Symbolic Bounded Model Checking of $\forall^+\exists^+$-Liveness Hyperproperties

Alcino Cunha[1 [0000−0002−2714−8027]], Hugo Pacheco[1 [0000−0003−0720−7744]], and Nuno Macedo[1 [0000−0002−4817−948X]]

Universidade do Minho & INESC TEC, Portugal
{alcino,hpacheco,nmacedo}@di.uminho.pt

1 Overview

Among the existing model checkers for HyperLTL, the explicit-state AutoHyper [2] stands alone in supporting arbitrary properties over arbitrary reactive systems, at the cost of severe state-explosion. Symbolic bounded model checking (BMC) tools, notably HyperQB [10,9], scale better but are restricted to either (i) hyperproperties whose body is a safety LTL formula, or (ii) arbitrary bodies over *terminating* systems—and models with stuttering easily break the terminating assumption. The fragment of $\forall^+\exists^+$ hyperproperties whose inner LTL body includes liveness constraints, over arbitrary reactive systems—which captures, e.g., generalized non-interference, self-stabilization, robustness, and several synthesis problems—has so far had no symbolic BMC procedure.[1]

The reason is technical: refuting $\forall^+\exists^+\psi$ with a body ψ that is not a pure safety constraint amounts to finding a witness for $\exists^+\forall^+\neg\psi$, which requires reasoning over infinite (lasso-shaped) paths rather than finite prefixes. A direct BMC encoding loses soundness, because a k-loop instantiation of the existentials may be *spurious* — the inner universal counter-part might be a k'-loop with $k' > k$ that the bounded procedure never explored. Vanilla BMC for HyperLTL is therefore unusable for most non-safety bodies over most systems, since we cannot conclude whether the reported counter-example is valid.

We present HyperLasso, the first symbolic BMC tool for this fragment, and outline the underlying technique and evaluation. We achieve this by pairing a BMC procedure, that will be used as a candidate *synthesizer*, with a *checker* that verifies if candidates are real bugs of the full system.

2 Lasso-shaped paths for HyperLTL

Our work builds on the fundamental result of [11] that lasso-shaped traces capture all relevant behaviours for HyperLTL: any falsifiable formula admits a counter-example whose constituent traces are lassos. This is what makes a bounded encoding meaningful in the presence of liveness constraints. A second

[1] The full paper underlying this talk has been accepted at CAV 2026. You may check the associated artifact at https://doi.org/10.5281/zenodo.19850821.

observation from [11], used in their BMC procedure for $\forall\exists\Box\psi$ and $\exists\forall\Box\psi$ (implemented in [9]), is that lassos arising from different traces may have different lengths, so any sound procedure must align them across the least common multiple (LCM) of their periods.

The primary technical insight underpinning our synthesizer is that, for a hyperproperty over two models analysed up to bound k, the symbolic encoding need not unroll each model to depth $k \cdot k$—the worst case suggested by the LCM alignment. We unroll each model to depth k only, and the hyperformula, which may logically require analysis up to depth $k \cdot k$, refers back to the variables of the depth-k unrolling. By contrast, the procedure of [11] unrolls one model to bound k but the other to its full state-space cardinality $|S|$, with $|S| \cdot k$ variables for the simulation relation; our encoding keeps the variable space constant in the unrolling depth, independently of the explicit state count.

3 The synthesizer/checker loop

HyperLasso pairs a bounded *synthesizer* with a complete *checker* in a CEGAR-style loop. Given $\models_k \forall^+\exists^+\psi$, we negate to $\models_k \exists^+\forall^+\neg\psi$ and iterate:

1. The **synthesizer** extends the QBF encoding of HyperQB with lasso-shaped paths, producing a candidate instantiation of the outer existentials as a k-loop. As discussed above, the encoding aligns lassos of different periods without unrolling each model beyond depth k, and supports an independent bound k_i per trace variable (used by step 3 below).
2. The **checker** validates the candidate against the full system. With a single quantifier alternation, once the existentials are fixed, the residual problem $\forall^+\psi$ reduces by self-composition [1] to a plain LTL model checking problem, dispatched to an off-the-shelf complete LTL checker (e.g., nuXmv [3]).
3. If the checker finds the candidate spurious, it returns a k'-loop counter-part with $k' > k$; we get back to the synthesizer and lift the inner bounds k_i for the universally quantified traces to k'—which prunes not only the spurious candidate but every candidate refuted by counter-parts of length $\leq k'$—and re-invoke the BMC procedure. This iteratively approximates a completeness threshold for the residual LTL problem.

The procedure enjoys two guarantees. *Infinite inference* (soundness): every k-loop returned by HyperLasso is a witness of $\models \exists^+\forall^+\psi$ under the unbounded semantics—looping paths unroll to any $k' > k$, and longer universal counter-parts are already excluded by the complete checker. *Bounded inference* (partial completeness): if HyperLasso terminates without a witness for bound k, then $\not\models_k \exists^+\forall^+\psi$, i.e. no counter-example to the original $\forall^+\exists^+\psi$ exists with constituent traces of lasso length $\leq k$; witnesses with lasso length $k' > k$ may still exist. We do not compute or stop at a global completeness threshold – doing so remains an orthogonal challenge, addressed only for restricted fragments [11] and arguably infeasible at realistic scales – but bounded inference gives the user an incremental, monotone notion of confidence as k grows.

4 Implementation and evaluation

HyperLasso consumes SMV models and HyperLTL specifications in standard formats, and reduces synthesis to QBF (in practice, we use Z3 [6] with only quantifier and Boolean/bit theories as a QBF solver) and checking to LTL model checking via nuXmv. To evaluate it we built a new benchmark suite of hyperproperties with liveness constraints over parameterised reactive systems, since prior benchmarks are dominated by purely safety properties or carefully crafted terminating systems. It comprises five verification problems—generalized non-interference in a conference management system [13,15], equivalence of database isolation levels (Read Committed vs. Serializability) [5], concurrent non-interference for asynchronous multi-threaded programs [16], Herman's self-stabilization on a process ring [8,14], and robustness of Lamport's bakery protocol [4,12]—and two synthesis problems[2]—controller synthesis for traffic semaphores cast as an LTL synthesis problem [7], and robot path planning [10]—with both valid and invalid instances.

AutoHyper, the only competing tool that supports the fragment supported by HyperLasso[3], was used as baseline. Across the benchmark, AutoHyper solves only the smallest scopes (often below the parameter ranges where the property's nominal outcome holds, leading to vacuous answers) and times out at 300 s on virtually every non-trivial instance. HyperLasso consistently scales further: it produces witnesses for invalid/satisfiable problems and increasing bounds of confidence for valid/unsatisfiable ones, with the synthesiser dominating runtime. Spurious candidates do arise—in self-stabilization, bakery robustness, and the synthesis problems—confirming that the checker is not optional for soundness, but their number stays small (at most two per run in our experiments) and the checker's cost is negligible compared with the synthesizer's. In several cases a single refinement step lifts k' enough to close the loop in the next iteration.

5 Outlook

We are exploring how to extend the technique to multiple quantifier alternations, where the residual problem after fixing the outer existentials is itself a hyperproperty rather than plain LTL, and exploring dedicated QBF backends and tighter refinement strategies for the inner bounds.

Acknowledgments. The work of Alcino Cunha is financed by National Funds through the FCT - Fundação para a Ciência e a Tecnologia, I.P. (Portuguese Foundation for Science and Technology) within the CMU Portugal Visiting Faculty and Researchers Program. The work of Hugo Pacheco is funded by national funds through FCT – Fundação para a Ciência e a Tecnologia, I.P., under the support UID/50014/2025

[2] We consider $\exists^+\forall^+\psi$ properties with safety constraints, which are the dual of $\forall^+\exists^+\psi$ with liveness constraints and can be solved with the same procedure.

[3] LTL synthesis tools [7] could in fact be used for the controller synthesis benchmark almost off-the-shelf, but with slightly different formats and encoding.

(https://doi.org/10.54499/UID/50014/2025). The work of Nuno Macedo is financed by National Funds through the FCT - Fundação para a Ciência e a Tecnologia, I.P. (Portuguese Foundation for Science and Technology) within the project POISE, with reference 2024.15783.PEX (https://doi.org/10.54499/2024.15783.PEX).

References

1. Barthe, G., D'argenio, P.R., Rezk, T.: Secure information flow by self-composition. Mathematical Structures in Computer Science **21**(6), 1207–1252 (2011)
2. Beutner, R., Finkbeiner, B.: AutoHyper: Explicit-state model checking for HyperLTL. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 145–163. Springer (2023)
3. Cavada, R., Cimatti, A., Dorigatti, M., Griggio, A., Mariotti, A., Micheli, A., Mover, S., Roveri, M., Tonetta, S.: The nuXmv symbolic model checker. In: International Conference on Computer Aided Verification. pp. 334–342. Springer (2014)
4. Coenen, N., Finkbeiner, B., Sánchez, C., Tentrup, L.: Verifying hyperliveness. In: International Conference on Computer Aided Verification. pp. 121–139. Springer (2019)
5. Crooks, N., Pu, Y., Alvisi, L., Clement, A.: Seeing is believing: A client-centric specification of database isolation. In: Proceedings of the ACM Symposium on Principles of Distributed Computing. pp. 73–82 (2017)
6. De Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: International conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 337–340. Springer (2008)
7. Faymonville, P., Finkbeiner, B., Tentrup, L.: Bosy: An experimentation framework for bounded synthesis. In: Majumdar, R., Kunčak, V. (eds.) Computer Aided Verification. pp. 325–332. Springer International Publishing, Cham (2017)
8. Herman, T.: Probabilistic self-stabilization. Inf. Process. Lett. **35**(2), 63–67 (1990)
9. Hsu, T.H., Rabizadeh, M., Rogale, K., Filippov, F., de Oliveira Batista, M.A., Sánchez, C., Bonakdarpour, B.: Hyperqb 2.0: A bounded model checker for hyperproperties (2025), https://arxiv.org/abs/2109.12989
10. Hsu, T.H., Sánchez, C., Bonakdarpour, B.: Bounded model checking for hyperproperties. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 94–112. Springer (2021)
11. Hsu, T.H., Sánchez, C., Sheinvald, S., Bonakdarpour, B.: Efficient loop conditions for bounded model checking hyperproperties. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 66–84. Springer (2023)
12. Lamport, L.: A new solution of Dijkstra's concurrent programming problem. In: Concurrency: the works of Leslie Lamport, pp. 171–178 (2019)
13. Lamport, L., Schneider, F.B.: Verifying hyperproperties with TLA. In: 2021 IEEE 34th Computer Security Foundations Symposium (CSF). pp. 1–16. IEEE (2021)
14. Lin, A.W., Rümmer, P.: Liveness of randomised parameterised systems under arbitrary schedulers. In: CAV (2). LNCS, vol. 9780, pp. 112–133. Springer (2016)
15. Macedo, N., Pacheco, H.: Hyper model checking for high-level relational models (2025), https://arxiv.org/abs/2512.12024
16. Smith, G., Volpano, D.M.: Secure information flow in a multi-threaded imperative language. In: POPL. pp. 355–364. ACM (1998)