Selecting Quantifiers for Instantiation in SMT *

Mikoláš Janota[1], Jelle Piepenbrock[1,2], and Bartosz Piotrowski[1,3]

[1] Czech Technical University, Prague
[2] Radboud University Nijmegen, The Netherlands
[3] University of Warsaw, Poland

Introduction In satisfiability modulo theories (SMT), quantifiers are treated opaquely: they are seen as sources of possible instantiations. A quantified sub-formula $\forall \vec{x} \phi$ is abstracted as a Boolean variable Q and instantiations are registered in the form of implications $Q \Rightarrow \phi[\vec{x} \mapsto \vec{t}]$ for some ground terms $\vec{t}$. Consider the following toy formula:

$$f(3) < 0 \wedge ((\forall x f(x) > 0) \vee (\forall x f(x) > x)),$$

where $f\colon \text{int} \to \text{int}$. To obtain a refutation, we abstract and instantiate as follows:

abstraction:	$Q_1 \equiv (\forall x f(x) > 0), Q_2 \equiv (\forall x f(x) > x)$
abstracted formula:	$f(3) < 0 \wedge (Q_1 \vee Q_2)$
instantiation 1:	$Q_1 \Rightarrow f(3) > 0$
instantiation 2:	$Q_2 \Rightarrow f(3) > 3$

Most approaches instantiate quantifiers *gradually*, meaning, the new instantiations are added after testing that the old ones do not already yield a contradiction in the ground solver. In this work, we propose to use machine learning to decide which quantifiers should be instantiated during solving. Figure 1 schematically illustrates the considered setup.

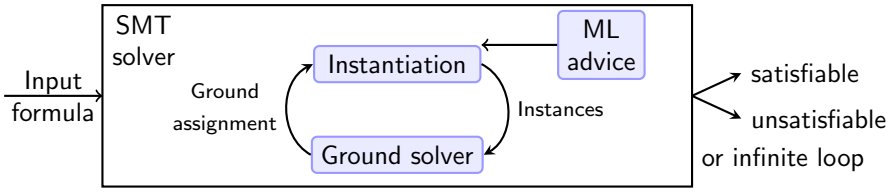

Figure 1: Schematic of the SMT solver with ML guidance for quantifier instantiation.

There are a number of methods for devising new instantiations [2, 3]. Here we consider a simple setup where instantiation terms are taken from the set of terms present in the current ground formula—this includes the instantiations already performed and therefore this set grows throughout the life of the solver. Without theories, the set of candidate ground terms grows towards the Herbrand universe. This approach is referred to as *enumerative instantiation* [8, 5].

In standard setting, the SMT solver would instantiate *all* quantifiers that are true in the current ground model. Here, we propose to instantiate only a subset, based on a pre-trained ML-predictor. Selecting the right quantifier for instantiation puts less burden on the ground solver but also facilitates the search for the right ground terms to be used for instantiations.

*The results were supported by the Ministry of Education, Youth and Sports within the dedicated program ERC CZ under the project POSTMAN no. LL1902.

Approach Consider quantified sub-formulas $Q_1, \ldots, Q_n$ true in the current model of the ground solver. Consider some pre-trained ML-predictor $\mathcal{V}$ from formulas to the interval $[0, 1]$, where $\mathcal{V}(Q) = 1$ means that the predictor believes that Q is important for the solution of the whole SMT problem. A straightforward approach would be to instantiate Q with the probability $V(Q)$. This brings about the risk of instantiating too seldom if the ML predictor has low overall confidence. Therefore, we rescale to $\max \mathcal{V}(Q_i)$, i.e., the quantifier $\text{argmax}_i \mathcal{V}(Q_i)$ is instantiated with probability 1. But even with this policy, we are running the risk that a bad judgment of the ML prediction will cause some important quantifiers never to be instantiated. To that end, with probability $\epsilon \in [0, 1]$ a quantifier is always instantiated. In essence, this corresponds to the well-known ϵ-greedy policy [9] and a quantifier Q is instantiated with probability:

$$\epsilon + (1 - \epsilon) \frac{\mathcal{V}(Q)}{\max_i \mathcal{V}(Q_i)}.$$

Implementation and experiments The approach was implemented in the SMT-solver cvc5 [1] with LightGBM [6] as the ML-predictor. Currently, the ML-predictor only gets as features the bag of words representation of the quantified sub-formula, i.e., ignoring the overall context. For evaluation we use SMT-LIB problems from the UFLIA/sledgehammer category, while filtering out problems solvable without instantiation. Looping-style evaluation is run (similar to [4, 7, 10]). In the first iteration, an unguided SMT-solver is run on the benchmark and data extracted from the solved problems is used to train the ML model. Then, the model is used to guide the solver in the next iteration of solving the benchmark. The success rate is recorded and examples extracted from the newly solved problems are added to the training set. This solving-training procedure is repeated 6 times with time limit 60 s for the solver. Figure 2 shows results for $\epsilon \in \{0, 0.1, 0.5\}$ and a version with no ML-guidance.

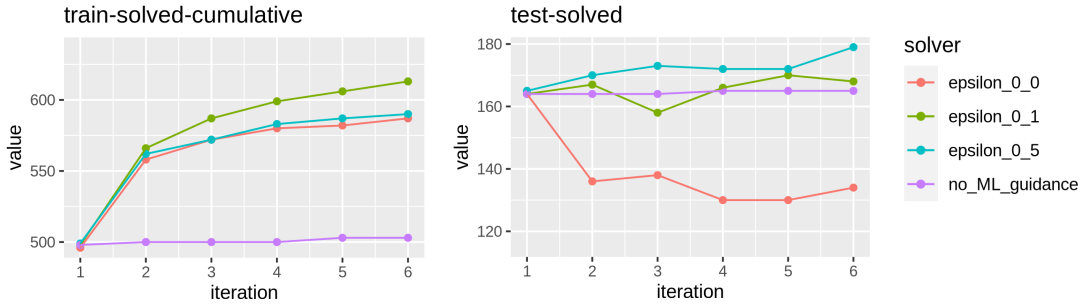

Figure 2: Looping evaluation on UFLIA/sledgehammer dataset. Left: number of training problems solved cumulatively. Right: number of testing problems solved in individual iterations.

Conclusions and future work The paper shows that that ML guidance can effectively guide quantifier instantiation in SMT. The evaluation points to a significant impact of learning on the number of solved instances but we only observe improvements on the training set, not on testing. This suggests that there is some version of over-fitting, even though not in the standard sense because the training examples constantly change. We plan to improve featurization by including the context of quantifiers (looking at the whole SMT problem) and using more advanced featurization approaches such as graph neural networks.

References

[1] Haniel Barbosa, Clark W. Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Andrew Reynolds, Ying Sheng, Cesare Tinelli, and Yoni Zohar. cvc5: A versatile and industrial-strength SMT solver. In Dana Fisman and Grigore Rosu, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS*, volume 13243 of *Lecture Notes in Computer Science*, pages 415–442. Springer, 2022.

[2] David Detlefs, Greg Nelson, and James B. Saxe. Simplify: A theorem prover for program checking. *J. ACM*, 52(3):365–473, 2005.

[3] Yeting Ge and Leonardo Mendonça de Moura. Complete instantiation for quantified formulas in satisfiability modulo theories. In *Computer Aided Verification, 21st International Conference, CAV*, pages 306–320, 2009.

[4] Mikoláš Janota, Jelle Piepenbrock, and Bartosz Piotrowski. Towards learning quantifier instantiation in SMT. In *Theory and Applications of Satisfiability Testing – SAT 2022*. LIPIcs, 2022.

[5] Mikoláš Janota, Haniel Barbosa, Pascal Fontaine, and Andrew Reynolds. Fair and adventurous enumeration of quantifier instantiations. In *Formal Methods in Computer-Aided Design*, 2021.

[6] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[7] Bartosz Piotrowski and Josef Urban. ATPboost: Learning premise selection in binary setting with ATP feedback. In Didier Galmiche, Stephan Schulz, and Roberto Sebastiani, editors, *Automated Reasoning - 9th International Joint Conference, IJCAR 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings*, volume 10900 of *Lecture Notes in Computer Science*, pages 566–574. Springer, 2018.

[8] Andrew Reynolds, Haniel Barbosa, and Pascal Fontaine. Revisiting enumerative instantiation. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 10806, pages 112–131, 2018.

[9] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning*. Adaptive Computation and Machine Learning series. Bradford Books, Cambridge, MA, 2 edition, November 2018.

[10] Josef Urban, Geoff Sutcliffe, Petr Pudlák, and Jirí 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.