Machine Learning for Instance Selection in SMT Solving

Jasmin Christian Blanchette[1,2], Daniel El Ouraoui[2],
Pascal Fontaine[2], and Cezary Kaliszyk[3]

[1] Vrije Universiteit Amsterdam, Amsterdam, the Netherlands
j.c.blanchette@inria.fr
[2] University of Lorraine, CNRS, Inria, and LORIA, Nancy, France
{jasmin.blanchette,daniel.el-ouraoui,pascal.fontaine}@inria.fr
[3] University of Innsbruck, Innsbruck, Austria
cezary.kaliszyk@uibk.ac.at

Abstract

SMT solvers are among the most suited tools for quantifier-free first-order problems, and their support for quantified formulas has been improving in recent years. To instantiate quantifiers, they rely on heuristic techniques that generate thousands of instances, most of them useless. We propose to apply state-of-the-art machine learning techniques as classifiers for instances on top of the instantiation process. We show that such techniques can indeed decrease the number of generated useless instances. We envision that this could lead to more efficient SMT solving for quantified problems.

Satisfiability-modulo-theories (SMT) solvers are among the best backends for verification tools and "hammers" in proof assistants. When proof obligations contain quantified formulas, SMT solvers rely on *instantiation*, replacing quantified subformulas by sets of ground instances. Three main techniques have been designed: enumerative [11], trigger-based [4], and conflict-based [12] instantiation. Among these, only conflict-based instantiation computes instances that are guaranteed to be relevant, but it is incomplete and is normally used in combination with other techniques. Enumerative and trigger-based techniques are highly heuristic and generate a large number of instances, most of them useless. As a result, the search space of the solver explodes. Reducing the number of instances could improve the solver's efficiency and success rate within a given time limit.

We propose to use a state-of-the-art machine learning algorithm as a predictor over the generated set of instances to filter out irrelevant instances, and thus decrease the number of instances given to the ground solver. The predictor is invoked after each instantiation round to rate the potential usefulness of each generated instance. Several strategies are then used to build a subset of potentially relevant instances that are immediately added to the ground solver. Adding the other instances is postponed.

We conducted our experiment in veriT [2], an SMT solver that implements all three instantiation techniques described above. We chose as predictor the XGBoost gradient boosting toolkit [3] with the binary classification objective. This configuration had already been used successfully in the context of theorem proving [6, 10].

Choosing a suitable set of features is crucial for effective machine learning. The features determine how precise the representation of the problem is. Previous works already investigate features for theorem proving [1, 5, 6, 8–10]. Our features are more specifically inspired by ENIGMA [6] and RLCoP [7]. They are basically term symbols and term walks with symbol sequences projected to features using Vowpal Wabbit hashing. Term variables and Skolem constants are translated analogously to constants. The model is further enriched with abstract features such as term size, term depth, and the number of instances.

To encode our problem into sparse vectors, we use three kinds of information available to the solver: the ground part of the formula (set of literals $l_1, \ldots, l_m$), the quantified formula

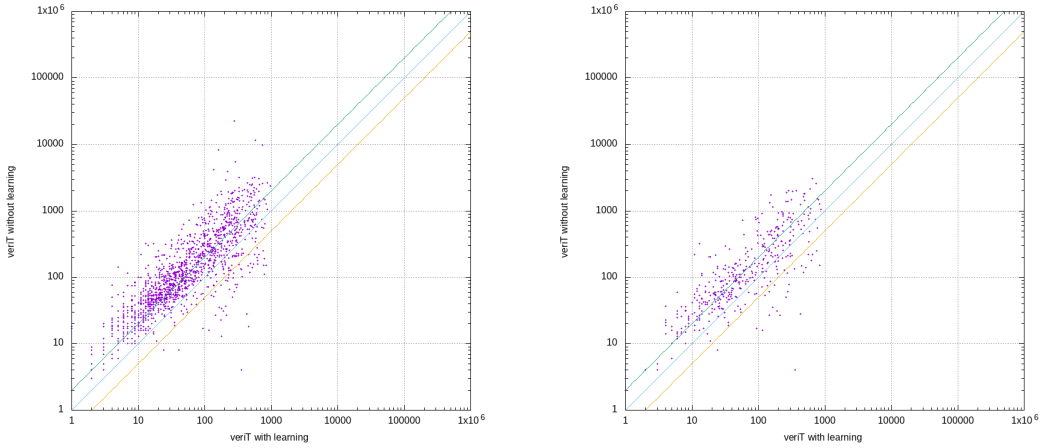

Figure 1: Comparison of veriT configurations on UF SMT-LIB benchmarks

$\psi[\overline{x_n}]$ to instantiate, and the substitution. A query to the predictor is represented by a tuple of the form

$$(\; l_1, \ldots, l_m \; , \; \psi[\overline{x_n}] \; , \; x_1 \mapsto t_1, \ldots, x_n \mapsto t_n \;)$$

In the learning phase, to discriminate useful instances from others in the run of the solver, we consider the pruned proof with only the relevant facts. An instance produced in the run is tagged as useful if it also occurs in the pruned proof. To train the predictor, we used the SMT-LIB benchmarks in the UF category. We ran veriT on 1866 SMT formulas randomly divided into two sets: 560 for the test set (30%) and 1306 for the training set (70%). One run of the solver generates many more useless instances (99%) than useful ones (1%). Consequently, we used undersampling to obtain a balanced data set for learning.

Figure 1 compares the numbers of generated instances for two versions of the veriT solver: with instance selection (x-axis) and without (y-axis). The fewer instances, the better. The left-hand side shows the results of the solvers on the entire data set (training and test), whereas the right-hand side shows the results only on the test set. In both plots, a cluster of points is forming along the line corresponding to the equation $f(x) = 2x$. On average, instance selection allows veriT to find proofs with about half the number of instances. On the other hand, a time comparison would not look good for instance selection, because the current prototype relies on extremely time consuming external calls to the predictor.

These results show that our approach seems to be suitable to substantially reduce the number of generated instances. The implementation of this first prototype is suboptimal. We are now working on an implementation of the predictor inside veriT, hoping to make the prediction cost close to negligible. We hope that the good results in terms of number of instances required will eventually translate into strong performance.

Acknowledgement The work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 713999, Matryoshka). Experiments were carried out using the Grid'5000 testbed (https://www.grid5000.fr/), supported by a scientific interest group hosted by Inria and including CNRS, RENATER, and several universities as well as other organizations.

References

[1] Miltiadis Allamanis, Pankajan Chanthirasegaran, Pushmeet Kohli, and Charles Sutton. Learning continuous semantic representations of symbolic expressions. In Doina Precup and Yee Whye Teh, editors, *ICML 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 80–88. PMLR, 2017.

[2] Thomas Bouton, Diego Caminha B. de Oliveira, David Déharbe, and Pascal Fontaine. veriT: an open, trustable and efficient SMT-solver. In Renate A. Schmidt, editor, *CADE–22*, volume 5663 of *LNCS*, pages 151–156. Springer, 2009.

[3] Tianqi Chen and Carlos Guestrin. XGBoost: a scalable tree boosting system. In Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi, editors, *KDD 2016*, pages 785–794. ACM, 2016.

[4] Leonardo de Moura and Nikolaj Bjørner. Efficient E-matching for SMT solvers. In Frank Pfenning, editor, *CADE–21*, volume 4603 of *LNCS*, pages 183–198. Springer, 2007.

[5] Geoffrey Irving, Christian Szegedy, Alexander A. Alemi, Niklas Een, François Chollet, and Josef Urban. DeepMath–deep sequence models for premise selection. In Daniel D. Lee, Masashi Sugiyama, Ulrike V. Luxburg, Isabelle Guyon, and Roman Garnett, editors, *NIPS 2016*, pages 2235–2243. Curran Associates, 2016.

[6] Jan Jakubuv and Josef Urban. ENIGMA: efficient learning–based inference guiding machine. In Herman Geuvers, Matthew England, Osman Hasan, Florian Rabe, and Olaf Teschke, editors, *CICM 2017*, volume 10383 of *LNCS*, pages 292–302. Springer, 2017.

[7] Cezary Kaliszyk, Josef Urban, Henryk Michalewski, and Mirek Olšák. Reinforcement learning of theorem proving. In Samy Bengio, Hanna Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *NeurIPS 2018*, pages 8835–8846. Curran Associates, 2018.

[8] Cezary Kaliszyk, Josef Urban, and Jiri Vyskočil. Efficient semantic features for automated reasoning over large theories. In Qiang Yang and Michael Wooldridge, editors, *IJCAI 2015*, pages 3084–3090. AAAI Press, 2015.

[9] Sarah Loos, Geoffrey Irving, Christian Szegedy, and Cezary Kaliszyk. Deep network guided proof search. In Thomas Eiter and David Sands, editors, *LPAR-21*, volume 46 of *EPiC Series in Computing*, pages 85–105. EasyChair, 2017.

[10] Bartosz Piotrowski and Josef Urban. ATPboost: Learning premise selection in binary setting with ATP feedback. In Didier Galmiche, Stephan Schulz, and Roberto Sebastiani, editors, *IJCAR 2018*, volume 10900 of *LNCS*, pages 566–574. Springer, 2018.

[11] Andrew Reynolds, Haniel Barbosa, and Pascal Fontaine. Revisiting enumerative instantiation. In Dirk Beyer and Marieke Huisman, editors, *TACAS 2018*, volume 10806 of *LNCS*, pages 112–131. Springer, 2018.

[12] Andrew Reynolds, Cesare Tinelli, and Leonardo de Moura. Finding conflicting instances of quantified formulas in SMT. In Koen Claessen and Viktor Kuncak, editors, *FMCAD 2014*, pages 195–202. IEEE, 2014.