First-Order Instantiation using Discriminating Terms

Chad E. Brown[1], Mikoláš Janota[1]

[1]Czech Technical University in Prague, Czech Institute of Informatics, Robotics and Cybernetics, Jugoslávských partyzánů 1580/3, 160 00 Prague 6, Dejvice, Czech Republic

Abstract

This paper proposes a technique to limit the number of possible terms to be considered in quantifier instantiation. One of the major hurdles that SMT solvers face when dealing with quantifiers is that there are simply too many terms to instantiate with. So even if the right set of terms is available to the solver, meaning they appear in the formula, the solver might not have enough resources to come upon the right combination. This motivates the technique presented in this paper, which instantiates only by a certain type of terms, called discriminating terms. The paper introduces a class of formulas, where the proposed technique has a considerable impact.

Keywords

SMT, quantifiers, instantiation

1. Introduction

Quantifiers represent one of the major challenges for contemporary SMT solvers and since typically they lead to undecidability or extreme computational complexity, they are likely to remain a challenge for times to come.

Most commonly, the general techniques for dealing with quantifiers gradually instantiate the quantified part of the formula with ground terms until the resulting ground formula becomes unsatisfiable. The terms to be used in instantiations may be chosen either by syntactic properties (E-matching [1]) or semantic properties (e.g. model-based quantifier instantiation [2]). Interestingly, it has been shown that these techniques do not always pay off and simple enumeration of terms gives better results in some cases [3].

This is where this paper comes in. We propose a technique to limit the set of terms to enumerate. Roughly speaking, in the context of first order logic with equality, a term is labeled as *discriminating* if it participates in a disequality.

We have modified the enumeration instantiation algorithm in CVC4 [4] so that only discriminating terms are considered. We further construct a family of formulas where this approach demonstrably helps.

SMT 2021: 19th International Workshop on Satisfiability Modulo Theories, July 18-19, 2021, Los Angeles, CA

Mikolas.Janota@cvut.cz (M. Janota)

http://people.ciirc.cvut.cz/~janotmik/ (M. Janota)

0000-0003-3487-784X (M. Janota)

© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

CEUR-WS.org/Vol-2908/short6.pdf

2. A Class of Problems

Kaminski and Smolka [5] consider an identity that holds over the booleans: $f(f(f(c))) = f(c)$. Here c is a boolean and f is a unary function on booleans. It is easy to informally see why this identity holds by considering all four possible interpretations of f. Obviously the identity also holds if the domain of interest has only one element. Hence one can make an easy first-order problem by including an axiom

$$\forall xyz.x = y \vee x = z \vee y = z$$

stating there are at most two elements and making the conclusion $f(f(f(c))) = f(c)$. A formal proof would proceed by equational reasoning after instantiating the quantifiers using the subterms of the conjecture: c, $f(c)$, $f(f(c))$ and $f(f(f(c)))$. This problem can be made slightly more difficult by including n unary functions $g_0, \ldots, g_{n-1}$ and writing the conclusion as $f(f(f(g_0(\cdots g_{n-1}(c)\cdots)))) = f(g_0(\cdots g_{n-1}(c)\cdots))$. The modified problem has $n + 4$ subterms instead of only 4. The subterms of the form $g_i(\cdots g_{n-1}(c)\cdots)$ with $i > 0$ are red herrings. The four subterms $f^k(g_0(\cdots g_{n-1}(c)\cdots))$ with $k \in \{0, 1, 2, 3\}$ are sufficient to use as instantiations to complete the proof.

The problems can also be made more difficult in a different way. Following a proof from [6] we will argue that for each natural number $m > 0$ there are natural numbers $m_1 > 0$ and $m_2 \geq 0$ such that $f^{m_1+m_2}(c) = f^{m_2}(c)$ if the domain of interest has at most m elements. If $m = 2$, we can take $m_1 = 2$ and $m_2 = 1$ as above. More generally, we choose m_2 as $m - 1$ and m_1 is the least common multiple (lcm) of the sequence $2, \ldots, m$. The reason behind these choices are explored in the following subsection.

To construct the family of formulae in question, define atmost_m to be the first-order formula

$$\forall x_0 \cdots x_m. \bigvee_{0 \leq i < j \leq m} x_i = x_j$$

and let kam^n_m be the first-order formula

$$\mathsf{atmost}_m \wedge f^{\mathsf{lcm}(\{2,\ldots,m\})+m-1}(g_0(\cdots g_{n-1}(c)\cdots)) \neq f^{m-1}(g_0(\cdots g_{n-1}(c)\cdots)).$$

2.1. Unsatisfiability of kam^n_m

Following the proof (and terminology) of Theorem 7 in [6] we show that for an interpretation of size at most m the following identity holds:

$$f^{\mathsf{lcm}(\{2,\ldots,m\})+m-1}(c) = f^{m-1}(c)$$

The intuition for the equality is as follows. The sequence $c, f(c), \ldots, f^k(c), \ldots$ must eventually repeat. Both sides of the equation will be in the part that repeats and the length of the repeating part will be a number at most m. Since this length divides $\mathsf{lcm}(\{2, \ldots, m\})$ we will be able to conclude

$$f^{\mathsf{lcm}(\{2,\ldots,m\})}(f^{m-1}(c)) = f^{m-1}(c)$$

Let us now expand this idea more carefully.

Consider an arbitrary interpretation of the function f and the constant c assuming that the universe has at most m elements. For the purpose of this subsection, we abuse notation by writing $f(\cdots)$ for the value of f under such interpretation and write c for the value of c under the interpretation. By the pigeonhole principle there must exist q_1 and q_2 such that $0 \leq q_2 < q_1 \leq m$ such that $f^{q_1}(c) = f^{q_2}(c)$. Let q_1 and q_2 be the least numbers with this property. Following [6] we call q_1 the *size* and q_2 the *prefix*. We also call the positive number $q_1 - q_2$ the *lasso* and say $f^k(c)$ is *in the lasso* if $k \geq q_2$. The sequence can be written as follows:

$$\overbrace{f^0(c), \ldots, f^{q_2-1}(c)}^{q_2-1}, \underbrace{f^{q_2}(c), \ldots, f^{q_1-1}(c)}_{q_1-q_2}, \ldots, \underbrace{f^{q_2+j(q_1-q_2)}(c), \ldots, f^{q_1-1+j(q_1-q_2)}(c)}_{q_1-q_2}, \ldots \tag{1}$$

For every $f^k(c)$ in the lasso it is clear that $f^{j(q_1-q_2)}(f^k(c)) = f^k(c)$. Since $q_2 \leq m-1$ we know $f^{m-1}(c)$ is in the lasso. Since $q_1 \leq m$ we know the lasso is at most m and thus divides $\mathsf{lcm}(\{2, \ldots, m\})$. Hence we know

$$f^{\mathsf{lcm}(\{2,\ldots,m\})}(f^{m-1}(c)) = f^{m-1}(c)$$

as desired.

3. Quasidiscriminating Terms

The tableau calculus from [7] restricts first-order quantifier instantiation to so-called *discriminating terms*, i.e., terms that occur on one side of a disequation on the branch. A consequence of the completeness proof for the tableau calculus is a refinement of Herbrand's theorem indicating that (in the presence of the tableau calculus rules or analogous rules) instantiating with members of the universe of discriminating terms is sufficient to lead to an inconsistency, if the branch is inconsistent.

The change of setting from the tableau calculus of [7] to SMT means discriminating terms are not always sufficient. As a simple example we consider kam_2^0. Assume we have clause normalized so that there is one quantified formula

$$\forall xyz.x = y \vee x = z \vee y = z$$

and one disequation $f(f(f(c))) \neq f(c)$. There are only two discriminating terms: $f^3(c)$ and $f(c)$. Instantiating x, y and z with these two terms will always lead to at least two of x, y and z being the same term so that the resulting disjunction will always have a literal that is trivial by reflexivity. In the calculus of [7] there is a decomposition rule that would add $f(f(c)) \neq c$ to the branch since $f^3(c) \neq f(c)$ is on the branch. This new disequation means there are now four discriminating terms. As discussed above, these four terms are sufficient to use as instantiations to derive a contradiction.

One option would be to extend CVC4 to behave in ways that simulate the additional rules of [7]. In the example above, this would mean when the current propositional model sets the literal $f^3(c) = f(c)$ to false, CVC4 could mimic the decomposition rule by adding a propositional

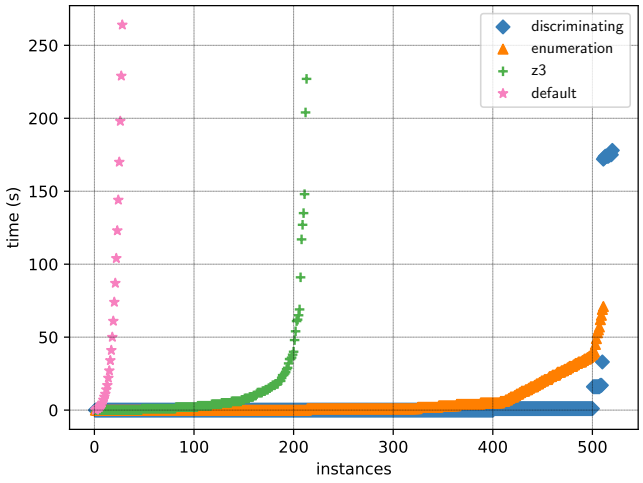

Figure 1: Cactus plot of the results.

clause corresponding to $f^3(c) = f(c) \vee f^2(c) \neq c$. Further tableau rules that would need to have a similar counterpart are the mating and confrontation rules.

We have chosen a simpler, heuristic approach without attempting to maintain completeness. However, a heuristic that restricts to discriminating terms without simulating the tableau rules would be far too restrictive. An intermediate heuristic is to restrict to terms that would be discriminating if the decomposition rule were included. We call these *quasidiscriminating terms*.

As a technical definition, we say a pair (s, t) is a *discriminating pair* if the literal $s = t$ is assigned false by the propositional model. We recursively define *quasidiscriminating pairs* (s, t) as follows: Every discriminating pair is a quasidiscriminating pair. If $(f(s_1, \ldots, s_n), f(t_1, \ldots, t_n))$ is a quasidiscriminating pair, then (s_i, t_i) is a quasidiscriminating pair for each $i \in \{1, \ldots, n\}$ where s_i and t_i are not the same term. A term s is *quasidiscriminating* if there is some t such that (s, t) or (t, s) is a quasidiscriminating pair.

We have modified the enumeration instantiation algorithm in CVC4 [4] so that only quasidiscriminating terms are considered.

4. Results

The problems used for evaluation are the formulas kam_m^n as defined above. The parameters were chosen as follows. The parameter m ranges between $4..10$ and the value of n ranges between $0..99$ for $m \in 4..8$ and it ranges between $0..9$ for $m \in 9..10$. Recall that the parameter m represents the domain size and n the number of the "dummy" terms g.

For the comparison we considered the default version of CVC4, CVC4 run only in the enumeration mode, and Z3. Figure 1 shows a cactus plot for the experiment results under 5-minute timeout (300s). Table 1 breaks down the number of solved instances by the domain size, i.e. by the parameter m.

domain	discriminating	enumeration	Z3	default
4	100	100	75	28
5	100	100	44	0
6	100	100	29	0
7	100	100	24	0
8	100	100	21	0
9	10	10	10	0
10	10	1	10	0
Total (520)	520	511	213	28

Table 1
Results

Our strategy using discriminating terms solved all the considered benchmarks very quickly except for the largest ones. CVC4's enumerative is also performing quite well but the time starts to increase much more quickly and eventually it times out on the largest problems. The default version of CVC4 performs rather poorly; our explanation for this is that the conflict-based instantiation [8] is taking up too much time because of the deep terms. The results for Z3 are surprising because it can successfully solve the largest instances but tends to fail on the smaller ones in a somewhat nonuniform fashion. We have also tried running Vampire [9] but that has timed out on all the considered problems.

5. Summary and Future Work

This paper proposes a way to restrict possible candidates term for quantifier instantiation by looking at syntactic properties of the given formula. In particular, we consider only terms that participate in a disequality. We construct a family of formulas where this technique has demonstrably the best results.

The presented techniques opens a number of avenues for future work. Since the presented technique disregards theories, a natural generalization would be to include theory-specific predicates, other than just disequality. For instance, in the context of arithmetic strict comparisons ($<$) imply disequality and therefore could be used in a similar fashion. While the technique is clearly performing well on the constructed family of formulas, as of now we don't have any dividends that is helpful on general formulas. We conjecture that this might happen in problems with many function nesting but also, careful integration with other techniques will be needed. A natural next step to take would be to run the modified CVC4 over the problem sets in the SMT-LIB to obtain data for how often the quasi-discriminating terms technique is helpful and how often it is not.

The family of formulas proposed here is interesting on its own, if only because they are easy to understand and yet present a challenge to existing SMT solvers and first-order automated theorem provers. In general, it is unclear whether it is better to instantiate with deeper terms or with more shallow terms. In the provided family, the outermost terms are actually the right ones and the inner ones are "red herrings." But one can envision scenarios where the opposite is true. So the question is, how to distinguish scenarios like these.

Acknowledgments

The results were supported by the Ministry of Education, Youth and Sports within the dedicated program ERC CZ under the project POSTMAN no. LL1902. This scientific article is part of the RICAIP project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857306.

References

[1] D. Detlefs, G. Nelson, J. B. Saxe, Simplify: a theorem prover for program checking, J. ACM 52 (2005) 365–473. doi:10.1145/1066100.1066102.

[2] Y. Ge, L. M. de Moura, Complete instantiation for quantified formulas in satisfiability modulo theories, in: Computer Aided Verification, 21st International Conference, CAV, 2009, pp. 306–320. doi:10.1007/978-3-642-02658-4_25.

[3] A. Reynolds, H. Barbosa, P. Fontaine, Revisiting enumerative instantiation, in: Tools and Algorithms for the Construction and Analysis of Systems, volume 10806, 2018, pp. 112–131. doi:10.1007/978-3-319-89963-3_7.

[4] C. W. Barrett, C. L. Conway, M. Deters, L. Hadarean, D. Jovanovic, T. King, A. Reynolds, C. Tinelli, CVC4, in: G. Gopalakrishnan, S. Qadeer (Eds.), Computer Aided Verification - 23rd International Conference, CAV, volume 6806, Springer, 2011, pp. 171–177. doi:10.1007/978-3-642-22110-1_14.

[5] M. Kaminski, G. Smolka, A finite axiomatization of propositional type theory in pure lambda calculus, in: Reasoning in Simple Type Theory: Festschrift in Honor of Peter B. Andrews on His 70th Birthday, College Publications, 2008, pp. 243–258.

[6] M. P. Bonacina, C. A. Lynch, L. de Moura, On deciding satisfiability by theorem proving with speculative inferences, Journal of Automated Reasoning 47 (2011) 161–189. doi:10.1007/s10817-010-9213-y.

[7] C. E. Brown, G. Smolka, Analytic tableaux for simple type theory and its first-order fragment, Logical Methods in Computer Science 6 (2010). doi:10.2168/LMCS-6(2:3)2010.

[8] A. Reynolds, C. Tinelli, L. M. de Moura, Finding conflicting instances of quantified formulas in SMT, in: Formal Methods in Computer-Aided Design, FMCAD 2014, Lausanne, Switzerland, October 21-24, 2014, IEEE, 2014, pp. 195–202. doi:10.1109/FMCAD.2014.6987613.

[9] L. Kovács, A. Voronkov, First-Order Theorem Proving and Vampire, in: International Conference on Computer Aided Verification, volume 8044, 2013, pp. 1–35. doi:10.1007/978-3-642-39799-8_1.