Project Proposal: Machine Learning for Model-Based Quantifier Instantiation *

Marek Dančo, Petra Hozzová, and Mikoláš Janota

Czech Technical University in Prague, Prague, Czechia

1 Introduction

Formulas involving quantifiers are typically undecidable and therefore represent a challenge for automated reasoning in general. Satisfiability Modulo Theories (SMT) solvers tackle quantifiers by instantiating quantified variables by ground terms. A plethora of techniques exists that attempt to find instantiations that will most likely lead to a contradiction, such as *e-matching* [2], *syntax-guided* [7], *conflict-based* [9], or *enumerative instantiation* [5, 8].

A method that stands out is *model-based quantifier instantiation (MBQI)* [4] because it is semantically driven, enables proving satisfiable results, and is complete under certain conditions. This project will focus on the improvement of MBQI by developing guidance by ML.

2 Preliminaries

General familiarity with logic and SMT is assumed [1]. A brief overview of MBQI follows. For simplicity, assume a closed formula with a single quantifier over the integers/reals of the following form:

$$G \wedge \forall x.\, F \quad (G \text{ is ground}) \tag{1}$$

Let $\mathcal{M}$ be a model of G that can be expressed in the underlying theory, e.g. $c = 3, f = \lambda x.2x + 1$, and let $F_{\mathcal{M}}$ be F with $\mathcal{M}$ evaluated in it, i.e. with each symbol in F interpreted by $\mathcal{M}$ replaced by its interpretation. Note that the formula $F_{\mathcal{M}}$ is quantifier-free and contains only the free variable x and interpreted functions and predicates. Next, check the satisfiability of the following formula:

$$\neg F_{\mathcal{M}} \tag{2}$$

If formula (2) is *unsatisfiable*, then the original formula (1) is true. Further, we know that $\mathcal{M}$ is a model of (1) and the algorithm terminates. If formula (2) is *satisfiable*, then the value v of x gives a *counterexample* to the model $\mathcal{M}$. Select a ground term t, s.t. $\mathcal{M}(t) = v$, instantiate with it, and repeat the process with this instantiation, i.e., repeat with the following formula:

$$F[x \mapsto t] \wedge G \wedge \forall x.\, F \tag{3}$$

This idea naturally generalizes to a vector of variables $\bar{x}$ and to conjunctions of formulas (e.g., clauses).

*Supported by the Czech MEYS under the ERC CZ project no. LL1902 *POSTMAN*, by the European Union under the project *ROBOPROX* (reg. no. CZ.02.01.01/00/22_008/0004590).

3 Project Plan

The objective is to use machine learning to select the terms to be instantiated with. The project will be carried out in the following steps.

1. Implement a basic implementation of MBQI in Python, to facilitate experiments.
   (a) The PySMT interface [3] enables taking advantage of various SMT solvers, depending on the theory at hand.
   (b) Initially we will focus on arithmetic logics (UF-{L,N}{I,R}A), which are easier to work with, as there is an immediate translation from models to the logical representation. Namely, each rational number already exists in the language of SMT, allowing us to represent the values of constants as well as function interpretations from the model by terms.
2. Select ground terms based on hand-crafted features and simple ML models, such as SVM or logistic regression.
   (a) For a model $\mathcal{M}$ and the set of ground terms $\mathcal{T}$ of (1), consider the set of terms $\mathcal{T}_v \subseteq \mathcal{T}$ with the value v in $\mathcal{M}$ (i.e., $\mathcal{M}(t) = v$ for all $t \in \mathcal{T}_v$). For each quantified variable, we can use a pre-trained ML-model to select a term $t \in \mathcal{T}_v$ based on characteristics of the term, such as its depths, size, or age.
   (b) Training is done on a set of problems and evaluated on a holdout set. Positive examples are extracted from successfully solved formulas by looking at the instantiations that contributed to the final unsat proof. In the case of satisfiable instance, the training data is yet to be determined.
3. Just as bullet 2, but consider more complicated features: e.g. bag-of-words—similarly as done by Janota et al. for enumerative instantiation [6].
4. Evaluate our techniques on SMT-LIB problems.[1]

Possible future directions involve synthesizing new terms, rather than just picking terms from the existing set of ground terms, and extending the approach to theories beyond arithmetic.

References

[1] Clark W. Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability modulo theories. In *Handbook of Satisfiability*. IOS Press, 2009. URL: http://dx.doi.org/10.3233/978-1-58603-929-5-825, doi:10.3233/978-1-58603-929-5-825.

[2] David Detlefs, Greg Nelson, and James B. Saxe. Simplify: A theorem prover for program checking. *J. ACM*, 52(3):365–473, 2005. doi:10.1145/1066100.1066102.

[3] Marco Gario and Andrea Micheli. PySMT: a solver-agnostic library for fast prototyping of smt-based algorithms. In *SMT Workshop 2015*, 2015.

[4] Yeting Ge and Leonardo Mendonça de Moura. Complete instantiation for quantified formulas in satisfiabiliby modulo theories. In *Computer Aided Verification CAV*, volume 5643, pages 306–320. Springer, 2009. doi:10.1007/978-3-642-02658-4_25.

[1] https://smt-lib.org/

[5] Mikoláš Janota, Haniel Barbosa, Pascal Fontaine, and Andrew Reynolds. Fair and adventurous enumeration of quantifier instantiations. In *Formal Methods in Computer-Aided Design*, pages 256–260. IEEE, 2021. doi:10.34727/2021/ISBN.978-3-85448-046-4_35.

[6] Mikoláš Janota, Jelle Piepenbrock, and Bartosz Piotrowski. Towards Learning Quantifier Instantiation in SMT. In Kuldeep S. Meel and Ofer Strichman, editors, *25th International Conference on Theory and Applications of Satisfiability Testing (SAT 2022)*, volume 236 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:18, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.SAT.2022.7, doi:10.4230/LIPIcs.SAT.2022.7.

[7] Aina Niemetz, Mathias Preiner, Andrew Reynolds, Clark W. Barrett, and Cesare Tinelli. Syntax-guided quantifier instantiation. In *Tools and Algorithms for the Construction and Analysis of Systems, TACAS*. Springer, 2021. doi:10.1007/978-3-030-72013-1_8.

[8] Andrew Reynolds, Haniel Barbosa, and Pascal Fontaine. Revisiting enumerative instantiation. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*. Springer, 2018. doi:10.1007/978-3-319-89963-3_7.

[9] Andrew Reynolds, Cesare Tinelli, and Leonardo Mendonça de Moura. Finding conflicting instances of quantified formulas in SMT. In *Formal Methods in Computer-Aided Design, FMCAD*, pages 195–202. IEEE, 2014. doi:10.1109/FMCAD.2014.6987613.