# Machine Learning for Quantifier Selection in cvc5*

Jan Jakubův[1], Mikoláš Janota[1], Jelle Piepenbrock[1,2], and Josef Urban[1]

[1] Czech Technical University in Prague, Prague, Czech Republic
[2] Radboud University, Nijmegen, Netherlands
jakubuv@gmail.com

## 1 Quantified Formulas and SMT Solvers

In this work we considerably improve the real-time performance of state-of-the-art satisfiability modulo theories (SMT) solving on first-order quantified problems by efficient machine learning guidance of quantifier selection. Quantifiers represent a significant challenge for SMT and are technically a source of undecidability. In our approach, we train an efficient machine learning model that informs the solver which quantifiers should be instantiated and which not. Each quantifier may be instantiated multiple times and the set of the currently active quantifiers changes as the solving progresses. Therefore, we invoke the machine learning (ML) predictor many times, during the whole run of the solver. To make this efficient, we use fast ML models based on gradient boosted decision trees. We integrate our approach into the state-of-the-art cvc5 SMT solver [2] and show a considerable increase of the system's holdout-set performance after training it on large sets of first-order problems. The method is tested in several ways, using both single-strategy and portfolio approaches. The evaluation is done on two large formal verification corpora: first-order problems created from the Mizar Mathematical Library, and first-order problems created from the HOL4 standard library.

Our methods improve upon a related method of *offline premise selection* [1, 8], where initial quantified assumptions are filtered *only once* before launching the solver. Such filtering can result in an irrecoverable mistake (a *deletion abstraction* [14] resulting in a too weak theory) when a necessary assumption is deleted, whereas the method presented here ensures that every quantified formula will be considered for instantiations with a non-zero probability. This yields a more complex, probabilistically guided framework implementing deletion and instantiation abstractions in the framework proposed in [14]. As we employ a highly effective version of *gradient boosting decision trees* with efficiently computable *symbol-based features*, our implementation produces only a minimal overhead over the standard cvc5 run.

Formulas with *with* quantifiers represent a significant challenge for SMT. In general, SMT solvers use *instantiations*—unless they deal with decidable quantified theories [5, 3, 15]. Instantiations are done with the goal of achieving a contradiction. This style of reasoning can be seen as a direct application of the Herbrand's theorem. For example, for $(\forall x : \mathbb{R}. x > 0)$ instantiating $x$ with the value 0 yields $0 > 0$, which immediately gives a contradiction (in $\mathbb{R}$).

The solving process alternates between a *ground solver* and an *instantiation module*, where the ground solver perceives quantifiers as opaque propositions. After identifying a model for the ground part, control shifts to the *instantiation module*. This module generates new instances of the quantified sub-formulas that are currently meant to hold. A new instance is added to the ground part of the formula, thus making it stronger. The process stops if the ground part becomes unsatisfiable, if ever (model-based quantifier instantiation can also lead to satisfiable answers [6]).

*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), by the Czech Science Foundation grants no. 25-17929X and 24-12759S, and Amazon Research Awards.

## 2 Machine learning for instantiation guidance

The instantiation methods supported by cvc5 are implemented through various *instantiation modules* that share a common interface. An instantiation module is invoked via its *check* method to refine information about quantified formulas by introducing appropriate formula instances. Each module is provided with information about currently asserted quantified formulas, and it selects formulas and generates their instances in accordance with the implemented instantiation method. Subsequently, control is returned to the ground solver.

Effective quantifier and instance selection can significantly enhance performance, and different instantiation methods offer grounds for various possible applications of machine learning methods at a method-specific level [11, 12]. However, we propose a generic method for *quantifier selection* by limiting the formulas visible to the modules. As all instantiation modules iterate over available quantified formulas and process them one by one, we can seamlessly integrate a quantifier selector into any module and simply skip the processing of undesirable quantifiers. To predict the quality of quantifiers, we represent quantified formulas by feature vectors similar to ENIGMA features [10, 9], and we utilize an efficient implementation of *decision tree ensembles* (LightGBM [13]) that enables easy and fast integration with cvc5. Decision tree models can be trained to classify quantified formulas as *positive* or *negative* based on provided training examples. The trained model can be employed within an instantiation module to skip the processing of negative quantifiers.

The two most prominent modules for quantifier selection in cvc5 are *enumerative instantiation* and *e-matching*. We implement our machine learning guidance for them, as well as for the modules relying on *conflict-based quantifier instantiation* and *finite model finding*.

Next, we conduct a straightforward and reproducible experiment to construct an ML-enhanced portfolio of strategies and compare its performance with state-of-the-art cvc5 portfolio. We use two different benchmark datasets for our experiments: Mizar's MPTP [16, 17, 18] and HOL4's GRUNGE [4]. Both datasets originate from translations of large mathematical libraries into first-order logic, specifically into the UF theory in the context of SMTs. Thanks to the versatility of our approach, we could integrate it with various instantiation strategies, thereby enhancing multiple solver configurations and achieving a notable 22.46% **improvement over the baseline state-of-the-art portfolio on Mizar and** 12.86% **improvement on GRUNGE**.

Our methods also exhibit remarkable qualities of cross-strategy *model transfer*, where a model trained on samples from one strategy improves the performance of other strategies. This is not always guaranteed in practice. For instance, in the context of ATPs, where machine learning is used to guide *given clause selection* [10, 9], such transfer typically does not occur [8]. A key reason for this difference lies in the syntactic forms of formulas encountered during proof search. In ATPs, different *term orderings* lead to distinct normal forms of first-order terms, meaning that strategies operate on syntactically different formulas, making model transfer challenging. However, in the context of quantifier selection for SMTs, all strategies process syntactically equivalent formula representations, which enables effective knowledge transfer.

The above allowed us to improve the state-of-the-art performance of cvc5 on Mizar by more than 20% in both the *single strategy* and *portfolio* scenarios. Our best machine-learned strategy, alone, outperforms the state-of-the-art cvc5's portfolio from the latest CASC competition by more than 10%. Our best strategy solves 1,810 problems from the Mizar benchmark in 30 seconds. This number can be informatively compared with experiments on the same dataset from the literature [7], where the state-of-the-art ATP prover E with advanced machine learning methods, solves only 1,632 of the holdout problems in 30 seconds.

# 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] 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 *TACAS (1)*, volume 13243. Springer, 2022.

[3] Nikolaj S. Bjørner and Mikoláš Janota. Playing with quantified satisfaction. In *LPAR*, pages 15–27. EasyChair, 2015.

[4] Chad E. Brown, Thibault Gauthier, Cezary Kaliszyk, Geoff Sutcliffe, and Josef Urban. GRUNGE: A grand unified ATP challenge. In *CADE*, volume 11716 of *Lecture Notes in Computer Science*, pages 123–141. Springer, 2019.

[5] James Harold Davenport, Yvon Siret, and Evelyne Tournier. *Computer algebra - systems and algorithms for algebraic computation (2. ed.)*. Academic Press, 1993.

[6] 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.

[7] Zarathustra Amadeus Goertzel, Karel Chvalovský, Jan Jakubův, Miroslav Olsák, and Josef Urban. Fast and slow Enigmas and parental guidance. In *FroCoS*, volume 12941 of *Lecture Notes in Computer Science*, pages 173–191. Springer, 2021.

[8] Zarathustra Amadeus Goertzel, Jan Jakubův, Cezary Kaliszyk, Miroslav Olsák, Jelle Piepenbrock, and Josef Urban. The Isabelle ENIGMA. In *ITP*, volume 237 of *LIPIcs*, pages 16:1–16:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[9] Jan Jakubův, Karel Chvalovský, Miroslav Olsák, Bartosz Piotrowski, Martin Suda, and Josef Urban. ENIGMA anonymous: Symbol-independent inference guiding machine (system description). In *IJCAR (2)*, volume 12167 of *Lecture Notes in Computer Science*, pages 448–463. Springer, 2020.

[10] Jan Jakubův and Josef Urban. Enhancing ENIGMA given clause guidance. In Florian Rabe, William M. Farmer, Grant O. Passmore, and Abdou Youssef, editors, *Intelligent Computer Mathematics - 11th International Conference, CICM 2018, Hagenberg, Austria, August 13-17, 2018, Proceedings*, volume 11006 of *Lecture Notes in Computer Science*, pages 118–124. Springer, 2018.

[11] Jan Jakubův, Mikoláš Janota, Bartosz Piotrowski, Jelle Piepenbrock, and Andrew Reynolds. Selecting quantifiers for instantiation in SMT. In Stéphane Graham-Lengrand and Mathias Preiner, editors, *Proceedings of the 21st International Workshop on Satisfiability Modulo Theories (SMT 2023) co-located with the 29th International Conference on Automated Deduction (CADE 2023), Rome, Italy, July, 5-6, 2023*, volume 3429 of *CEUR Workshop Proceedings*, pages 71–77. CEUR-WS.org, 2023.

[12] 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, August 2-5, 2022, Haifa, Israel*, volume 236 of *LIPIcs*, pages 7:1–7:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[13] 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 *NIPS*, pages 3146–3154, 2017.

[14] Julio César López-Hernández and Konstantin Korovin. An abstraction-refinement framework for reasoning with large theories. In *IJCAR*, volume 10900 of *Lecture Notes in Computer Science*, pages 663–679. Springer, 2018.

[15] Andrew Reynolds, Tim King, and Viktor Kuncak. Solving quantified linear arithmetic by counterexample-guided instantiation. *Formal Methods Syst. Des.*, 51(3):500–532, 2017.

[16] J. Urban. Translating Mizar for First Order Theorem Provers. In A. Asperti, B. Buchberger, and J.H. Davenport, editors, *Proceedings of the 2nd International Conference on Mathematical Knowledge Management*, number 2594 in LNCS, pages 203–215. Springer, 2003.

[17] Josef Urban. MPTP – Motivation, Implementation, First Experiments. *J. Autom. Reasoning*, 33(3-4):319–339, 2004.

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