Discovering Universally Quantified Solutions for Constrained Horn Clauses

Arie Gurfinkel[1], Sharon Shoham[2], and Yakir Vizel[3]

[1] University of Waterloo
[2] Tel-Aviv University
[3] The Technion

1 Extended Abstract

The logic of Constrained Horn Clauses (CHC) provides an effective logical characterization of many problems in (software) verification. For example, CHC naturally capture inductive invariant discovery for sequential programs [4], compositional verification of concurrent and distributed systems [15, 19, 17], and verification of program equivalence [11]. CHC is used as an intermediate representation by several state-of-the-art program analysis tools, including SeaHorn [16] and JayHorn [20].

IC3 [6], initially introduced for model checking of finite state transition systems, has become the dominant model checking algorithm for hardware verification. Even more impressively, the IC3 framework (i.e., many algorithms built in the style of IC3) has become the dominant framework for exploring and building SAT/SMT-based verification algorithms. In particular, the framework has been extended to CHC modulo SMT theories in [7, 18, 8, 23, 3, 22, 21, 9]. The efficiency of IC3 for software verification is demonstrated by the effectiveness of such tools as SeaHorn. However, current extensions of IC3 are limited either in supported theories (e.g., no arithmetic), shape of the solution (i.e., quantifier free), or are not fully integrated within the IC3 framework.

In this work, we extend the IC3 framework to discovering universally quantified solutions to CHC. In the case that CHC are applied to software verification, these solutions correspond to universally quantified inductive invariants. This extends applicability of the framework, in particular, to reasoning about array manipulating programs and compositional verification of distributed protocols that require quantified invariants to reference arbitrary array locations and arbitrary processes.

Constrained Horn Clauses (CHC) is a fragment of First Order Logic (FOL) in which a formula is a conjunction of clauses, where each clause is a universally quantified formula of the form: $\forall \vec{x} \cdot p_1(\vec{x}) \wedge \cdots \wedge p_l(\vec{x}) \wedge \varphi \Rightarrow p_0(\vec{x})$, where each p_i is an uninterpreted predicate, and φ is a *constraint* over interpreted predicates and functions of some background theory $\mathcal{T}$. A set Φ of CHC is satisfiable modulo theory $\mathcal{T}$ if and only if there is a first order model that satisfies every clause of Φ and is consistent with the background theory $\mathcal{T}$. A *symbolic solution* Ψ to a set of CHC Φ is a map from each predicate p_i to a FOL formula $\psi(p_i)$ such that $\Phi[p_i \mapsto \psi(p_i)]$ is a valid sentence in $\mathcal{T}$. That is, $\psi(p_i)$ is a symbolic representation of a model for p_i.

We consider CHC where the constraints are in the combined theory of Linear Integer Arithmetic (LIA) and Arrays[1]. In many cases, solutions to such systems are definable by universally quantified formulas over the background theory. For example, defining that an array A is filled with 0 requires a quantified formula $\forall i \cdot A\,[i] = 0$. Quantifiers introduce two major challenges: (i) they tremendously increase the search space for a candidate solution, and (ii) they require deciding satisfiability of quantified formulas – itself an undecidable problem.

[1] However, our framework is more general and extends to arbitrary background SMT theory.

Existing techniques for inferring universally quantified solutions to CHC (and closely related techniques for inferring universally quantified invariants) work by either fixing the shape of quantified formulas and reducing to quantifier free inference (e.g., [5, 26, 17]), or by guessing quantified candidates by post-processing the solutions of bounded instances (e.g., [1]).

We exploit the IC3 framework to integrate the discovery of the necessary quantifiers into the search for the solution. To that end, we develop Quic3 – a generalization of the IC3 framework to discovering universally quantified solutions to CHC. Quic3 builds on Spacer [23] – an SMT-based extension of IC3 [6, 18]. Rather than fixing the quantifier structure apriori (e.g.[5, 26, 17]), or discovering quantifiers in a post-processing phase [1], Quic3 discovers the necessary quantifiers *on demand*. Moreover, to ensure convergence of each satisfiability check, it carefully manages the instantiations of each quantified lemma.

Discovery of quantifiers. The discovery of quantifiers is done by taking quantifiers into account during the blocking phase of IC3. The key ideas are to use existential quantifiers in proof obligations (or, counterexamples to induction) so that they are *blocked* by universally quantified lemmas, and to extend lemma generalization to add quantifiers.

Namely, in IC3, proof obligations are generated by taking a backward step from existing proof obligations (POBs). Such a backward step introduces existential quantifiers. Some can be eliminated, but the rest are typically projected by a model witness. Instead, we keep them in the POB, and ultimately, when the POB is blocked, get a universally quantified lemma. Nonetheless, this might still lead to proof obligations with concrete values, hindering convergence. To tackle this obstacle, we introduce an additional mechanism, that identifies concrete values in lemmas and generalizes them into universally quantified variables.

Handling Instantiations. Generating quantifiers on demand gives more control over the validity checks of a candidate solution (which corresponds to the pushing phase of IC3, or, inductiveness checks in software verification). This requires deciding satisfiability of *universally quantified* formulas over the combined theory of Arrays and LIA – an undecidable problem. Such checks are typically addressed in SMT solvers by *quantifier instantiation* where a universally quantified formula $\forall x \cdot \varphi(x)$ is approximated by a finite set of ground instances of φ. SMT solvers, such as Z3 [10], employ sophisticated heuristics (e.g., [13]) to find a sufficient set of instantiations. However, these heuristics are only complete in limited situations (recall, the problem is undecidable in general). It is typical for the solver to return *unknown*, or, even worse, diverge in an infinite set of instantiations.

Instead of using an SMT solver as a black-box, Quic3 generates and maintains a set of instantiations on demand. This ensures that Quic3 always makes progress and is never stuck in any single SMT call. The generation of instances is driven by the *blocking* phase of IC3 and is supplemented by traditional pattern-based triggers. Generating both universally quantified lemmas and their instantiations on demand, driven by the property, offers additional flexibility compared to the eager quantifier instantiation approach of [5, 26, 17].

Refutation Completeness. Combining the search for all of the ingredients (quantified and quantifier-free formulas, and instantiations) in a single procedure gives better control over the solving process. In particular, even though there is no guarantee of convergence (the problem is, after all, undecidable), we guarantee that Quic3 makes progress, exploring more of the problem, and discovering a refutation (even the shortest one) if the CHC system is unsatisfiable. In verification applications, refutations to CHC correspond to counterexamples, thus, Quic3 guarantees to find the shortest counterexample if it exists.

Implementation. We have implemented Quic3 in Z3 based on existing engines for Generalized

PDR [18, 23]. The input is a CHC instance in SMT-LIB format. The output is either a solution described in LIA and Arrays or a refutation derivation. To evaluate Quic3, we have used it to verify CHC instances from the domain of program verification. The instances are generated by SeaHorn from Array category of SV-COMP and other examples in the literature. We show that Quic3 extends applicability of SeaHorn to new problems, while maintaining competitive performance on SV-COMP benchmarks. Our implementation is competitive and can automatically discover non-trivial quantified invariants.

Related Work. Classical predicate abstraction [14, 2] has been adapted to quantified invariants by extending predicates with *skolem* (fresh) variables [12, 24]. This approach easily extends to finding quantified solutions to CHC. These techniques require a decision procedure for satisfiability of universally quantified formulas, and, significantly complicate predicate discovery (e.g., [25]). Quic3 extends this work to the IC3 framework in which the predicate discovery is automated and quantifier instantiation and instance discovery are carefully managed throughout the procedure.

Most CHC solvers do not support generating quantified solutions. Most common techniques for the ones that do support them is to use eager quantifier instantiations to approximate quantified solutions by quantifier free ones [5, 26, 17]. To our knowledge, UPDR [21] is the only current extension of IC3 to quantified solutions. While there are many differences in the supported input language, the key difference is that UPDR focuses on generating invariants in the Effectively PRopositional (EPR) fragment of first order logic for which quantified satisfiability is decidable. For that reason, UPDR does not need to deal with quantifier instantiation. Furthermore, UPDR does not use quantifier generalization and is limited to abstract counterexamples (i.e., counterexamples to existence of universal solutions, as opposed to counterexamples to satisfiability).

Contributions. In summary, we make the following contributions: (1) we extend the IC3 framework to deal with universally quantified lemmas in the combined theory of LIA and Arrays by discovering and maintaining quantified lemmas as well as their instantiations; (2) we develop new generalization techniques geared towards discovering universal quantifiers; (3) we implemented the algorithm in Z3; and (4) experimented with the approach in the context of software verification.

Acknowledgements. This publication is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No [759102-SVIS]). The research was partially supported by Len Blavatnik and the Blavatnik Family foundation, Tel Aviv University. We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), RGPAS-2017-507912.

References

[1] F. Alberti, R. Bruttomesso, S. Ghilardi, S. Ranise, and N. Sharygina. SAFARI: smt-based abstraction for arrays with interpolants. In *Computer Aided Verification - 24th International Conference, CAV 2012, Berkeley, CA, USA, July 7-13, 2012 Proceedings*, pages 679–685, 2012.

[2] T. Ball, A. Podelski, and S. K. Rajamani. Boolean and cartesian abstraction for model checking C programs. In *Tools and Algorithms for the Construction and Analysis of Systems, 7th International Conference, TACAS 2001 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2001 Genova, Italy, April 2-6, 2001, Proceedings*, pages 268–283, 2001.

[3] J. Birgmeier, A. R. Bradley, and G. Weissenbacher. Counterexample to induction-guided abstraction-refinement (CTIGAR). In A. Biere and R. Bloem, editors, *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings*, volume 8559 of *Lecture Notes in Computer Science*, pages 831–848. Springer, 2014.

[4] N. Bjørner, A. Gurfinkel, K. L. McMillan, and A. Rybalchenko. Horn clause solvers for program verification. In L. D. Beklemishev, A. Blass, N. Dershowitz, B. Finkbeiner, and W. Schulte, editors, *Fields of Logic and Computation II - Essays Dedicated to Yuri Gurevich on the Occasion of His 75th Birthday*, volume 9300 of *Lecture Notes in Computer Science*, pages 24–51. Springer, 2015.

[5] N. Bjørner, K. L. McMillan, and A. Rybalchenko. On solving universally quantified horn clauses. In *Static Analysis - 20th International Symposium, SAS 2013, Seattle, WA, USA, June 20-22, 2013. Proceedings*, pages 105–125, 2013.

[6] A. R. Bradley. SAT-Based Model Checking without Unrolling. In *Verification, Model Checking, and Abstract Interpretation - 12th International Conference, VMCAI 2011, Austin, TX, USA, January 23-25, 2011. Proceedings*, pages 70–87, 2011.

[7] A. Cimatti and A. Griggio. Software model checking via IC3. In P. Madhusudan and S. A. Seshia, editors, *Computer Aided Verification - 24th International Conference, CAV 2012, Berkeley, CA, USA, July 7-13, 2012 Proceedings*, volume 7358 of *Lecture Notes in Computer Science*, pages 277–293. Springer, 2012.

[8] A. Cimatti, A. Griggio, S. Mover, and S. Tonetta. IC3 modulo theories via implicit predicate abstraction. In E. Ábrahám and K. Havelund, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 20th International Conference, TACAS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014. Proceedings*, volume 8413 of *Lecture Notes in Computer Science*, pages 46–61. Springer, 2014.

[9] A. Cimatti, A. Griggio, S. Mover, and S. Tonetta. Infinite-state invariant checking with IC3 and predicate abstraction. *Formal Methods in System Design*, 49(3):190–218, 2016.

[10] L. M. de Moura and N. Bjørner. Z3: an efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, pages 337–340, 2008.

[11] D. Felsing, S. Grebing, V. Klebanov, P. Rümmer, and M. Ulbrich. Automating regression verification. In I. Crnkovic, M. Chechik, and P. Grünbacher, editors, *ACM/IEEE International Conference on Automated Software Engineering, ASE '14, Vasteras, Sweden - September 15 - 19, 2014*, pages 349–360. ACM, 2014.

[12] C. Flanagan and S. Qadeer. Predicate abstraction for software verification. In *Conference Record of POPL 2002: The 29th SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Portland, OR, USA, January 16-18, 2002*, pages 191–202, 2002.

[13] Y. Ge and L. M. de Moura. Complete instantiation for quantified formulas in satisfiabiliby modulo theories. In *Computer Aided Verification, 21st International Conference, CAV 2009, Grenoble, France, June 26 - July 2, 2009. Proceedings*, pages 306–320, 2009.

[14] S. Graf and H. Saïdi. Construction of abstract state graphs with PVS. In *Computer Aided Verification, 9th International Conference, CAV '97, Haifa, Israel, June 22-25, 1997, Proceedings*, pages 72–83, 1997.

[15] S. Grebenshchikov, N. P. Lopes, C. Popeea, and A. Rybalchenko. Synthesizing software verifiers from proof rules. In J. Vitek, H. Lin, and F. Tip, editors, *ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '12, Beijing, China - June 11 - 16, 2012*, pages 405–416. ACM, 2012.

[16] A. Gurfinkel, T. Kahsai, A. Komuravelli, and J. A. Navas. The seahorn verification framework. In *Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA,*

USA, July 18-24, 2015, Proceedings, Part I, pages 343–361, 2015.

[17] A. Gurfinkel, S. Shoham, and Y. Meshman. Smt-based verification of parameterized systems. In *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2016, Seattle, WA, USA, November 13-18, 2016*, pages 338–348, 2016.

[18] K. Hoder and N. Bjørner. Generalized property directed reachability. In *Theory and Applications of Satisfiability Testing - SAT 2012 - 15th International Conference, Trento, Italy, June 17-20, 2012. Proceedings*, pages 157–171, 2012.

[19] H. Hojjat, P. Rümmer, P. Subotic, and W. Yi. Horn clauses for communicating timed systems. In N. Bjørner, F. Fioravanti, A. Rybalchenko, and V. Senni, editors, *Proceedings First Workshop on Horn Clauses for Verification and Synthesis, HCVS 2014, Vienna, Austria, 17 July 2014.*, volume 169 of *EPTCS*, pages 39–52, 2014.

[20] T. Kahsai, P. Rümmer, H. Sanchez, and M. Schäf. Jayhorn: A framework for verifying java programs. In S. Chaudhuri and A. Farzan, editors, *Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part I*, volume 9779 of *Lecture Notes in Computer Science*, pages 352–358. Springer, 2016.

[21] A. Karbyshev, N. Bjørner, S. Itzhaky, N. Rinetzky, and S. Shoham. Property-directed inference of universal invariants or proving their absence. In *Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part I*, pages 583–602, 2015.

[22] A. Komuravelli, N. Bjørner, A. Gurfinkel, and K. L. McMillan. Compositional verification of procedural programs using horn clauses over integers and arrays. In *Formal Methods in Computer-Aided Design, FMCAD 2015, Austin, Texas, USA, September 27-30, 2015.*, pages 89–96, 2015.

[23] A. Komuravelli, A. Gurfinkel, and S. Chaki. SMT-Based Model Checking for Recursive Programs. In *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings*, pages 17–34, 2014.

[24] S. K. Lahiri and R. E. Bryant. Constructing quantified invariants via predicate abstraction. In *Verification, Model Checking, and Abstract Interpretation, 5th International Conference, VMCAI 2004, Venice, January 11-13, 2004, Proceedings*, pages 267–281, 2004.

[25] S. K. Lahiri and R. E. Bryant. Indexed predicate discovery for unbounded system verification. In *Computer Aided Verification, 16th International Conference, CAV 2004, Boston, MA, USA, July 13-17, 2004, Proceedings*, pages 135–147, 2004.

[26] D. Monniaux and L. Gonnord. Cell morphing: From array programs to array-free horn clauses. In *Static Analysis - 23rd International Symposium, SAS 2016, Edinburgh, UK, September 8-10, 2016, Proceedings*, pages 361–382, 2016.