HyperLTL Model Checking

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Author: Ondrej Alexaj
Supervisor: Ing. Ondřej Lengál, Ph.D

Excel@FIT 2024

HyperLTL

The HyperLTL formulas extend LTL formulas with quantifiers over traces and are generated by the following grammar [3]:

$$\phi \coloneqq \exists \pi.\phi \mid \forall \pi.\phi \mid \psi$$

$$\psi \coloneqq a_\pi \mid \neg\psi \mid \psi \wedge \psi \mid \mathrm{X}\psi \mid \psi \mathrm{U} \psi.$$

As an illustration, consider the following HyperLTL formula, which specifies *general noninterference* (GNI) property for the Boolean program:

$$\forall \pi_1 \cdot \forall \pi_2 \cdot \exists \pi_3 \cdot G(h[0]_{\pi_1} \leftrightarrow h[0]_{\pi_3}) \wedge G(l[0]_{\pi_2} \leftrightarrow l[0]_{\pi_3}) \wedge G(o[0]_{\pi_2} \leftrightarrow o[0]_{\pi_3}).$$

Kripke structure $\mathcal{K} = (S, s_0, \delta, AP, L)$ provides formalism to represent system behavior. HyperLTL model checking then decides whether $\mathcal{K} \models \varphi$.

HyperLTL Model Checking scheme

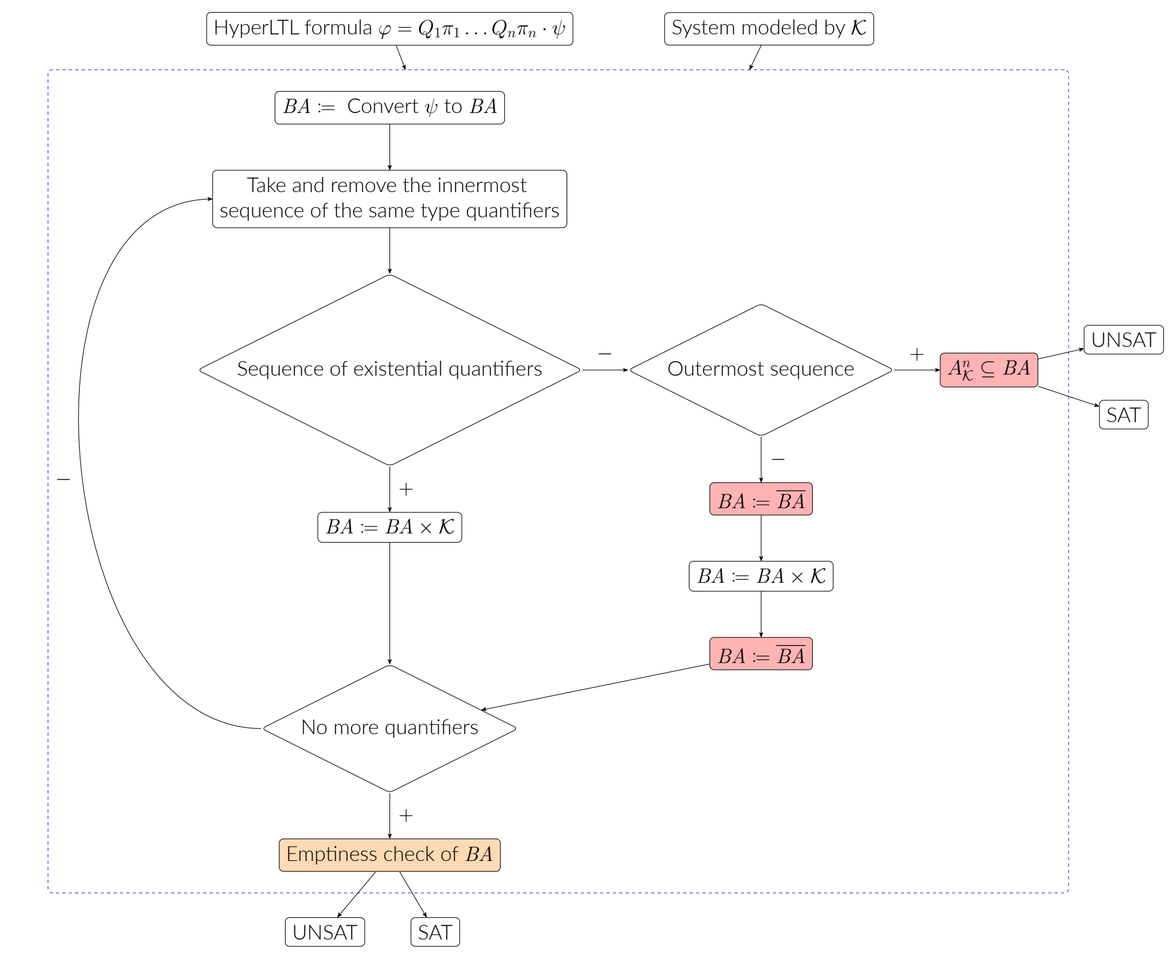
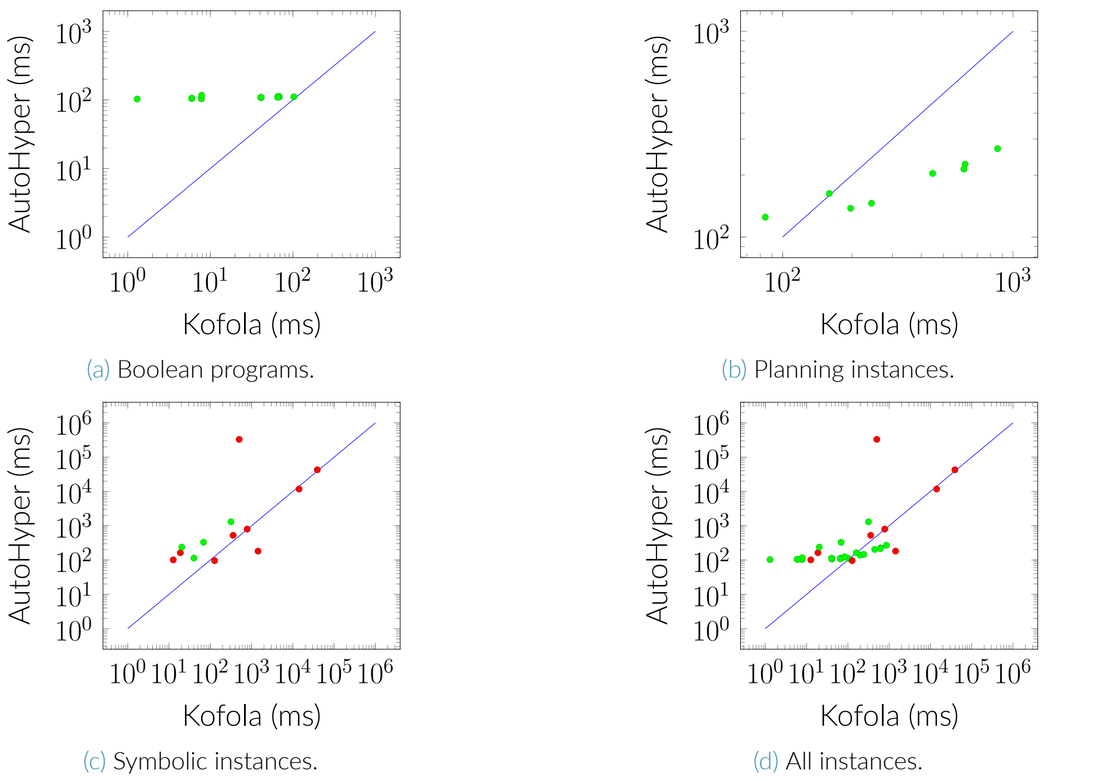


Figure 1. The scheme summarizes the steps of the Automata-based Verification [4] and automata operations it employs. Colors indicate the focus on optimizing specific operations, with red frames indicating a greater point of interest and orange frames indicating a lower focus.

Kofola vs AutoHyper



(a) Boolean programs.

(b) Planning instances.

(c) Symbolic instances.

(d) All instances.

Figure 2. The scatter plots compare the execution time of HyperLTL model checking procedure in Kofola [5] vs AutoHyper [1], tested on 36 instances. Green circles indicate instances where the system satisfies the formula, while red circles indicate non-satisfaction.

As an illustration, consider the following Boolean program, which has been verified against the GNI property:

Algorithm 1 Simple boolean program.

```
h: 1; // bitwidth 1
l: 1;
o: 1;

o ← 1 * true; // set bit to true
while true do
    o ← !o; // update bit value by negating
end while
```

Language inclusion

$$L(A) \overset{?}{\subseteq} L(B) \iff L(A) \cap \overline{L(B)} \overset{?}{=} \emptyset.$$

The following implications illustrate the manner in which subsumptions can be employed to decide inclusion and avoid constructing the entire counterexample.

$$(p \overset{u}{\underset{\bullet}{\rightsquigarrow}}_{\bullet} q \wedge p \preceq_{e^{+1}} q) \implies L(p) \neq \emptyset, \tag{1}$$

$$(p \overset{u}{\underset{\bullet}{\rightsquigarrow}} q \wedge p \preceq_{e} q) \implies L(p) \neq \emptyset. \tag{2}$$

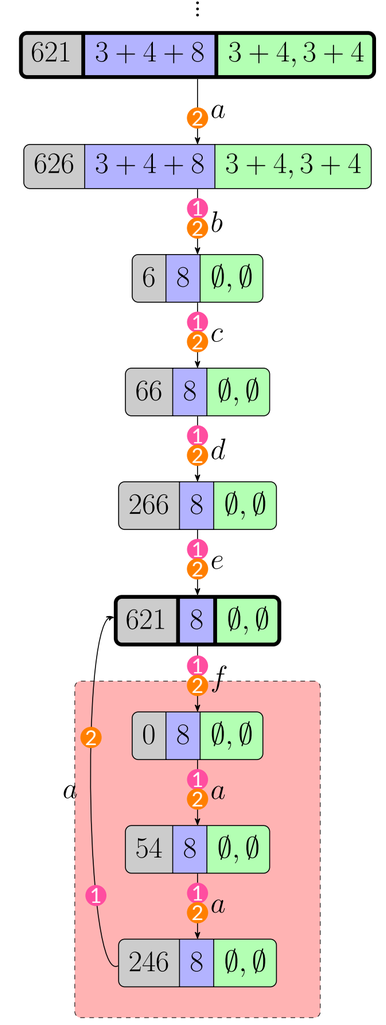


Figure 3. A fragment of the product automaton $A \cap B_{compl}$ with $Acc = Inf(1) \wedge Inf(2)$ illustrates the effect of the implication (2), where $p = (621 \mid 3 + 4 + 8 \mid 3 + 4, 3 + 4)$ and $q = (621 \mid 8 \mid \emptyset, \emptyset)$.

Inclusion checker Kofola vs Spot

Table 1. The table presents a comparison of state spaces generated by solving the inclusion. Specifically, it compares the best Kofola's [5] inclusion procedure (hereafter referred to as Kofola - max) with Spot [2] and non-optimized Kofola's inclusion procedure. The column **"wins"/"losses"** contains the number of cases where the Kofola - max produced strictly less/more states. The (number) in the **"wins"/"losses"** column means how many times it was due to the other procedure's timeout.

tool	solved	mean	median	wins	losses
Kofola - max	33	1828.70	75	-	-
Kofola - basic	35	2833.74	185	3 (3)	16 (1)
Spot	35	5264.21	513	11 (2)	20 (1)

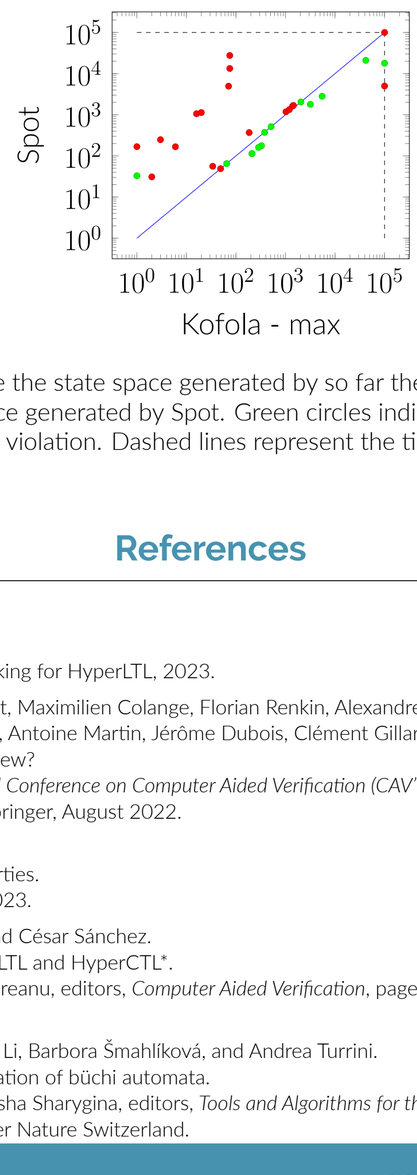


Figure 4. The scatter plots compare the state space generated by so far the most optimized algorithm used within Kofola against the state space generated by Spot. Green circles indicate instances where the inclusion holds, while red circles indicate the violation. Dashed lines represent the timeout (set to 10 minutes).

References

[1] Raven Beutner and Bernd Finkbeiner.
Autohyper: Explicit-state model checking for HyperLTL, 2023.

[2] Alexandre Duret-Lutz, Etienne Renault, Maximilien Colange, Florian Renkin, Alexandre Gbaguidi Aisse, Philipp Schlehuber-Caissier, Thomas Medioni, Antoine Martin, Jérôme Dubois, Clément Gillard, and Henrich Lauko.
From Spot 2.0 to Spot 2.10: What's new?
In *Proceedings of the 34th International Conference on Computer Aided Verification (CAV'22)*, volume 13372 of *Lecture Notes in Computer Science*, pages 174–187. Springer, August 2022.

[3] Bernd Finkbeiner.
Logics and algorithms for hyperproperties.
ACM SIGLOG News, 10(2):4–23, jul 2023.

[4] Bernd Finkbeiner, Markus N. Rabe, and César Sánchez.
Algorithms for model checking HyperLTL and HyperCTL*.
In Daniel Kroening and Corina S. Păsăreanu, editors, *Computer Aided Verification*, pages 30–48, Cham, 2015. Springer International Publishing.

[5] Vojtěch Havlena, Ondřej Lengál, Yong Li, Barbora Šmahlíková, and Andrea Turrini.
Modular mix-and-match complementation of büchi automata.
In Sriram Sankaranarayanan and Natasha Sharygina, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 249–270, Cham, 2023. Springer Nature Switzerland.

xalexa09@stud.fit.vutbr.cz