A Benchmark for Infinite Models in SMT [*]

Mikoláš Janota[1], Chad E. Brown[1], and Cezary Kaliszyk[2]

[1] Czech Technical University, Prague
[2] University of Innsbruck, Innsbruck

Introduction *Satisfiability modulo theories* (SMT) [1] requires model finding for formulas. This task is conceptually easy for quantifier-free theories but is extremely difficult in the presence of quantifiers. Indeed, in the theory of linear real arithmetic, one can easily see that for the formula $2c = 1$, setting $c = 0.5$ is a model, and, the same formula has no model in the theory of integer arithmetic. However, the formula $(\forall x)(fx > x)$ requires constructing the function f (e.g. $fx = x + 1$). Unlike proof-finding, model-finding is not even semi-decidable.

There is a long-standing tradition of *finite* model finders [6, 3], which are implemented by search on models of increasing, but finite, size. In the case of *infinite* models, the search space is extremely unwieldy and we expect that machine learning methods will be able to help there.

In a recent evaluation Parsert et al. evaluate the capabilities of several existing tools that enable infinite model finding [7]. This evaluation shows clear limitations of the existing tools but it does not provide a dataset on which we could evaluate or train statistical approaches. In this ongoing work, we aim to create a benchmark of formulas with infinite models.

Approach In practice, users generate *many* satisfiable problems. Indeed, any incorrectly specified or implemented program leads to a satisfiable SMT (in TPTP terminology *counter-satisfiable*). Unfortunately, these intermediate, incorrect formulas are typically not made available. Consequently, most formulas in the SMT-LIB are unsatisfiable. Further, it appears that most satisfiable formulas are completely beyond the capabilities of the current technology. Hence, we seek inspiration in the available SMT formula and look at their fragments.

An input SMT formula ϕ is seen as a conjunction of assertions (as provided by the user). A *fragment* of ϕ is a conjunction of some of these assertions (equivalent to the original formula if all the assertions are considered). We hope that solving such fragments, will help us to further the research area and eventually enable us to solve the original problems.

The approach we take here is to consider some parameter k and extract fragments that contain k uninterpreted functions. Given an SMT formula, we choose k uninterpreted function symbols that appear in the formula and filter out the conjuncts that are only weakly related to these k symbols.

Let us describe the process for $k = 2$. Consider an SMT formula ϕ and two uninterpreted function symbols f and g that occur in ϕ. Let ψ be an assertion of the form $(\forall \overline{x})\psi'$ where $\overline{x}$ is an arbitrary set of variables. We will say that the assertion ψ is an *f, g assertion of ϕ* if it contains at least one f or g and no other uninterpreted functions; there is no limitation on constants. The *f, g fragment of ϕ* is defined as the conjunction of all the f, g assertions in ϕ. We stress that in the current implementation we only consider assertions that are denoted by the user; this could be relaxed to arbitrary subformulas. The generated benchmark can be downloaded from the author's website.[1]

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

[1]https://people.ciirc.cvut.cz/~janotmik/aitp23/

Example A fragment extracted from the grasshopper family containing the membership predicate (in) and a unit set constructor (setenum), together with the constant for the empty set.

```
(set-logic UFLIA)
(declare-sort SetInt 0)
(declare-fun emptyset () SetInt)
(declare-fun in (Int SetInt) Bool)
(declare-fun setenum (Int) SetInt)
(assert (forall ((x Int) (y Int)) (= (in x (setenum y)) (= x y))))
(assert (forall ((x Int)) (not (in x emptyset))))
```

Experiments As an initial test set we use the SMT-LIB family UFNIA. These are formulas that contain uninterpreted functions (UF) and nonlinear integer arithmetic (NIA). The family contains 13,463 SMT formulas.

Extracting the global 2-fragments for all the formulas and all functions f, g appearing in them produces 19,566 non-duplicate fragments. Each fragment can be considered as a standalone SMT problem.

The SMT solver Z3 [4] can solve 15,626 out of these, most of them are trivial. This leaves 3,938 challenge benchmark problems. It is worth noting, that Z3 enables finding some simple infinite models (such as identity) throughout model-based quantifier instantiation (mbqi) [5].

Conclusions and Future Work The proposed approach enables us to easily produce formulas that are difficult for state-of-the-art SMT solvers. We plan to extend this generation process by focusing also on different families of formulas and consider different parameterization of the process. An interesting question arises, how do we know that the formulas require infinite models? In the case of formulas that use integers/reals/string etc., this is immediate. In general, of course, this is again an undecidable problem. However, one more might try to attempt using heuristic techniques such as Infinox [2].

References

[1] Clark Barrett and Cesare Tinelli. Satisfiability modulo theories. In *Handbook of Model Checking*, pages 305–343. Springer, 2018.

[2] Koen Claessen and Ann Lillieström. Automated inference of finite unsatisfiability. *J. Autom. Reason.*, 47(2):111–132, 2011.

[3] Koen Claessen and Niklas Sörensson. New techniques that improve MACE-style finite model finding. In *Proceedings of the CADE-19 Workshop: Model Computation - Principles, Algorithms, Applications*, 2003.

[4] Leonardo Mendonça de Moura and Nikolaj Bjørner. Z3: an efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008.

[5] 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.

[6] William McCune. Mace4 reference manual and guide. *CoRR*, cs.SC/0310055, 2003.

[7] Julian Parsert, Chad E. Brown, Mikolas Janota, and Cezary Kaliszyk. Experiments on infinite model finding in SMT solving. In Ruzica Piskac and Andrei Voronkov, editors, *24th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR 2023*, volume 94 of *EPiC*, pages 317–328. EasyChair, 2023.