Mining counterexamples for wide-signature algebras with an Isabelle server *

Wesley Fussner[1] and Boris Shminke[1]

Laboratoire J.A. Dieudonné, CNRS, and Université Côte d'Azur, France
wfussner@unice.fr
Boris.SHMINKE@univ-cotedazur.fr

Abstract

We propose an approach for searching for counterexamples of statements about algebraic structures with a medium-sized signature using the Isabelle proof assistant in an efficient, parallel manner. We contribute a Python client Isabelle server and other scripts implementing our approach, and provide results of our computational experiments. In particular, our experiments yield counterexamples that resolve a previously open question regarding the interdependencies between distributive-like identities in residuated binars.

In partnership with automated theorem proving, finite model builders have been applied highly effectively in studies of algebraic structures (e.g., for quasigroups and loops [9]). However, the more fundamental operations there are appearing in an algebraic language, the more expensive computations become. Most successful applications of computational tools concern semigroups, quasigroups, and other algebraic structures with only a few fundamental operations, and algebras of even slightly wider signatures pose considerable challenges (but see, e.g., [11, 6]).

This contribution offers a case study in computer-assisted counterexample construction for algebras of wider signature. Our case study concerns *residuated binars* [4], each of which consists of an algebra $\mathbf{A} = (A, \wedge, \vee, \cdot, \backslash, /)$ such that $(A, \wedge, \vee)$ is lattice, $(A, \cdot)$ is a set with a binary operation, and for all $x, y, z \in A$,

$$x \cdot y \leq z \iff y \leq x\backslash z \iff x \leq z/y. \tag{1}$$

In this context, [4] studies implications among the distributivity identities

$$x \cdot (y \wedge z) = (x \cdot y) \wedge (x \cdot z) \tag{2}$$
$$(x \wedge y) \cdot z = (x \cdot z) \wedge (y \cdot z) \tag{3}$$
$$x\backslash(y \vee z) = (x\backslash y) \vee (x\backslash z) \tag{4}$$
$$(x \vee y)/z = (x/z) \vee (y/z) \tag{5}$$
$$(x \wedge y)\backslash z = (x\backslash z) \vee (y\backslash z) \tag{6}$$
$$x/(y \wedge z) = (x/y) \vee (x/z) \tag{7}$$

in the presence of *lattice distributivity*

$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z). \tag{8}$$

*This project received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 670624). This work has also been supported by the French government, through the 3IA Côte d'Azur Investments in the Future project managed by the National Research Agency (ANR) with the reference number ANR-19-P3IA-0002. The authors would like to thank Makarius Wenzel for commenting on an early version of the Python client code.

The identities (2)–(7) have proven important in obtaining normal forms for terms in residuated structures. This has found applications everywhere from establishing non-trivial categorical equivalences [5] to obtaining decidability results for models of program execution [13]. These identities are interdependent, and [4] establishes the following:

Theorem 1 (Theorem 2.3 of [4])**.** *Let* **A** *be a residuated binar satisfying* (8)*. Then:*

1. (5),(6) *implies* (4).
2. (4),(7) *implies* (5).
3. (2),(5) *implies* (7).
4. (3),(4) *implies* (6).
5. (6),(2) *implies* (3).
6. (7),(3) *implies* (2).

Furthermore, by exhibiting some small countermodels (of size 4 and 5), [4] shows that the implications announced in the previous theorem completely characterize all interdependencies among (2)–(7) in the presence of lattice distributivity. [4] mentions the case without lattice distributivity as an open question.

This contribution resolves the aforementioned question by the computer-assisted construction of finite countermodels witnessing the failure of the previous theorem in the general (i.e., non-lattice-distributive) case. We obtain:

Theorem 2. *In general, none of the distributivity laws* (2)–(7) *follows from any combination of the others.*

Thanks to its accessibility and amenability to proof simplification strategies (see [7]), working algebraists tend to favor McCune's Prover9/Mace4 [8] for automated work. However, Prover9/Mace4 is now regarded as somewhat out-of-date among researchers in automated deduction. Its model search capability has been gradually surpassed by a series of improvements in the field. The Paradox [3] system introduced so-called static symmetry reduction, a technique reducing the number of isomorphic models (see [1] for Mace4 and Paradox comparison). Later, Kodkod (see [12] for realization details and comparison with Paradox) brought sparse representation of binary relations and even more symmetry-breaking schemes to the process of translating a model-search task into a SAT problem.

We obtain Theorem 2 with the help of Nitpick, a highly efficient tool for the construction of finite counterexamples packaged with the Isabelle proof assistant [14]. Nitpick serves as a translator from Isabelle language to Kodkod, which currently relies on Jingeling ([2], the winner of SAT 2020 competition). Our work takes advantage of the fact that the Isabelle server implementation can run Nitpick tasks in parallel, yielding an environment for countermodel search with large computational advantages. Nevertheless, we observed that model search for residuated binars in cardinalities higher than 14 did not finish even after a week of running an Isabelle server. For Mace4 the boundary was lower, and even models of size larger than 5 became intractable.

To our best knowledge, there was no established way to communicate with the Isabelle server from Python. Thus we created a Python client to the Isabelle server [10] and a parser of Isabelle server logs (from TCP-communicated JSON parcels to LaTeX code for Cayley tables and Hasse diagrams). We conducted our computational experiments yielding Theorem 2 on three Linux machines, the largest having 180 CPU cores (Intel® Xeon® Gold 6254 3.10GHz) and 832 GB of RAM, totaling to about two weeks of wall-clock time. The conclusion of these experiments resulted in mined counterexamples (the largest having the underlying set of 10 elements) supporting the proof of Theorem 2. These counterexamples and the code for getting them are available as supplementary material for this paper.[1]

[1] https://github.com/inpefess/residuated-binars

References

[1] P. Baumgartner, A. Fuchs, H. de Nivelle, and C. Tinelli. Computing finite models by reduction to function-free clause logic. *J. Appl. Log.*, 7(1):58–74, 2009.

[2] A. Biere. CaDiCaL, Lingeling, Plingeling, Treengeling, YalSAT Entering the SAT Competition 2017. In T. Balyo, M. Heule, and M. Järvisalo, editors, *Proc. of SAT Competition 2017 – Solver and Benchmark Descriptions*, volume B-2017-1 of *Department of Computer Science Series of Publications B*, pages 14–15. University of Helsinki, 2017.

[3] K. Claessen and N. Sörensson. New techniques that improve MACE-style finite model finding. In *Proceedings of the CADE-19 Workshop: Model Computation-Principles, Algorithms, Applications*, pages 11–27. Citeseer, 2003.

[4] W. Fussner and P. Jipsen. Distributive laws in residuated binars. *Algebra Universalis*, 80.54, 2019.

[5] N. Galatos and J.G. Raftery. A category equivalence for odd Sugihara monoids and its applications. *J. Pure Appl. Algebra*, 216:2177–2192, 2012.

[6] P. Jipsen and M. Kinyon. Nonassociative right hoops. *Algebra Universalis*, 80.47, 2019.

[7] M. Kinyon. Proof simplification and automated theorem proving. *Philos. Trans. Roy. Soc. A*, 377.20180034, 2019.

[8] W. McCune. Prover9 and Mace4. `http://www.cs.unm.edu/~mccune/prover9/`, 2005–2010.

[9] J.D. Phillips and D. Stanovský. Automated theorem proving in quasigroup and loop theory. *AI Communications*, 23:267–283, 2010.

[10] B. Shminke. Python client for isabelle server. https://pypi.org/project/isabelle-client/.

[11] M. Spinks and R. Veroff. Constructive logic with strong negation is a substructural logic I. *Studia Logica*, 88:325–348, 2008.

[12] E. Torlak and D. Jackson. Kodkod: A relational model finder. In *In Tools and Algorithms for Construction and Analysis of Systems (TACAS)*, pages 632–647. Wiley, 2007.

[13] S. van Gool, A. Guatto, G. Metcalfe, and S. Santschi. Time warps, from algebra to algorithms. 2021. Preprint. Available at arXiv:2106.06205.

[14] M. Wenzel. *The Isabelle System Manual*. https://isabelle.in.tum.de/doc/system.pdf.