From Bounded Checking to Verification of Equivalence via Symbolic up-to Techniques (Extended Abstract)

Vasileios Koutavas, Yu-Yang Lin and Nikos Tzevelekos

September 3, 2022

From Bounded Checking to Verification of Equivalence via Symbolic Up-to Techniques * (Extended Abstract).

Vasileios Koutavas[1], Yu-Yang Lin[1] (✉), and Nikos Tzevelekos[2]

[1] Trinity College Dublin, Dublin, Ireland {Vasileios.Koutavas,linhouy}@tcd.ie
[2] Queen Mary University of London, London, UK nikos.tzevelekos@qmul.ac.uk

Abstract. We present a bounded equivalence verification technique for higher-order programs with local state. This technique combines fully abstract *symbolic environmental bisimulations* similar to symbolic game semantics, novel *up-to techniques*, and lightweight *state invariant annotations*. This yields an equivalence verification technique with no false positives or negatives. The technique is bounded-complete, in that all inequivalences are automatically detected given large enough bounds. Moreover, several hard equivalences are proved automatically or after being annotated with state invariants. We realise the technique in a tool prototype called HOBBIT and benchmark it with an extensive set of new and existing examples. HOBBIT can prove many classical equivalences including all Meyer and Sieber examples.

Contextual equivalence is a relation over program expressions which guarantees that related expressions are interchangeable in any program context. It encompasses verification properties like safety and termination. It has attracted considerable attention from the semantics community (cf. the 2017 Alonzo Church Award), and has found its main applications in the verification of cryptographic protocols [4], compiler correctness [24] and regression verification [9,10,8,17].

In its full generality, contextual equivalence is hard as it requires reasoning about the behaviour of all program contexts, and becomes even more difficult in languages with higher-order features (e.g. callbacks) and local state. Advances in bisimulations [16,26,3], logical relations [1,12,14] and game semantics [18,23,7,19] have offered powerful theoretical techniques for hand-written proofs of contextual equivalence in higher-order languages with state. However, these advancements have yet to be fully integrated in verification tools for contextual equivalence in programming languages, especially in the case of bisimulation techniques. Existing tools [11,22,13] only tackle carefully delineated language fragments.

In this paper we aim to push the frontier further by proposing a bounded model checking technique for contextual equivalence for the entirety of a higher-order

* This work has emanated from research supported in part by a grant from Science Foundation Ireland under Grant number 13/RC/2094_2. It was presented in TACAS'22 [15]. We thank the PERR reviewers for their insightful comments.

language with local state ([15, Sec. 3]). This technique, realised in a tool called Hobbit,[3] automatically detects inequivalent program expressions given sufficient bounds, and proves hard equivalences automatically or semi-automatically.

Our technique uses a labelled transition system (LTS) for open expressions in order to express equivalence as a bisimulation. The LTS is symbolic both for higher-order arguments ([15, Sec. 4]), similarly to symbolic game models [7,19] and derived proof techniques [3,14], and first-order ones ([15, Sec. 6]), adopting established techniques (e.g. [5]) and tools such as Z3 [21]. This enables the definition of a fully abstract *symbolic environmental bisimulation*, the bounded exploration of which is the task of the Hobbit tool. Full abstraction guarantees that our tool finds all inequivalences given sufficient bounds, and only reports true inequivalences. As is corroborated by our experiments, this makes Hobbit a practical inequivalence detector, similar to traditional bounded model checking [2] which has been proved an effective bug detection technique in industrial-scale C code [5,6,27].

However, while proficient in bug finding, bounded model checking can rarely prove the absence of errors, and in our setting prove an equivalence: a bound is usually reached before all—potentially infinite—program runs are explored. Inspired by hand-written equivalence proofs, we address this challenge by proposing two key technologies: new *bisimulation up-to techniques*, and lightweight user guidance in the form of *state invariant annotations*. Hence we increase significantly the number of equivalences proven by Hobbit, including for example all classical equivalences due to Meyer and Sieber [20].

Up-to techniques [25] are specific to bisimulation and concern the reduction of the size of bisimulation relations, oftentimes turning infinite transition systems into finite ones by focusing on a core part of the relation. Although extensively studied in the theory of bisimulation, up-to techniques have not been used in practice in an equivalence checker. We specifically propose three novel up-to techniques: *up to separation* and *up to re-entry* ([15, Sec. 5]), dealing with infinity in the LTS due to the higher-order nature of the language, and *up to state invariants* ([15, Sec. 7]), dealing with infinity due to state updates. Up to separation allows us to reduce the knowledge of the context the examined program expressions are running in, similar to a frame rule in separation logic. Up to re-entry removes the need of exploring unbounded nestings of higher-order function calls under specific conditions. Up to state invariants allows us to abstract parts of the state and make finite the number of explored configurations by introducing state invariant predicates in configurations.

State invariants are common in equivalence proofs of stateful programs, both in handwritten (e.g. [16]) and tool-based proofs. In the latter they are expressed manually in annotations (e.g. [8]) or automatically inferred (e.g. [13]). In Hobbit we follow the manual approach, leaving heuristics for automatic invariant inference for future work. An important feature of our annotations is the ability to express relations between the states of the two compared terms, enabled by the up to

[3] Higher Order Bounded BIsimulation Tool (Hobbit), https://github.com/LaifsV1/Hobbit.

state invariants technique. This leads to finite bisimulation transition systems in examples where concrete value semantics are infinite state.

The above technologies, combined with standard up-to techniques, transform Hobbit from a bounded checker into an equivalence prover able to reason about infinite behaviour in a finite manner in a range of examples, including classical example equivalences (e.g. all in [20]) and some that previous work on up-to techniques cannot algorithmically decide [3] (cf. [15, Ex. 22]). We have benchmarked Hobbit on examples from the literature and newly designed ones ([15, Sec. 8]). Due to the undecidable nature of contextual equivalence, up-to techniques are not exhaustive: no set of up-to techniques is guaranteed to finitise all examples. Indeed there are a number of examples where the bisimulation transition system is still infinite and Hobbit reaches the exploration bound. For instance, Hobbit is not able to prove examples with inner recursion and well-bracketing properties, which we leave to future work. Nevertheless, our approach provides a contextual equivalence tool for a higher-order language with state that can prove many equivalences and inequivalences which previous work could not handle due to syntactic restrictions and other limitations ([15, Sec. 9]).

Related work

Our paper marries techniques from environmental bisimulations up-to [16,26,25,3] with the work on fully abstract game models for higher-order languages with state [18,7,19]. The closest to our technique is that of Biernacki et al. [3], which introduces up-to techniques for a similar symbolic LTS to ours, albeit with symbolic values restricted to higher-order types, resulting in infinite LTSs in examples such as [15, Ex. 21], and with inequivalence decided outside the bisimulation by (non-)termination, precluding the use up-to techniques in examples such as [15, Ex. 22]. Close in spirit is the line of research on logical relations [1,12,14] which provides a powerful tool for hand-written proofs of contextual equivalence. The relational verification engine behind [9,10] is also based on a bounded software model checker and can provide (unbounded) full proofs. Also related are the tools Hector [11] and Coneqct [22], and SyTeCi [13], based on game semantics and step-indexed logical relations respectively (cf. [15, Sec. 9]).

References

1. Ahmed, A., Dreyer, D., Rossberg, A.: State-dependent representation independence. In: POPL. Association for Computing Machinery (2009)
2. Biere, A., Cimatti, A., Clarke, E., Zhu, Y.: Symbolic model checking without BDDs. In: TACAS. Springer Berlin Heidelberg (1999)
3. Biernacki, D., Lenglet, S., Polesiuk, P.: A complete normal-form bisimilarity for state. In: FOSSACS 2019, ETAPS 2019, Prague, Czech Republic. Springer (2019)
4. Blanchet, B.: A computationally sound mechanized prover for security protocols. In: IEEE Symposium on Security and Privacy (2006)
5. Clarke, E., Kroening, D., Lerda, F.: A tool for checking ANSI-C programs. In: TACAS. Springer Berlin Heidelberg (2004)
6. Cordeiro, L., Kroening, D., Schrammel, P.: JBMC: Bounded model checking for Java Bytecode. In: TACAS. Springer (2019)
7. Dimovski, A.: Program verification using symbolic game semantics. TCS **560** (2014)

8. Felsing, D., Grebing, S., Klebanov, V., Rümmer, P., Ulbrich, M.: Automating regression verification. In: ACM/IEEE ASE '14. ACM (2014)
9. Godlin, B., Strichman, O.: Inference rules for proving the equivalence of recursive procedures. Acta Informatica **45**(6) (2008)
10. Godlin, B., Strichman, O.: Regression verification. In: DAC. ACM (2009)
11. Hopkins, D., Murawski, A.S., Ong, C.L.: Hector: An equivalence checker for a higher-order fragment of ML. In: CAV. LNCS, Springer (2012)
12. Hur, C.K., Dreyer, D., Neis, G., Vafeiadis, V.: The marriage of bisimulations and Kripke logical relations. SIGPLAN Not. (2012)
13. Jaber, G.: SyTeCi: Automating contextual equivalence for higher-order programs with references. Proc. ACM Program. Lang. **4**(POPL) (2020)
14. Jaber, G., Tabareau, N.: Kripke open bisimulation - A marriage of game semantics and operational techniques. In: APLAS. Springer (2015)
15. Koutavas, V., Lin, Y., Tzevelekos, N.: From bounded checking to verification of equivalence via symbolic up-to techniques. In: TACAS. Lecture Notes in Computer Science, vol. 13244, pp. 178–195. Springer (2022)
16. Koutavas, V., Wand, M.: Small bisimulations for reasoning about higher-order imperative programs. In: POPL. ACM (2006)
17. Lahiri, S.K., Hawblitzel, C., Kawaguchi, M., Rebêlo, H.: SYMDIFF: A language-agnostic semantic diff tool for imperative programs. In: CAV. Springer (2012)
18. Laird, J.: A fully abstract trace semantics for general references. In: ICALP, Wroclaw, Poland. LNCS, Springer (2007)
19. Lin, Y., Tzevelekos, N.: Symbolic execution game semantics. In: FSCD. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020)
20. Meyer, A.R., Sieber, K.: Towards fully abstract semantics for local variables. In: POPL. Association for Computing Machinery (1988)
21. de Moura, L., Bjørner, N.: Z3: An efficient smt solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 337–340. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
22. Murawski, A.S., Ramsay, S.J., Tzevelekos, N.: A contextual equivalence checker for IMJ*. In: ATVA. Springer (2015)
23. Murawski, A.S., Tzevelekos, N.: Nominal game semantics. FTPL **2**(4) (2016)
24. Patterson, D., Ahmed, A.: The next 700 compiler correctness theorems (functional pearl). Proc. ACM Program. Lang. **3**(ICFP) (2019)
25. Pous, D., Sangiorgi, D.: Enhancements of the bisimulation proof method. In: Advanced Topics in Bisimulation and Coinduction. CUP (2012)
26. Sangiorgi, D., Kobayashi, N., Sumii, E.: Environmental bisimulations for higher-order languages. In: LICS. IEEE Computer Society (2007)
27. Schrammel, P., Kroening, D., Brain, M., Martins, R., Teige, T., Bienmüller, T.: Successful use of incremental BMC in the automotive industry. In: FMICS (2015)