. We know (nearly) nothing! But can we learn? In *Proc. ARCADE 2017*, volume 51 of *EPiC*, pages 29–32, 2017.

[8] S. Winkler and G. Moser. MædMax: A maximal ordered completion tool. In *Proc. 9th IJCAR*, volume 10900 of *LNCS*, pages 472–480, 2018.