Volume 12, Number 3
July 2025

Published by the Association for Computing Machinery
Special Interest Group on Logic and Computation

SIGLOG news

TABLE OF CONTENTS

General Information

1 From the Editor-in-Chief *Daniele Gorla*

2 From the SIGLOG Chair *Catuscia Palamidessi*

Technical Columns

3 Security and Privacy *Steve Kremer*

Regular Features

23 Conference Report *Joost-Pieter Katoen*

27 SIGLOG Monthly 260 *Elli Anastasiadi*

Association for Computing Machinery

Advancing Computing as a Science & Profession

SIGLOG NEWS

Published by the ACM Special Interest Group on Logic and Computation

SIGLOG Executive Committee

Chair	Catuscia Palamidessi	Inria Saclay
Vice-Chair	Andrzej Murawski	University of Oxford
Secretary	Sandra Alves	Universidad do Porto
Treasurer	Elaine Pimentel	University College London
	Franck van Breugel	York University
	Liron Cohen	Ben-Gurion University
	Carsten Fuhs	Birkbeck, University of London
	Prakash Panangaden	McGill University
	Simona Ronchi Della Rocca	Univ. of Torino
CADE President	Christoph Weidenbach	Max Planck Inst. Saarbrücken
EATCS President	Artur Czumaj	University of Warwick
EACSL President	Maribel Fernandez	King's College London
ACM ToCL E-in-C	Anuj Dawar	University of Cambridge
ASL Rep	Phokion Kolaitis	UC Santa Cruz

EDITOR IN CHIEF

Daniele Gorla Sapienza University of Rome

TECHNICAL COLUMN EDITORS

Automata	Mahesh Viswanathan	U. of Illinois Urbana-Champaign
Complexity	Andrei Bulatov	Simon Fraser University
Security and Privacy	Steve Kremer	LORIA, Inria Nancy, U. Lorraine
Semantics	Prakash Panangaden	McGill University
Verification	Christel Baier	Technische Universität Dresden

REGULAR FEATURES

Conference Reports	Joost-Pieter Katoen	RWTH Aachen Univ.
SIGLOG Monthly	Elli Anastasiadi	University of Uppsala

Notice to Contributing Authors to SIG Newsletters

As a contributing author, you retain copyright to your article. ACM will refer all requests for republication directly to you. By submitting your article for distribution in any newsletter of the ACM Special Interest Groups, you hereby grant to ACM the following non-exclusive, perpetual, worldwide rights:

— to publish your work online or in print on condition of acceptance by the editor
— to include the article in the ACM Digital Library and in any Digital Library-related services
— to allow users to make a personal copy of the article for noncommercial, educational, or research purposes
— to upload your video and other supplemental material to the ACM Digital Library, the ACM YouTube channel, and the SIG newsletter site

Furthermore, you affirm that:

— if third-party materials were used in your published work, supplemental material, or video, that you have the necessary permissions to use those third-party materials in your work

---

SIGLOG News (ISSN 2372-3491) is an electronic quarterly publication by the Association for Computing Machinery.

From the Editor-in-Chief



This third issue of 2025 starts with an important letter of the SIGLOG Chair Catuscia Palamidessi describing the situation of SIGLOG and its future. Then, we include a security and privacy column, collected by Steve Kremer and written by Raven Beutner, Arthur Correnson, and Bernd Finkbeiner on the verification of *temporal* $\forall\exists$*-hyperliveness properties*, that include noninference and generalized noninterference; these express requirements that relate multiple system executions through combinations of universal and existential quantification and pose fundamental challenges that go beyond those encountered in standard trace or k-hypersafety properties. Then, the conference coordinator of SIGLOG News, Joost-Pieter Katoen, introduces a report on the flagship conference of SIGLOG, namely LICS 2025 (the 40th edition of the ACM/IEEE Symposium on Logic in Computer Science), written by the PC Chairs Barbara König and Lars Birkedal. Finally, we wrap up this number with the latest issue of SIGLOG Monthly, prepared by Elli Anastasiadi.

Enjoy reading!

Daniele Gorla
Sapienza University of Rome
SIGLOG News Editor-in-Chief
editor@siglog.org

Chair's Letter



Dear colleagues,

There is some exciting news about SIGLOG. As you may remember from my previous letters, about two years ago ACM decided to increase the overhead charges—i.e., the SIG's annual management fee—and for SIGLOG, the overhead would have risen to 25,000 USD (from the 10,000 USD it had been previously). We could not come up with a viable financial plan to cover this fee, and consequently, we explored the possibility of merging SIGLOG with SIGPLAN. This option, however, was not very popular among our community, as shown by the results of the recent consultation about LICS's institutional future, promoted by the LICS steering committee. In the same consultation, LICS explored the possibility of becoming an independent conference, which appeared to be a far more popular option.

This July, the other members of the SIGLOG Executive Committee and I, along with Igor Walukiewicz, the chair of the LICS steering committee, had a meeting with Neha Kumar and Jens Palsberg, the chairs of the SIG Governing Board. We reported the news about the consultation, and in particular, the unwillingness of our community to join SIGPLAN. They explained that we were not alone in this situation; there were other small SIGs that had also been struggling with the overhead increase and were unable to join other SIGs. As a consequence, the SGB started a working group that is planning to revise the overhead formula, with the express goal of helping small SIGs achieve financial stability. They said the current formula was devised to help cover the financial cost of ACM's transition toward open access. Since institutional support for the new ACM publication model is exceeding expectations, ACM can afford more flexibility toward the SIGs.

Neha invited the members of the SIGLOG Executive Committee to participate in the working group mentioned above. The goals of this group are not only to revise the SIGs' overhead formula, but also to rethink the overall SIG structure, and to devise a strategy to cover the publication fees of SIG authors whose institutions do not subscribe to the ACM open access program (authors from subscribing institutions are already covered, of course). We will begin these meetings in August, and hopefully, we will be able to report some good, concrete news soon. Stay tuned!

I would also like to take this opportunity to express our gratitude to everyone who subscribed to SIGLOG following our appeal several months ago. Your response has been very encouraging—our membership has grown from 185 to around 240. However, the numbers are now declining slowly, probably because older memberships are expiring and some people are not renewing. Hence, I would like to renew our appeal: a strong show of community support is vital as we work with ACM to ensure favorable terms for SIGLOG.

Catuscia Palamidessi
INRIA Saclay
ACM SIGLOG Chair

SECURITY AND PRIVACY COLUMN

STEVE KREMER,
LORIA, Inria Nancy-Grand Est, Université de Lorraine
Steve.Kremer@inria.fr



In 2008, Clarkson and Schneider introduced hyperproperties in their seminal paper. Classical trace properties are limited to reasoning about individual execution traces, whereas hyperproperties can reason about systems, that is sets of execution traces. In other words, a hyperproperty is a set of sets of traces. Hyperproperties were designed to provide a general theory that enables reasoning about *security policies*, which inherently require consideration of systems as a whole. Non-interference, introduced by Goguen and Meseguer in the early 80's, and its many variants, is a prominent example of a security policy that can be naturally expressed as a hyperproperty.

However, while providing a natural framework for reasoning about security policies, verifying hyperproperties is challenging. In particular, many security policies, including non-interference, are *hyperliveness* properties with a shape that can be described as *"for any trace, there exists a trace . . . "*. This quantifier alternation complicates verification significantly.

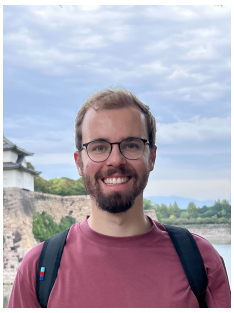
To address this problem, Beutner, Correnson, and Finkbeiner present three approaches:

— The first automata-theoretic approach involves complementing the automaton or, alternatively, deciding language inclusion. Although based on well-known techniques, this approach has high computational complexity.
— A second game-theoretic approach formulates the verification problem as a game between a refuter and verifier. Despite its incompleteness, it often succeeds in practice.
— The third approach uses coinduction: users interactively construct winning strategies. This method has already been formalized in the Rocq proof assistant.

This paper offers an excellent overview of three recent approaches for addressing the challenge of verifying hyperliveness, highlighting the distinct advantages and limitations of each. The authors are among the most significant contributors in the area of verification of hyperproperties and I am grateful they accepted to contribute this survey. I hope you'll enjoy the reading as much as I did.

Verifying ∀∃-Hyperliveness: Automata, Games, and Coinduction

Raven Beutner,
CISPA Helmholtz Center for Information Security



Arthur Correnson,
CISPA Helmholtz Center for Information Security



Bernd Finkbeiner,
CISPA Helmholtz Center for Information Security



Temporal ∀∃-hyperliveness properties, such as noninference and generalized noninterference, express requirements that relate multiple system executions through combinations of universal and existential quantification. Their verification poses fundamental challenges that go beyond those encountered in standard trace or k-hypersafety properties. This article surveys recent advances in the verification of ∀∃-hyperliveness properties, focusing on three complementary approaches: (1) automata-theoretic model checking via language inclusion, which offers completeness for finite-state systems; (2) game-based approximations that synthesize witness strategies with improved scalability; and (3) coinductive proof techniques that enable deductive reasoning using interactive proof assistants. We compare these methods in terms of their conceptual foundations and practical effectiveness.

1. INTRODUCTION

About two decades ago, Clarkson and Schneider introduced hyperproperties [Clarkson and Schneider 2008] as a general theory for system properties like noninterference, knowledge, and program refinement. Such properties cannot be characterized as properties of individual computations, but rather need to be judged with respect to the full set of computations. Generalizing trace properties, which are sets of infinite computation traces, Clarkson and Schneider definied hyperproperties as sets of sets of traces. Since hyperproperties refer to infinite computations, the notion applies to the general class of reactive systems, i.e., systems like communication protocols, operation and control systems, or web-based workflows that interact with their environment in potentially infinite executions. Standard specification languages for reactive systems, like temporal logic, describe the correctness of reactive systems as properties of individual execution traces; hyperproperties relate multiple traces. Hyperproperties are thus much more expressive. Specific hyperproperties have been studied in almost all

```
1: repeat
2:    h ← read_𝔹
3:    l ← read_𝔹
4:    o ← l
5:    print(o)
```

```
1: repeat
2:    h ← read_𝔹
3:    l ← read_𝔹
4:    if ⋆ then
5:       o ← h
6:    else
7:       o ← ¬h
8:    print(o)
```

Fig. 1: Two example programs. The program on the left satisfies *observational determinism*: every pair of traces that has the same low-security inputs l also has the same outputs o. The program on the right violates observational determinism but satisfies *noninference*: for every trace π_1 there exists a trace π_2 where the high-security input h is always false and yet π_1 and π_2 have the same outputs o.

branches of computer science: noninterference [Goguen and Meseguer 1982] in information flow security, epistemic properties in the design of distributed and multi-agent systems [Fagin et al. 2003; Halpern and Moses 1990], robustness in cyber-physical systems [Nguyen et al. 2017].

Much of the work on verification techniques for hyperproperties has focused on hypersafety properties, where counterexamples are guaranteed to consist of a finite set of traces. In particular, in k-hypersafety, we are guaranteed that all counterexamples are combinations of (at most) k traces. The key idea has been to analyze self-compositions of programs [Barthe et al. 2004; Terauchi and Aiken 2005]. A k-hypersafety property can be verified by checking a standard trace property of the k-fold self-composition of the given system. The implicit universal quantification over the traces is, however, an inherent limitation of this paradigm, which makes it difficult to abstract in an existential manner from phenomena like scheduling in concurrent programs, speculative execution, nondeterministic choice, or speed of execution in asynchronous computations. Alternations between quantifiers are furthermore essential for counterfactual reasoning about causation and blame.

As a simple example, consider the program shown on the left in Fig. 1. The program reads a high-security input h and a low-security input l, then assigns the low-security input to the variable o, which is then printed. The program is secure, because the output is independent of the high security input. This is captured by the notion of *observational determinism* [Zdancewic and Myers 2003], which can be expressed in the temporal logic HyperLTL as a formula with two universal trace quantifiers:

$$\forall \pi_1 . \forall \pi_2 . \; (o_{\pi_1} \leftrightarrow o_{\pi_2}) \; \mathbf{W} \; (l_{\pi_1} \not\leftrightarrow l_{\pi_2}) \tag{OD}$$

The formula states that every pair of traces π_1, π_2 must have the same outputs o *until* there is a difference in the low-security inputs. In other words, the program appears deterministic to a low-security user.

Now consider the program shown on the right of Fig. 1. Here, there is a nondeterministic choice between assigning o to h or its negation $\neg h$. The program is still secure, because one cannot derive the value of h from seeing the output o. However, observational determinism is violated, because there exists a pair of traces that agree on l but differ on o.

An information-flow property that captures the secrecy of programs with nondetermism is *noninference* [McLean 1994], which specifies that for every trace π_1 there exists a trace π_2 where all high-security inputs are replaced by a dummy input and where, nevertheless, the low-security output is the same as in π. In the following HyperLTL formula, we express noninference for a high-security input h, which we set to

false as dummy input, low-security input l, and output o:

$$\forall \pi_1 . \exists \pi_2 .\ \mathbf{G}\ (\neg h_{\pi_2} \wedge (l_{\pi_1} \leftrightarrow l_{\pi_2}) \wedge (o_{\pi_1} \leftrightarrow o_{\pi_2})) \tag{NI}$$

Observational determinism is a *2-hypersafety* property: Since both quantifiers in the formula are universal, any violation would manifest itself in a counterexample consisting of two traces that agree on l but differ on o. We can therefore verify observational determinism by checking a trace property on the self-composition of the given program, which combines two traces of the original program into a single trace of the self-composition.

By contrast, noninference is a *hyperliveness* property: a counterexample would consist of some universally chosen trace π_1 together with the full infinite set of traces that could potentially be chosen for the existential quantifier. Verifying noninference requires an effective resolution of the $\forall\exists$ quantifier combination. A standard technique for dealing with quantifier alternation is negation, i.e., to replace the $\forall \pi_1 . \exists \pi_2 .\ \varphi$ formula with the equivalent $\forall \pi_1 . \neg \forall \pi_2 .\ \neg \varphi$ formula. Reasoning about negation is expensive, however. In theory, the model checking problem of HyperLTL can be reduced to an emptiness check on an automaton that is constructed from the given system and the specificaton formula [Finkbeiner et al. 2015a]. The construction builds the automaton inductively, by first translating the quantifier-free subformula to an automaton, and then adding one quantifier at a time, starting with the innermost quantifier. If the quantifier is existential, the automaton is combined with another copy of the system, and the nondeterministic choices in this new copy are eliminated by projection. If the quantifier is universal, the problem is reduced to the case of the existential quantifier by complementing the automaton before and after the addition of the quantifier. However, the exponential cost of complementing the automaton creates a complexity-theoretic barrier for any such verification method. In practice, this has lead to a complete lack of implementations of the complementation-based approach. Nevertheless, practical verification methods exist.

In this paper, we discuss three major ideas that, despite the fundamental complexity of the problem, allow for the effective practical verification of $\forall\exists$-hyperliveness properties.

Automata. The first idea is to avoid complementation by reducing the verification problem of to *language containment* instead of language emptiness. Instead of computing an automaton for the formula $\forall \pi_1 . \exists \pi_2 .\ \varphi$, which requires complementation, we compute an automaton for $\exists \pi_2 .\ \varphi$, which can be obtained by inexpensive existential projection. The formula $\forall \pi_1 . \exists \pi_2 .\ \varphi$ is satisfied if the set of all traces of the system is *contained* in the language of the automaton for $\exists \pi_2 .\ \varphi$. In theory, the complexity of the language containment check is exactly the same as the complexity of complementation followed by an emptiness check. In practice, however, there are highly efficient algorithms and tools for language containment checking that avoid the construction of the complement automaton [Beutner and Finkbeiner 2025].

Games. The second idea is to approximate the satisfaction of the $\forall\exists$-hyperproperty as a *game* between two players, Refuter and Verifier. Refuter resolves the nondeterministic choices on the universally quantified trace, and Verifier responds by resolving the nonderministic choices on the existentially quantified traces. A winning strategy for Verifier is a witness for the existential quantifier. The game-based approach actually simplifies the problem: determining whether Verifier wins the game can be done algorithmically in polynomial time in the size of the system. The price for this simplification is the loss of completeness: While the resultion of the existential trace quantifier might, in general, depend on the resolution of future choices in the selection of the universally quantified trace, the strategy of Verifier has only access to choices that

happened in the past. In practice, however, future dependencies are rare, and can, when they do occur, be mitigated by (manually) adding prophecy variables [Coenen et al. 2019b; Beutner and Finkbeiner 2022b; Beutner and Finkbeiner 2022a].

Coinduction. The game-based view also provides the basis for a deductive proof technique where the existence of a wining strategy is established with the help of an interactive proof assistant. A naive approach would require the user to fix the strategy upfront and then verify that the strategy is indeed winning. In the deductive setting, finding the winning strategy is not only difficult; any error in the initial choice of strategy might furthermore only be revealed very late in the proof, requiring a restart of the proof from the very beginning. A practical proof technique should therefore be incremental, refining the instantiation of the existential quantifier on-the-fly as needed during the proof. One such proof technique with built-in support for incremental proofs is parameterized coinduction. The technique naturally generalizes to infinite-state software systems where (an approximation of) the winning region is not computed by an algorithm but rather defined by the user as part of the development of the proof. This coinduction-based approach precisely coincides with the game-based view in the sense that they can prove exactly the same properties. However, the former is much better suited for an interactive proof assistant [Correnson and Finkbeiner 2025].

The three ideas provide a trajectory towards more and more scalable verification. The automata-based approach has the great advantage of completeness, at least in the setting of finite-state model checking. The disadvantage is the exponential cost of complementation and language containment, even though the full cost can often be avoided in practice. Games provide a fundamental reduction in complexity and open the path for interactive techniques, because computation of the strategy is less expensive than complementation or language containment, and can be sidestepped altogether by providing the strategy manually. Parameterized coinduction, finally, allows for the full integration of this approach in a proof system supported by an interactive proof assistant.

2. SPECIFYING HYPERPROPERTIES

We use the temporal logic HyperLTL [Clarkson et al. 2014] to specify temporal hyperproperties. In the following, we briefly describe the syntax of the logic and describe its semantics with respect to sets of traces and transition systems.

HyperLTL. We fix a finite set AP of atomic propositions (APs). HyperLTL extends linear-time temporal logic (LTL) with explicit quantification over traces. Let $\mathcal{V} = \{\pi_1, \pi_2, \ldots\}$ be a set of trace variables. We define HyperLTL formulas by the following grammar:

$$\psi := a_\pi \mid \neg\psi \mid \psi \wedge \psi \mid \mathbf{X}\,\psi \mid \psi\,\mathbf{U}\,\psi$$
$$\varphi := \exists\pi.\,\varphi \mid \forall\pi.\,\varphi \mid \psi$$

where $\pi \in \mathcal{V}$ is a trace variable and $a \in \mathrm{AP}$ is an atomic proposition. We assume that the formula is closed, i.e., all trace variables that are used in the LTL body are bound by some quantifier. We use the usual derived Boolean constants and connectives $\mathit{true}, \mathit{false}, \vee, \rightarrow, \leftrightarrow$ and temporal operators $\mathbf{F}\,\psi := \mathit{true}\,\mathbf{U}\,\psi$, $\mathbf{G}\,\psi := \neg\mathbf{F}\,\neg\psi$, and $\psi_1\,\mathbf{W}\,\psi_2 := (\psi_1\,\mathbf{U}\,\psi_2) \vee \mathbf{G}\,\neg\psi_2$.

Semantics. A trace is an infinite sequence $t \in (2^{\mathrm{AP}})^\omega$. The semantics of HyperLTL is given with respect to a trace assignment $\Pi : \mathcal{V} \rightharpoonup (2^{\mathrm{AP}})^\omega$, which is a partial mapping from trace variables to traces. For $\pi \in \mathcal{V}$ and $t \in (2^{\mathrm{AP}})^\omega$, we write $\Pi[\pi \mapsto t]$ for the trace assignment obtained by updating the value of π to t. Given a set of traces $\mathbb{T} \subseteq (2^{\mathrm{AP}})^\omega$,

a trace assignment Π, and $i \in \mathbb{N}$, we define:

$$
\begin{array}{lll}
\Pi, i \models a_\pi & \text{iff} & a \in \Pi(\pi)(i) \\
\Pi, i \models \neg\psi & \text{iff} & \Pi, i \not\models \psi \\
\Pi, i \models \psi_1 \wedge \psi_2 & \text{iff} & \Pi, i \models \psi_1 \text{ and } \Pi, i \models \psi_2 \\
\Pi, i \models \mathbf{X}\,\psi & \text{iff} & \Pi, i+1 \models \psi \\
\Pi, i \models \psi_1\,\mathbf{U}\,\psi_2 & \text{iff} & \exists j \geq i.\, \Pi, j \models \psi_2 \text{ and } \forall i \leq k < j.\, \Pi, k \models \psi_1
\end{array}
$$

$$
\begin{array}{lll}
\Pi \models_{\mathbb{T}} \psi & \text{iff} & \Pi, 0 \models \psi \\
\Pi \models_{\mathbb{T}} \exists \pi.\, \varphi & \text{iff} & \exists t \in \mathbb{T}.\, \Pi[\pi \mapsto t] \models_{\mathbb{T}} \varphi \\
\Pi \models_{\mathbb{T}} \forall \pi.\, \varphi & \text{iff} & \forall t \in \mathbb{T}.\, \Pi[\pi \mapsto t] \models_{\mathbb{T}} \varphi
\end{array}
$$

The atomic formula a_π holds in the ith step, whenever a holds in ith step on the trace bound to π (i.e., $a \in \Pi(\pi)(i)$); Boolean and temporal operators are evaluated as expected; Quantification ranges over traces in $\mathbb{T}$ and binds them to the respective trace variable.

Examples. Typical examples for HyperLTL formulas are the *observational determinism* and *noninference* properties discussed in the introduction. *Generalized noninterference* [McCullough 1988] is similar to observational determinism in that it requires that the output (o) should not be influenced by high-security input (h). Unlike observational determinism, it allows, however, for nondeterministic behavior in the low-security input (l):

$$\forall\pi_1.\forall\pi_2.\exists\pi_3.\ \mathbf{G}\ ((h_{\pi_1} \leftrightarrow h_{\pi_3}) \wedge (l_{\pi_2} \leftrightarrow l_{\pi_3}) \wedge (o_{\pi_2} \leftrightarrow o_{\pi_3})) \tag{GNI}$$

The existentially quantified trace π_3 combines the high-security input of the universally quantified trace π_1 with the low-security input and output of the universally quantified trace π_2.

Hyperliveness properties also play an important role in applications beyond security. For example, *robust cleanness* [D'Argenio et al. 2017] specifies that significant differences in the output behavior are only permitted after significant differences in the input:

$$\forall\pi_1.\forall\pi_2.\exists\pi_3.\ \mathbf{G}\left(i_{\pi_2} = i_{\pi_3}\right)\ \wedge\ \left(\hat{d}(o_{\pi_1}, o_{\pi_3}) \leq \kappa_o\,\mathbf{W}\,\hat{d}(i_{\pi_1}, i_{\pi_3}) > \kappa_i\right) \tag{RC}$$

The differences are measured by a distance function $\hat{d}$ and compared to constant thresholds κ_i for the input and κ_o for the output. The formula specifies the existence of a trace π_3 that globally agrees with π_2 on the input and where the difference in the output o between π_1 and π_3 is bounded by κ_o unless the difference in the input i between π_1 and π_3 was greater than κ_i. Robust cleanness, thus, forbids unexpected jumps in the system behavior while allowing for behavioral differences due to nondeterminism.

Transition Systems.. We verify HyperLTL specifications with respect to system models given as transition systems.

Definition 2.1. A *transition system* is a a tuple $\mathcal{T} = (S, s_{init}, \rightarrow, L)$, where S is a finite set of states; $s_{init} \in S$ is a unique initial state; $\rightarrow \subseteq S \times S$ is a transition relation; and $L : S \rightarrow 2^{\mathrm{AP}}$ is a labeling function. $\triangle$

A path in $\mathcal{T}$ is an infinite sequence $s_0 s_1 s_2 \cdots \in S^\omega$, s.t., $s_0 = s_{init}$, and $(s_i, s_{i+1}) \in \rightarrow$ for all $i \in \mathbb{N}$. The associated trace is given by $L(s_0)L(s_1)L(s_2)\cdots \in (2^{\mathrm{AP}})^\omega$. We write $\mathit{Traces}(\mathcal{T}) \subseteq (2^{\mathrm{AP}})^\omega$ for the set of all traces generated by $\mathcal{T}$. A transition system $\mathcal{T}$ satisfies a HyperLTL formula φ, written $\mathcal{T} \models \varphi$, if $\emptyset \models_{\mathit{Traces}(\mathcal{T})} \varphi$, where $\emptyset$ denotes the empty trace assignment (i.e., the partial function $\mathcal{V} \rightharpoonup (2^{\mathrm{AP}})^\omega$ with empty domain).

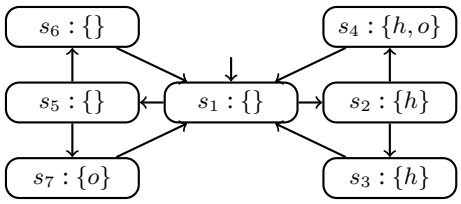


Fig. 2: A simplified transition system obtained from the right program in Figure 1

Example 2.2 (*Running Example*). As a running example, consider the transition system in Figure 2, modeling a simplified version of the right program in Figure 1. In the initial state s_1, we non-deterministically read a value for h by moving to s_2 and s_5. Afterward, we non-deterministically set the output o to h (by moving up, i.e., from s_2 to s_4 or from s_5 to s_6) or $\neg h$ (by moving down, i.e., from s_2 to s_3 or from s_5 to s_7). Throughout this paper, we will use the various methods to prove that this system satisfies noninference (NI). △

3. FINITE-STATE MODEL CHECKING

Given a HyperLTL property φ and a finite-state transition system $\mathcal{T} = (S, s_{init}, \rightarrow, L)$, model checking determines whether $\mathcal{T} \models \varphi$. In the following, we first describe a basic automata construction (explained in more detail in [Finkbeiner et al. 2015a]), which reduces the model-checking problem to the language emptiness problem of a Büchi automaton: the transition system satisfies the formula if and only if the language of the resulting automaton is empty. We then show how the this basic construction can be made more efficient in practice using language containment checking. This approach is implemented in the `AutoHyper` tool [Beutner and Finkbeiner 2025].

3.1. Automata-Based Model Checking

Automata. The construction is based on nondeterministic Büchi automata and their dual, universal co-Büchi automata.

Definition 3.1. A non-deterministic Büchi automaton (NBA) (resp. universal co-Büchi automaton, UCA) over some finite alphabet Σ is a tuple $\mathcal{A} = (Q, Q_0, \delta, F)$, where Q is a finite set of states, $Q_0 \subseteq Q$ is a set of initial states, $\delta \subseteq Q \times \Sigma \times Q$ is a transition relation, and $F \subseteq Q$ is a set of accepting (resp. rejecting) states. △

A *run* of $\mathcal{A}$ on a word $u \in \Sigma^\omega$ is an infinite sequence of states $q_0 q_1 q_2 \cdots \in Q^\omega$ such that $q_0 \in Q_0$ and for every $i \in \mathbb{N}$, $(q_i, u(i), q_{i+1}) \in \delta$. A word $u \in \Sigma^\omega$ is accepted by an NBA $\mathcal{A}$ if there exists *some* run on u that visits states in F *infinitely* many times. A word $u \in \Sigma^\omega$ is accepted by a UCA $\mathcal{A}$ if *all* runs on u visit states in F only *finitely* many times. Given an NBA or UCA $\mathcal{A}$, we write $\mathcal{L}(\mathcal{A}) \subseteq \Sigma^\omega$ for the set of infinite words accepted by $\mathcal{A}$. Note that we can translate NBAs into equivalent UCAs and vice versa with an exponential blowup using, e.g., automata complementation.

$\mathcal{T}$-Equivalence. The semantics of HyperLTL is based on trace assignments $\Pi : V \to (2^{\mathrm{AP}})^\omega$ for some $V \subseteq \mathcal{V}$ (cf. Section 2). To manipulate trace assignments using automata, we *zip* them into infinite traces.

Definition 3.2. For a trace assignment $\Pi : V \to (2^{\mathrm{AP}})^\omega$ (where $V \subseteq \mathcal{V}$ is the domains of the assignment), we define the trace $zip(\Pi) \in (2^{\mathrm{AP} \times V})^\omega$ by

$$zip(\Pi)(i) := \Big\{ (a, \pi) \mid a \in \mathrm{AP} \wedge \pi \in V \wedge a \in \Pi(\pi)(i) \Big\}$$

for each $i \in \mathbb{N}$. △

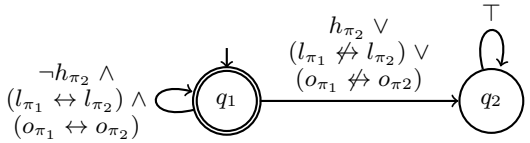


Fig. 3: A (deterministic) Büchi automaton for the LTL body of (NI). Accepting states are marked with a double circle.

That is, $(a, \pi) \in \mathrm{AP} \times V$ holds on $zip(\Pi)$ in the ith step iff a holds in the ith step on trace $\Pi(\pi)$. Note that zip defines a bijection between trace assignments $\Pi : V \to (2^{\mathrm{AP}})^\omega$ and traces in $(2^{\mathrm{AP} \times V})^\omega$. Using the bijection created by zip, we can use automata to summarize trace assignments that satisfy subformulas of φ. We formalize this using the concept of $\mathcal{T}$-equivalence [Finkbeiner et al. 2015a]:

Definition 3.3. Let φ be a HyperLTL formula with free trace variables $V \subseteq \mathcal{V}$. An automaton $\mathcal{A}$ over alphabet $2^{\mathrm{AP} \times V}$ is *$\mathcal{T}$-equivalent to* φ if for every trace assignments $\Pi : V \to (2^{\mathrm{AP}})^\omega$ we have $\Pi \models_{Traces(\mathcal{T})} \varphi$ if and only if $zip(\Pi) \in \mathcal{L}(\mathcal{A})$. △

That is, $\mathcal{A}$ accepts exactly the zippings of traces that constitute a satisfying trace assignment for φ.

Product Construction. To check if $\mathcal{T} \models \varphi$, we inductively construct an automation $\mathcal{A}_\varphi$ that is $\mathcal{T}$-equivalent to each subformula of φ. For the (quantifier-free) LTL body of φ, we can construct this automaton via a standard LTL-to-NBA construction [Baier and Katoen 2008].

Example 3.4. In Figure 3 we depict a NBA that is $\mathcal{T}$-equivalent to (NI)'s LTL body $\mathbf{G}\ (\neg h_{\pi_2} \wedge (l_{\pi_1} \leftrightarrow l_{\pi_2}) \wedge (o_{\pi_1} \leftrightarrow o_{\pi_2}))$. △

We then, iteratively, eliminate quantifiers by computing the product with the given system $\mathcal{T}$ (to eliminate trace quantifiers).

— **Case** $\varphi' = \exists \pi. \varphi$**.** We are given an inductively constructed automaton $\mathcal{A}_\varphi = (Q, Q_0, \delta, F)$ over alphabet $2^{\mathrm{AP} \times (V \uplus \{\pi\})}$ for some $V \subseteq \mathcal{V}$ that is $\mathcal{T}$-equivalent to φ. We ensure that $\mathcal{A}_\varphi$ is an NBA (by possibly translating a UCA into an NBA via complementation) and define the NBA $\mathcal{A}_{\varphi'}$ over alphabet $2^{\mathrm{AP} \times V}$ as

$$\mathcal{A}_{\varphi'} := (S \times Q, \{s_{init}\} \times Q_0, \delta', S \times F),$$

where $\delta' \subseteq (S \times Q) \times 2^{\mathrm{AP} \times V} \times (S \times Q)$ is defined as

$$\delta' := \Big\{ \big((s,q), \sigma, (s',q')\big) \mid s \to s' \wedge \Big(q, \sigma \uplus \big\{(a,\pi) \mid a \in L(s)\big\}, q'\Big) \in \delta \Big\}.$$

Intuitively, $\mathcal{A}_{\varphi'}$ reads the zipping of a trace assignment $V \to (2^{\mathrm{AP}})^\omega$; guesses a trace in $\mathcal{T}$ for trace variable π; and simulates the zipping of the extended trace assignment $(V \uplus \{\pi\}) \to (2^{\mathrm{AP}})^\omega$ in $\mathcal{A}_\varphi$ by adding the π-indexed guessed trace ($\big\{(a,\pi) \mid a \in L(s)\big\}$) to each letter $\sigma \in 2^{\mathrm{AP} \times V}$.

— **Case** $\varphi' = \forall \pi. \varphi$**.** We are given an automaton $\mathcal{A}_\varphi$ over alphabet $2^{\mathrm{AP} \times (V \uplus \{\pi\})}$ that is $\mathcal{T}$-equivalent to φ. We ensure that this automaton is a UCA (by possibly translating an NBA into a UCA via complementation) and define $\mathcal{A}_{\varphi'}$ as the UCA over $2^{\mathrm{AP} \times V}$ that is syntactically identical to the NBA constructed in the previous case.

THEOREM 3.5. *For every formula φ, the automaton $\mathcal{A}_\varphi$ is $\mathcal{T}$-equivalent to φ.*

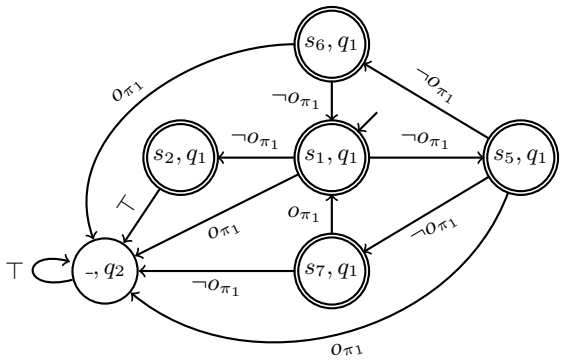


Fig. 4: Product of the TS in Figure 2 with the NBA in Figure 3

As the final formula φ is closed (i.e., contains no free variables), we eventually obtain a $\mathcal{T}$-equivalent automaton $\mathcal{A}_\varphi$ over the singleton alphabet $2^{\mathrm{AP}\times\emptyset} = 2^\emptyset$. By definition of $\mathcal{T}$-equivalence, we then have $\mathcal{T} \models \varphi$ iff $\emptyset \models_{\mathit{Traces}(\mathcal{T})} \varphi$ iff $\mathit{zip}(\emptyset) \in \mathcal{L}(\mathcal{A}_\varphi)$. Checking if $\mathcal{T} \models \varphi$ thus reduces to a word containment check for $\mathit{zip}(\emptyset)$ in $\mathcal{A}_\varphi$ (or, equivalently, as the alphabet of $\mathcal{A}_\varphi$ is a singleton set we can check if $\mathcal{L}(\mathcal{A}_\varphi)$ is non-empty).

Example 3.6. As an example, we consider the TS in Figure 2 and the (NI) property. The NBA $\mathcal{A}$ depicted Figure 3 recognizes the LTL body of (NI), i.e., it is $\mathcal{T}$-equivalent to subformula $\mathbf{G}\ (\neg h_{\pi_2} \wedge (l_{\pi_1} \leftrightarrow l_{\pi_2}) \wedge (o_{\pi_1} \leftrightarrow o_{\pi_2}))$. In Figure 4, we depict the NBA obtained with the above construction for subformula $\exists \pi_2.\, \mathbf{G}\ (\neg h_{\pi_2} \wedge (l_{\pi_1} \leftrightarrow l_{\pi_2}) \wedge (o_{\pi_1} \leftrightarrow o_{\pi_2}))$. For simplicity, we merged all states where the automaton reached (the non-accepting sink state) q_2. In the next step, we would translate this NBA to a UCA (e.g., by complementation). We then, again, compute the product of the resulting UCA with the TS, resulting a UCA over alphabet $2^\emptyset$. In this case, the automaton is non-empty; the system in Figure 2 satisfies (NI). $\triangle$

3.2. Leveraging Language Inclusion

The algorithm outlined above requires one complementation for each quantifier alternation in the HyperLTL formula. While we cannot avoid the theoretical cost of this complementation (see the lower-bound from [Rabe 2016; Clarkson et al. 2014]), we can reduce to a, in practice, more tamable problem: *language inclusion*.

For a transition system $\mathcal{T}$ and a finite set of trace variables $V \subseteq \mathcal{V}$, we define an NBA $\mathcal{A}_{\mathcal{T},V}$ over alphabet $2^{\mathrm{AP}\times V}$ that accepts the AP evaluations of all possible trace combinations in $\mathcal{T}$. That is, for any trace assignment $\Pi : V \to (2^{\mathrm{AP}})^\omega$, we have that $\mathit{zip}(\Pi) \in \mathcal{L}(\mathcal{A}_{\mathcal{T},V})$ iff $\Pi(\pi) \in \mathit{Traces}(\mathcal{T})$ for all $\pi \in V$. We can easily construct $\mathcal{A}_{\mathcal{T},V}$ by construction a $|V|$-fold self-composition of $\mathcal{T}$ [Barthe et al. 2011]:

Definition 3.7. Define the NBA $\mathcal{A}_{\mathcal{T},V}$ over $2^{\mathrm{AP}\times V}$ by $\mathcal{A}_{\mathcal{T},V} := (Q, Q_0, \delta, F)$, where

— $Q := (V \to S)$, i.e., the state-space consists of all functions $V \to S$ (which is a finite set as V and S are both finite), so each state $\vec{q} \in Q$ tracks a state $\vec{q}(\pi) \in S$ for all trace variables $\pi \in V$;
— $Q_0 := \{\lambda \pi \in V.\, s_{\mathit{init}}\}$, i.e., in the initial state all traces $\pi \in V$ start in the initial state s_{init};

— $\delta \subseteq Q \times 2^{\mathrm{AP} \times V} \times Q$ is defined by

$$\delta := \Big\{ (\vec{q}, \sigma, \vec{q}') \mid \forall \pi \in V.\, \vec{q}(\pi) \to \vec{q}'(\pi) \wedge \sigma = \big\{ (a, \pi) \mid a \in \mathrm{AP}, \pi \in V, a \in L\big(\vec{q}(\pi)\big) \big\} \Big\},$$

i.e., there is a transition from $\vec{q}$ to $\vec{q}'$ if, for each $\pi \in V$, the state tracked for π is updated according on some transition in $\mathcal{T}$ (i.e., $\vec{q}(\pi) \to \vec{q}'(\pi)$). The unique label of this transition contains exactly those indexed APs (a, π) where a holds in the state assigned to π (i.e., $a \in L\big(\vec{q}(\pi)\big)$);

— $F := Q$, i.e., we mark all states as accepting. △

We can now state a formal connection between language inclusion and HyperLTL model-checking [Beutner and Finkbeiner 2023].

THEOREM 3.8. *Let $\varphi = \forall \pi_1 \dots \forall \pi_n . \varphi'$ be a HyperLTL formula (where φ' may contain additional trace quantifiers) and let $\mathcal{A}_{\varphi'}$ be an automaton over $2^{\mathrm{AP} \times \{\pi_1, \dots, \pi_n\}}$ that is $\mathcal{T}$-equivalent to φ'. Then $\mathcal{T} \models \varphi$ if and only if $\mathcal{L}(\mathcal{A}_{\mathcal{T}, \{\pi_1, \dots, \pi_n\}}) \subseteq \mathcal{L}(\mathcal{A}_{\varphi'})$.*

We can use Theorem 3.8 to avoid a complementation for the outermost quantifier alternation.

Example 3.9. We, again, verify that the system in Figure 2 satisfies (NI). In Figure 4 we already computed an NBA $\mathcal{A}$ that is $\mathcal{T}$-equivalent to $\exists \pi_2 .\, \mathbf{G}\, (\neg h_{\pi_2} \wedge (l_{\pi_1} \leftrightarrow l_{\pi_2}) \wedge (o_{\pi_1} \leftrightarrow o_{\pi_2}))$. Importantly, we can construct this automaton in linear time in the size of $\mathcal{T}$. Using Theorem 3.8, we can then check if the language of the TS in Figure 2 (formally, its 1-fold self-composition indexed by trace variable π_1) is contained in the language of $\mathcal{A}$. As expected, the language containment holds, so $\mathcal{T}$ satisfies (NI). △

Note that complementation and subsequent emptiness check is a theoretically optimal method to check for language inclusion. Theorem 3.8 thus offers no asymptotic advantages over complementation-based model-checking (cf. Section 3.1). In *practice*, constructing an explicit complemented automaton is often unnecessary as language inclusion or non-inclusion might be witnessed without a complete complementation [Duret-Lutz et al. 2022; Doveri et al. 2021; Clemente and Mayr 2019; Doveri et al. 2022; Li et al. 2020].

4. GAME-BASED APPROXIMATION

An effective approach to the verification of $\forall\exists$-hyperproperties, that avoids the exponential cost of complementation or language inclusion checking, is to interpret the verification problem as a game betweeen *Refuter*, who selects the universally quantified traces, and *Verifier*, who selects the existentially quantified traces in response. For a $\forall \pi_1 . \exists \pi_2 . \psi$ formula, the refuter controls the universally quantified trace by moving through the system (thereby producing a trace π_1) while the verifier iteratively reacts with moves in a separate copy of the system (thereby producing a trace π_2). The objective of the verifier is to ensure that π_1 and π_2, together, satisfy the LTL body ψ [Coenen et al. 2019b; Beutner and Finkbeiner 2022a; Beutner and Finkbeiner 2024b].

Büchi Games. We formalize the interaction between Refuter and Verifier as a Büchi game. A Büchi game is a tuple $\mathcal{G} = (V_{Verifier}, V_{Refuter}, v_{init}, T, F)$ where $V := V_{Verifier} \uplus V_{Refuter}$ is the finite set of vertices. The vertices in $V_{Verifier}$ are controlled by the verifier and those in $V_{Refuter}$ are controlled by the refuter. $v_{init} \in V$ is a unique initial vertex, $T \subseteq V \times V$ is the transition relation (we assume that for each v there is at least one v' with $(v, v') \in T$), and $F \subseteq V$ is a set of accepting vertices. A positional strategy for the verifier is a function $\sigma : V_{Verifier} \to V$ such that for every $v \in V_{Verifier}$, $(v, \sigma(v)) \in T$. A play in $\mathcal{G}$ is an infinite sequence $r \in V^\omega$ such that for every i, $(r(i), r(i+1)) \in T$. The play r is compatible with strategy σ if for every i where $r(i) \in V_{Verifier}$ we have

$$\frac{s_1 \to s_1' \qquad \big(q, s_1 \otimes s_2, q'\big) \in \delta_\psi}{\langle s_1, s_2, q, \forall\rangle \to \langle s_1', s_2, q', \exists\rangle} \ (\forall)$$

$$\frac{s_2 \to s_2'}{\langle s_1, s_2, q, \exists\rangle \to \langle s_1, s_2', q, \forall\rangle} \ (\exists)$$

Fig. 5: Transition rules for the game-based synthesis of winning strategies for the $\exists$-player.

that $r(i+1) = \sigma(r(i))$. We say that player *Verifier* wins a vertex $v \in V$ if there exists a strategy σ such that every play that starts in v and is compatible with σ visits vertices in F infinitely many times. We write $wins(\mathit{Verifier}, \mathcal{G})$ if the verifier wins the initial vertex v_{init}.

Model Checking As a Game. The idea of the model checking approach is that the Büchi game mimic the iterative trace construction of both players. Assume we are given a system $\mathcal{T} = (S, S_0, \to, L)$ and a HyperLTL formula $\varphi = \forall\pi_1.\exists\pi_2.\,\psi$, i.e., for simplicity we consider a property with a single universal and single existential quantifier. To track the LTL body ψ during the game, we translate it to a *deterministic* Büchi automaton (DBA) over alphabet $= 2^{\mathrm{AP}\times\{\pi_1,\pi_2\}}$. Formally, a DBA is a NBA $\mathcal{A}_\psi = (Q_\psi, \{q_{0,\psi}\}, \delta_\psi, F_\psi)$ with a unique initial state $q_{0,\psi}$ and a transition relation $\delta_\psi \subseteq S_\psi \times 2^{\mathrm{AP}\times\{\pi_1,\pi_2\}} \times S_\psi$, s.t., for each $s \in S_\psi$ and $\sigma \in 2^{\mathrm{AP}\times\{\pi_1,\pi_2\}}$, there exists a ***unique*** $s' \in S_\psi$ with $(s, \sigma, s') \in \delta_\psi$.[1]

Definition 4.1. Given two states s_1, s_2 for π_1, π_2, respectively, we define the corresponding indexed letter $s_1 \otimes s_2 \in 2^{\mathrm{AP}\times\{\pi_1,\pi_2\}}$ as

$$s_1 \otimes s_2 := \big\{(a, \pi_1) \mid a \in L(s_1)\big\} \cup \big\{(a, \pi_2) \mid a \in L(s_2)\big\}$$

i.e., we take all APs that hold in s_1 (resp. s_2) and index them with π_1 (resp. π_2). $\triangle$

The vertices in game $\mathcal{G}_{\mathcal{T},\varphi}$ are defined as $V_{\mathit{Verifier}} := \big\{\langle s_1, s_2, q, \exists\rangle \mid s_1, s_2 \in S, q \in Q_\psi\big\}$ and $V_{\mathit{Refuter}} := \big\{\langle s_1, s_2, q, \forall\rangle \mid s_1, s_2 \in S, q \in Q_\psi\big\}$. The vertices of the game thus have the form $\langle s_1, s_2, q, \flat\rangle$, where $s_1, s_2 \in S$ gives the current state of the two system copies for π_1, π_2, respectively, $q \in Q_\psi$ is the current state of the DBA tracking ψ, and $\flat \in \{\forall, \exists\}$ defines whether the universal ($\flat = \forall$) or existential ($\flat = \exists$) copies move next. We start the game in $\langle s_{init}, s_{init}, q_{0,\psi}, \forall\rangle$, i.e., both systems start in $\mathcal{T}$'s initial state, $\mathcal{A}_\psi$ start in initial state $q_{0,\psi}$, and the refuter makes the first move. The transitions T of the game are defined via the rules in Fig. 5. In the ($\forall$)-rule the universally quantified copy for π_1 is updated by moving to successor states within $\mathcal{T}$. Simultaneously, we update the automaton state of $\mathcal{A}_\psi$, i.e., we transition to the (unique) automaton state, reached when reading letter $s_1 \otimes s_2$ (cf. Definition 4.1), i.e., the indexed atomic proposition of the s_1 and s_2 state. Similarly, in the ($\exists$)-rule, the existentially quantified copy for π_2 are updated. We define the set of accepting vertices as $F := \big\{\langle s_1, s_2, q, \flat\rangle \mid q \in F_\psi\big\}$. The final game is thus defined as $\mathcal{G}_{\mathcal{T},\varphi} := (V_{\mathit{Verifier}}, V_{\mathit{Refuter}}, \langle s_{init}, s_{init}, q_{0,\psi}, \forall\rangle, T, F)$.

The game $\mathcal{G}_{\mathcal{T},\varphi}$ mimics the strategic behavior of the $\exists$-player. In each step, the refuter chooses successors for the universally quantified trace π_1, followed by the verifier

[1]Note that we cannot translate every LTL formula into an equivalent DBA. Instead, we can use a more expressive acceptance condition like parity; every LTL formula can be translated to a *deterministic* parity automaton. This would result in parity instead of Büchi games, which we can still solve effectively (see [Beutner and Finkbeiner 2022a] for details). In this article, we restrict to Büchi to allow a uniform presentation.

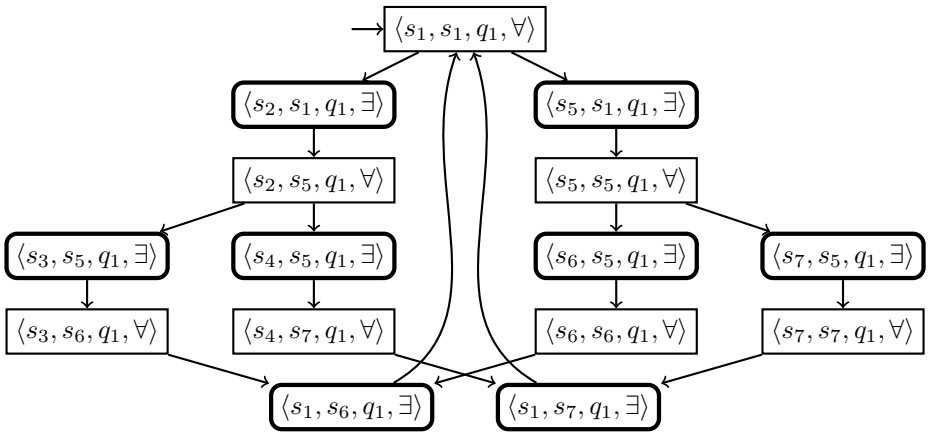


Fig. 6: Winning strategy for the verifier. Vertices controlled by the verifier are depicted in bold.

who selects a successor for the existentially quantified π_2. The automaton state in the vertices of $\mathcal{G}_{\mathcal{T},\varphi}$ tracks the (unique) run of $\mathcal{A}_\psi$ on the resulting pair of traces.

THEOREM 4.2. *If* $wins(Verifier, \mathcal{G}_{\mathcal{T},\varphi})$ *then* $\mathcal{T} \models \varphi$.

Example 4.3. We, again, verify that the TS $\mathcal{T}$ in Figure 2 satisfies (NI). In Figure 3, we already constructed a determinsitic Büchi automaton for (NI)'s LTL body. We depict a winning strategy for the verifier in game $\mathcal{G}_{\mathcal{T},(NI)}$ in Figure 6. For each vertex controlled by the refuter we include all possible successor vertices; for each vertex controlled by the verifier, we depict the unique successor chosen by the strategy. Initially, no matter where the refuter moves the copy for π_1, the verifier always moves the π_2-copy to state s_5 as it has to ensure $\mathbf{G}\,\neg h_{\pi_2}$. Afterward, the verifier mimics the output decision of the refuter, for example if the refuter moves π_1 to s_4 or s_7, the verfier moves π_2 to s_7 to ensure $\mathbf{G}\,(o_{\pi_1} \leftrightarrow o_{\pi_2})$. △

Incompleteness. Strategy-based verification is incomplete because the existential player must choose the next state after observing only a finite prefix of the universal traces. That is, the decisions of the verifier in the ith stage of the game, can only depend on the first i moves of the verifier. In contrast, in the semantics of a $\forall\pi_1.\exists\pi_2.\psi$ formula, the choice of π_2 can depend on the entire trace π_1. We can counteract this lack of information, by explicitly providing the verifier with information about the *future* behavior of the refuter using prophecies [Abadi and Lamport 1991]. In fact, for every finite-state system we can always find finitely many prophecies that ensure that the verifier wins the game iff the property holds [Beutner and Finkbeiner 2022a].

Abstraction, Asynchronous Hyperproperties, and Infinite State. The game-based approximations also paves the way towards the verification of infinite-state systems. While the automata-based methods from Section 3 are inherently limited to finite-state systems, there exist many approaches that solve games over infinite game-graphs [Farzan and Kincaid 2018; Baier et al. 2021; Walker and Ryzhyk 2014; Beyene et al. 2014; Faella and Parlato 2023; Heim and Dimitrova 2024]. Moreover, the game-based approximation works well when combined with abstractions. For example, we can use a set of relational predicates and construct the game over abstract game states. Moreover – especially when reasoning about software – we are not interested in every step along some trace, but rather in observable events (like prints), i.e., we are interested

in hyperproperties that compare the traces *asynchronously* [Baumeister et al. 2021]. We can incorporate this idea in our game by letting the verifier control the *speed* of the trace, only aligning them at the observed locations. We refer to [Beutner and Finkbeiner 2022b] for details.

5. DEDUCTIVE REASONING VIA COINDUCTION

The game-based approach to the automated verification of $\forall\exists$-hyperproperties offers an efficient (albeit incomplete) alternative to automata-based verification. Coincidentally, it also provides a basis for an interactive deductive proof technique. Instead of explicitly constructing a 2-player game and solving it algorithmically, we can devise a deductive system to prove the existence of a winning strategy. The key idea is to observe that, when the body ψ is a safety formula, the winning region can be described as a greatest fixed point. Consequently, we can prove the existence of a wining strategy by coinduction (i.e., by exhibiting a safe fragment of the wining region containing the initial states of the game). Combined with the framework of *parameterized coinduction* [Hur et al. 2013], this observation leads to an elegant proof method for interactive deductive verification, where an approximation of the wining region is progressively discovered and refined as needed.

5.1. The Case of Relational Invariants

To explain the coinduction-based approach, we first focus on verifying systems against HyperLTL formulas of the form $\varphi = \forall\pi_1.\exists\pi_2.\ \mathbf{G}\,\psi$, where ψ is free of temporal modalities (i.e., ψ is a Boolean formula over $\mathrm{AP} \times \{\pi_1, \pi_2\}$). Given such a formula and a transition system $\mathcal{T} = (S, s_{init}, \to, L)$, we start by defining the following coinductive relation $\mathtt{win}^{\mathcal{T}}_{\psi}$.

Definition 5.1. For a (non-temporal) assertion ψ over atomic propositions in AP labeled with trace variables in $\{\pi_1, \pi_2\}$ we define:

$$\mathtt{winF}^{\mathcal{T}}_{\psi}(R) \triangleq \{\ (s_1, s_2) \mid s_1 \otimes s_2 \models \psi \wedge \forall s_1 \to s_1'.\exists s_2 \to s_2'.\ (s_1', s_2') \in R\ \}$$
$$\mathtt{win}^{\mathcal{T}}_{\psi} \triangleq \nu R.\ \mathtt{winF}^{\mathcal{T}}_{\psi}(R)$$

where we write $s_1 \otimes s_2 \models \psi$ if the assignment $s_1 \otimes s_2 \subseteq \mathrm{AP} \times \{\pi_1, \pi_2\}$ (cf. Definition 4.1) satisfies ψ. △

The intuition behind this definition is the following: two states $s_1, s_2 \in S$ are in $\mathtt{win}^{\mathcal{T}}_{\psi}$ if their labels are not immediately violating the relational invariant ψ. Further, we demand that for any successor s_1' of s_1, there must exists a successor s_2' of s_2 such that, again, $(s_1', s_2') \in \mathtt{win}^{\mathcal{T}}_{\psi}$.

Fixpoint vs. Winning Region. It turns out that $\mathtt{win}^{\mathcal{T}}_{\psi}$ characterizes exactly the winning region of the game constructed in Section 4. That is, assume we consider the game $\mathcal{G}_{\mathcal{T},\varphi}$ from Section 4 for formula $\varphi = \forall\pi_1.\exists\pi_2.\mathbf{G}\,\psi$. First observe that we can express $\mathbf{G}\,\psi$ using a Büchi automaton with a single state q (we stay in q as long as ψ holds). In this case, $\mathtt{win}^{\mathcal{T}}_{\psi}$ characterizes all states from which the verifier wins $\mathcal{G}_{\mathcal{T},\varphi}$:

THEOREM 5.2. *The verifier wins game $\mathcal{G}_{\mathcal{T},\varphi}$ from vertex $\langle s_1, s_2, q, \forall\rangle$ if and only if $(s_1, s_2) \in \mathtt{win}^{\mathcal{T}}_{\psi}$.*

Similar to the game-based approach, the greatest fixed point thus gives a sound approximate method to verify $\mathcal{T} \models \varphi$.

THEOREM 5.3. *If $(s_{init}, s_{init}) \in \mathtt{win}^{\mathcal{T}}_{\psi}$, then $\mathcal{T} \models \varphi$.*

$$\textsc{Init}\ \frac{\boxed{\emptyset} \vdash_{\texttt{inv}} {s_{init}}^{\forall} \mid {s_{init}}^{\exists}\ \{\ \psi\ \}}{\mathcal{T} \models \forall\pi_1.\exists\pi_2.\mathbf{G}\,\psi} \qquad \textsc{Cycle}\ \frac{(s_1, s_2) \in H}{\boxed{\!\!\lceil H \rceil\!\!} \vdash_{\texttt{inv}} {s_1}^{\forall} \mid {s_2}^{\exists}\ \{\ \psi\ \}} \qquad \textsc{Guard}\ \frac{\boxed{H} \vdash_{\texttt{inv}} {s_1}^{\forall} \mid {s_2}^{\exists}\ \{\ \psi\ \}}{\boxed{\!\!\lceil H \rceil\!\!} \vdash_{\texttt{inv}} {s_1}^{\forall} \mid {s_2}^{\exists}\ \{\ \psi\ \}}$$

$$\textsc{Step}\ \frac{s_1 \otimes s_2 \models \psi \quad \forall s_1 \to s_1'.\ \exists s_2 \to s_2'.\ \boxed{\!\!\lceil H \rceil\!\!} \vdash_{\texttt{inv}} {s_1'}^{\forall} \mid {s_2'}^{\exists}\ \{\ \psi\ \}}{\boxed{H} \vdash_{\texttt{inv}} {s_1}^{\forall} \mid {s_2}^{\exists}\ \{\ \psi\ \}} \qquad \textsc{Invariant}\ \frac{(s_1, s_2) \in H' \quad \forall (s_1', s_2') \in H'.\ \boxed{H \cup H'} \vdash_{\texttt{inv}} {s_1}^{\forall} \mid {s_2}^{\exists}\ \{\ \psi\ \}}{\boxed{H} \vdash_{\texttt{inv}} {s_1'}^{\forall} \mid {s_2'}^{\exists}\ \{\ \psi\ \}}$$

Fig. 7: Incremental reasoning rules for relational invariants

Coinduction. Since $\texttt{win}_\psi^{\mathcal{T}}$ is defined as the greatest fixed point of $\texttt{winF}_\psi^{\mathcal{T}}$, we can prove membership by coinduction: to prove that a pair of states (s_1, s_2) is contained in $\texttt{win}_\varphi^{\mathcal{T}}$, it suffices to find a postfixed point of $\texttt{winF}_\varphi^{\mathcal{T}}$ containing (s_1, s_2).

THEOREM 5.4. *We have* $(s_1, s_2) \in \texttt{win}_\varphi^{\mathcal{T}}$ *iff* $\exists R.\ (s_1, s_2) \in R \wedge R \subseteq \texttt{winF}_\varphi^{\mathcal{T}}(R)$.

Combined with Theorem 5.3, this theorem gives a simple technique to prove $\mathcal{T} \models \varphi$ by exhibiting a relational invariant R. In practice, however, finding a suitable relational invariant up-front might be tedious. Instead, we can leverage the framework of parameterized coinduction [Hur et al. 2013] to incrementally construct a solution. The main idea is to parameterize the greatest fixed point construction of Definition 5.1 by a *hypothesis* H representing a partial solution. At the beginning of a proof, we start with $H = \emptyset$. The hypothesis is then progressively enlarged, as needed. To achieve this style of incremental proof, we start by defining the following *proof quadruples*:

$$\boxed{H} \vdash_{\texttt{inv}} {s_1}^{\forall} \mid {s_2}^{\exists}\ \{\ \psi\ \} \triangleq (s_1, s_2) \in \nu R.\ \texttt{winF}_\psi^{\mathcal{T}}(R \cup H)$$
$$\boxed{\!\!\lceil H \rceil\!\!} \vdash_{\texttt{inv}} {s_1}^{\forall} \mid {s_2}^{\exists}\ \{\ \psi\ \} \triangleq (s_1, s_2) \in H \cup \nu R.\ \texttt{winF}_\psi^{\mathcal{T}}(R \cup H)$$

In these quadruples, s_1, s_2 are two states of $\mathcal{T}$, and $H \subseteq S \times S$ is a relation between states which should intuitively be understood as being a known subset of the wining region. The subtlety is that H is not necessarily a subset of the wining region, but rather a *guess* about which pairs of states are part of it. This subtlety is reflected in the choice of notation: $\boxed{H}$ represents a *guarded* guess (i.e., we are not sure yet that it is a subset of the wining region), whereas $\boxed{\!\!\lceil H \rceil\!\!}$ represents an *unguarded* guess (i.e., it can be used to immediately conclude a proof if $(s_1, s_2) \in H$). Note that, by definition, when $H = \emptyset$ we have

$$(s_1, s_2) \in \texttt{win}_\psi^{\mathcal{T}} \iff \boxed{\emptyset} \vdash_{\texttt{inv}} {s_1}^{\forall} \mid {s_2}^{\exists}\ \{\ \psi\ \}$$

In turn, to prove $\mathcal{T} \models \forall\pi_1.\exists\pi_2.\ \mathbf{G}\,\psi$, it suffices to show $\boxed{\emptyset} \vdash_{\texttt{inv}} {s_1}^{\forall} \mid {s_2}^{\exists}\ \{\ \psi\ \}$. To prove the validity of such a quadruple, one can then use the incremental reasoning rules presented in Fig. 7.

The rule INIT exploits the soundness of $\texttt{win}_\psi^{\mathcal{T}}$ to initiate a proof by parameterized coinduction. Initially, the hypothesis is set to be the empty relation. The rule STEP enables making progress by proving two things. First, the current states s_1, s_2 should satisfy the desired invariant ψ. Second, for every transition $s_1 \to s_1'$, there must exists a corresponding transition $s_2 \to s_2'$ such that s_1' and s_2' are again connected by a quadruple. Importantly, using the rule STEP releases the guard around the current

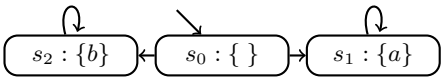


Fig. 8: Example transition system

hypothesis. CYCLE allows us to finish a proof by creating a cycle: if the hypothesis H is unguarded, we can conclude a proof if the current pair of states is contained in H. Finally, the rule INVARIANT extends the current hypothesis H with a larger guess $H \cup H'$, where H' contains at least the current pair of states. The cost to pay is that applying INVARIANT requires to prove a quadruple for every state in $H \cup H'$ instead of just the initial pair of states. The benefit is that, after applying the rule INVARIANT, we can conclude a proof as soon as we reach again a pair of states in H'. We note that the rule INVARIANT can always be used to *accumulate* the current pair of states without generating any additional verification conditions. This is reflected by the following derived rule Acc.

$$\textsc{Acc}\ \frac{\boxed{H \cup \{\ (s_1, s_2)\ \}} \vdash_{\mathtt{inv}} {s_1}^{\forall} \mid {s_2}^{\exists}\ \{\ \psi\ \}}{\boxed{H} \vdash_{\mathtt{inv}} {s_1}^{\forall} \mid {s_2}^{\exists}\ \{\ \psi\ \}}$$

Example 5.5. To demonstrate the usage of this deductive system, let us consider the simple transition system $\mathcal{T}$ in Figure 8 over $\mathrm{AP} = \{a, b\}$. Suppose we wish to prove $\mathcal{T} \models \forall \pi_1 . \exists \pi_2 .\ \mathbf{G}\,(a_{\pi_1} \leftrightarrow b_{\pi_2})$. We give an incremental proof using the rules of Fig. 7. We first use the rule INIT to initiate a proof by parameterized coinduction.

$$\begin{aligned} & \mathcal{T} \models \forall \pi_1 . \exists \pi_2 .\ \mathbf{G}\,(a_{\pi_1} \leftrightarrow b_{\pi_2}) \\ \Longleftarrow\ & \boxed{\emptyset} \vdash_{\mathtt{inv}} {s_0}^{\forall} \mid {s_0}^{\exists}\ \{\ a_{\pi_1} \leftrightarrow b_{\pi_2}\ \} \qquad \text{(by INIT)} \end{aligned}$$

Since $L(s_0) = \{\ \}$, the invariant $a_{\pi_1} \leftrightarrow b_{\pi_2}$ holds initially, and we can use the rule STEP to explore the possible outgoing transitions from the initial state s_0. We match the transition $s_0 \to s_1$ with the transition $s_0 \to s_2$, and vice versa.

$$\begin{aligned} & \boxed{\emptyset} \vdash_{\mathtt{inv}} {s_0}^{\forall} \mid {s_0}^{\exists}\ \{\ a_{\pi_1} \leftrightarrow b_{\pi_2}\ \} \\ \Longleftarrow\ & \lceil\!\boxed{\emptyset}\!\rfloor \vdash_{\mathtt{inv}} {s_1}^{\forall} \mid {s_2}^{\exists}\ \{\ a_{\pi_1} \leftrightarrow b_{\pi_2}\ \} \wedge \lceil\!\boxed{\emptyset}\!\rfloor \vdash_{\mathtt{inv}} {s_2}^{\forall} \mid {s_1}^{\exists}\ \{\ a_{\pi_1} \leftrightarrow b_{\pi_2}\ \} \qquad \text{(by STEP)} \end{aligned}$$

Then, we observe that the only way out of the pair (s_1, s_2) (resp. (s_2, s_1)) is to go back to (s_1, s_2) (resp. (s_2, s_1)). We therefore use the rule ACC to accumulate $H_1 = \{\ (s_1, s_2)\ \}$ and $H_2 = \{\ (s_2, s_1)\ \}$.

$$\begin{aligned} & \lceil\!\boxed{\emptyset}\!\rfloor \vdash_{\mathtt{inv}} {s_1}^{\forall} \mid {s_2}^{\exists}\ \{\ a_{\pi_1} \leftrightarrow b_{\pi_2}\ \} \wedge \lceil\!\boxed{\emptyset}\!\rfloor \vdash_{\mathtt{inv}} {s_2}^{\forall} \mid {s_1}^{\exists}\ \{\ a_{\pi_1} \leftrightarrow b_{\pi_2}\ \} \\ \Longleftarrow\ & \boxed{H_1} \vdash_{\mathtt{inv}} {s_1}^{\forall} \mid {s_2}^{\exists}\ \{\ a_{\pi_1} \leftrightarrow b_{\pi_2}\ \} \wedge \boxed{H_2} \vdash_{\mathtt{inv}} {s_2}^{\forall} \mid {s_1}^{\exists}\ \{\ a_{\pi_1} \leftrightarrow b_{\pi_2}\ \} \\ & \qquad \text{(by GUARD and INVARIANT)} \end{aligned}$$

Since both pairs (s_1, s_2) and (s_2, s_1) satisfy the invariant $a_{\pi_1} \leftrightarrow b_{\pi_2}$, we can use the rule STEP to cycle back to (s_1, s_2) and (s_2, s_1). Taking a step releases the guard, so we can then conclude with CYCLE.

$$\begin{aligned} & \boxed{H_1} \vdash_{\mathtt{inv}} {s_1}^{\forall} \mid {s_2}^{\exists}\ \{\ a_{\pi_1} \leftrightarrow b_{\pi_2}\ \} \wedge \boxed{H_2} \vdash_{\mathtt{inv}} {s_2}^{\forall} \mid {s_1}^{\exists}\ \{\ a_{\pi_1} \leftrightarrow b_{\pi_2}\ \} \\ \Longleftarrow\ & \lceil\!\boxed{H_1}\!\rfloor \vdash_{\mathtt{inv}} {s_1}^{\forall} \mid {s_2}^{\exists}\ \{\ a_{\pi_1} \leftrightarrow b_{\pi_2}\ \} \wedge \lceil\!\boxed{H_2}\!\rfloor \vdash_{\mathtt{inv}} {s_2}^{\forall} \mid {s_1}^{\exists}\ \{\ a_{\pi_1} \leftrightarrow b_{\pi_2}\ \} \qquad \text{(by STEP)} \\ \Longleftarrow\ & (s_1, s_2) \in H_1 \wedge (s_2, s_1) \in H_2 \qquad \text{(by CYCLE)} \end{aligned}$$

△

Note that in the previous example, the winning region is $\{ (s_0, s_0), (s_1, s_2), (s_2, s_1) \}$, and we never had to entirely spell it out over the course of the proof. Instead, it was sufficient to identify smaller and disjoint subsets of it (namely $\{(s_1, s_2)\}$ and $\{(s_2, s_1)\}$).

5.2. Beyond Relational Invariants

So far, we have focused on the special case $\forall\pi_1.\exists\pi_2.\mathbf{G}\,\psi$ where ψ is free of temporal modalities. The approach can be generalized to an arbitrary *safety* formula ψ by utilizing a safety automaton for ψ. Formally, a deterministic safety automaton (DSA) is a deterministic Büchi automaton $\mathcal{A} = (Q, \{q_0\}, \delta, F)$ (cf. Section 4) where all non-accepting states are selfloops. This implies that once we visit a non-accepting state, the run cannot be accepted; we can interpret $Q \setminus F$ as the set of *bad* states. We assume, w.l.o.g., that $q_0 \in F$, i.e., the initial state is not immediately losing. The automaton in Figure 3 is a DSA.

As in Section 4, we now assume that ψ – the LTL body of φ – is accepted by a DSA $\mathcal{A}_\psi = (Q_\psi, \{q_{0,\psi}\}, \delta_\psi, F_\psi)$ over alphabet $2^{\mathrm{AP}\times\{\pi_1,\pi_2\}}$. Given a system $\mathcal{T}$, and the DSA $\mathcal{A}_\psi$, we generalize the binary relation of Theorem 5.1 to a ternary relation between two states of $\mathcal{T}$ and a state of $\mathcal{A}_\varphi$. Formally, we define

$$\begin{aligned}\mathtt{winF}^{\mathcal{T}}_{\mathcal{A}_\psi}(R) \triangleq \{\ (s_1, s_2, q) \mid \forall s_1 \to s_1'.\exists s_2 \to s_2'.\exists q' \in F_\psi.\\ (q, s_1 \otimes s_2, q') \in \delta_\psi \wedge (s_1', s_2', q') \in R\ \}\\ \mathtt{win}^{\mathcal{T}}_{\mathcal{A}_\psi} \triangleq \nu R.\ \mathtt{winF}^{\mathcal{T}}_{\mathcal{A}_\psi}(R)\end{aligned}$$

As a proof technique for $\forall\exists$-hyperproperties, $\mathtt{win}^{\mathcal{T}}_{\mathcal{A}_\psi}$ generalizes Theorem 5.1 in the sense of the following theorem.

THEOREM 5.6. *If* $(s_{init}, s_{init}, q_{0,\psi}) \in \mathtt{win}^{\mathcal{T}}_{\mathcal{A}_\psi}$ *then* $\mathcal{T} \models \forall\pi_1.\exists\pi_2.\ \psi$.

As before, we can build a deductive system on top of $\mathtt{win}^{\mathcal{T}}_{\psi}$. We redefine our semantic quadruples as follows:

$$\begin{aligned}\boxed{H} \vdash_{\mathcal{A}_\psi} {s_1}^{\forall} \mid {s_2}^{\exists}\ \{\ q\ \} &\triangleq (s_1, s_2, q) \in \nu R.\ \mathtt{winF}^{\mathcal{T}}_{\mathcal{A}_\psi}(R \cup H)\\ \lceil\!\lceil \bar{H} \rfloor\!\rfloor \vdash_{\mathcal{A}_\psi} {s_1}^{\forall} \mid {s_2}^{\exists}\ \{\ q\ \} &\triangleq (s_1, s_2, q) \in H \cup \nu R.\ \mathtt{winF}^{\mathcal{T}}_{\mathcal{A}_\psi}(R \cup H)\end{aligned}$$

The associated reasoning rules are described in Figure 9.

Example 5.7. Let us apply these reasoning principles to verify that the running example Figure 2 satisfies (NI). Note that the LTL body of (NI) is an invariant and could thus be proven using the rules in Figure 7; we use the more general rules in Figure 9 for demonstration. Figure 3 already displays a DSA $\mathcal{A}_\psi$ for (NI)'s body. The proof starts by exploring the system from s_1, s_1, q_1, and we immediately accumulate this triple of states. Define $H := \{\ (s_1, s_1, q_1)\ \}$:

$$\begin{aligned}&\mathcal{T} \models \forall\pi_1.\exists\pi_2.\ \mathbf{G}\left(\neg h_{\pi_2} \wedge (l_{\pi_1} \leftrightarrow l_{\pi_2}) \wedge (o_{\pi_1} \leftrightarrow o_{\pi_2})\right)\\ \Longleftarrow\ &\boxed{\emptyset} \vdash_{\mathcal{A}_\psi} {s_1}^{\forall} \mid {s_1}^{\exists}\ \{\ q_1\ \} && \text{(by INIT)}\\ \Longleftarrow\ &\boxed{H} \vdash_{\mathcal{A}_\psi} {s_1}^{\forall} \mid {s_1}^{\exists}\ \{\ q_1\ \} && \text{(by INVARIANT)}\\ \Longleftarrow\ &\lceil\!\lceil \bar{H} \rfloor\!\rfloor \vdash_{\mathcal{A}_\psi} {s_2}^{\forall} \mid {s_5}^{\exists}\ \{\ q_1\ \} \wedge \lceil\!\lceil \bar{H} \rfloor\!\rfloor \vdash_{\mathcal{A}_\psi} {s_5}^{\forall} \mid {s_5}^{\exists}\ \{\ q_1\ \} && \text{(by STEP)}\end{aligned}$$

Here, as in the strategy in Figure 6, we matched both universal transitions $s_1 \to s_2$ and $s_1 \to s_5$ with the transition $s_1 \to s_5$. This generates two new proof quadruples. For

$$\textsc{Init}\ \frac{\boxed{\emptyset} \vdash_{\mathcal{A}_\psi} {s_{init}}^{\forall} \mid {s_{init}}^{\exists}\ \{\ q_{0,\psi}\ \}}{\mathcal{T} \models \forall\pi_1.\exists\pi_2.\psi} \qquad \textsc{Cycle}\ \frac{(s_1, s_2, q) \in H}{\lceil\!\lceil \bar{H} \rceil\!\rceil \vdash_{\mathcal{A}_\psi} {s_1}^{\forall} \mid {s_2}^{\exists}\ \{\ q\ \}} \qquad \textsc{Guard}\ \frac{\boxed{H} \vdash_{\mathcal{A}_\psi} {s_1}^{\forall} \mid {s_2}^{\exists}\ \{\ q\ \}}{\lceil\!\lceil \bar{H} \rceil\!\rceil \vdash_{\mathcal{A}_\psi} {s_1}^{\forall} \mid {s_2}^{\exists}\ \{\ q\ \}}$$

$$\textsc{Step}\ \frac{\forall s_1 \to s_1'.\exists s_2 \to s_2'.\exists q' \in F_\psi.\ (q, s_1 \otimes s_2, q') \in \delta_\psi \wedge \lceil\!\lceil \bar{H} \rceil\!\rceil \vdash_{\mathcal{A}_\psi} {s_1'}^{\forall} \mid {s_2'}^{\exists}\ \{\ q'\ \}}{\boxed{H} \vdash_{\mathcal{A}_\psi} {s_1}^{\forall} \mid {s_2}^{\exists}\ \{\ q\ \}}$$

$$\textsc{Invariant}\ \frac{(s_1, s_2, q) \in H' \qquad \forall (s_1, s_2, q) \in H'.\ \boxed{H \cup H'} \vdash_{\mathcal{A}_\psi} {s_1}^{\forall} \mid {s_2}^{\exists}\ \{\ q\ \}}{\boxed{H} \vdash_{\mathcal{A}_\psi} {s_1}^{\forall} \mid {s_2}^{\exists}\ \{\ q\ \}}$$

Fig. 9: Incremental reasoning rules for arbitrary safety property ψ

demonstration, we focus on the first one.

$$\begin{aligned} & \lceil\!\lceil \bar{H} \rceil\!\rceil \vdash_{\mathcal{A}_\psi} {s_2}^{\forall} \mid {s_5}^{\exists}\ \{\ q_1\ \} \\ \Longleftarrow\ & \lceil\!\lceil \bar{H} \rceil\!\rceil \vdash_{\mathcal{A}_\psi} {s_4}^{\forall} \mid {s_7}^{\exists}\ \{\ q_1\ \} \wedge \lceil\!\lceil \bar{H} \rceil\!\rceil \vdash_{\mathcal{A}_\psi} {s_3}^{\forall} \mid {s_6}^{\exists}\ \{\ q_1\ \} \qquad \text{(by GUARD and STEP)} \end{aligned}$$

I.e., we match $s_2 \to s_4$ with transition $s_5 \to s_7$ and match $s_2 \to s_3$ with $s_5 \to s_6$. We again focus on the first conjunct. By matching transition $s_4 \to s_1$ with $s_7 \to s_1$ we cycle back to (s_1, s_1, q_1), allowing us to conclude.

$$\begin{aligned} & \lceil\!\lceil \bar{H} \rceil\!\rceil \vdash_{\mathcal{A}_\psi} {s_4}^{\forall} \mid {s_7}^{\exists}\ \{\ q_1\ \} \\ \Longleftarrow\ & \lceil\!\lceil \bar{H} \rceil\!\rceil \vdash_{\mathcal{A}_\psi} {s_1}^{\forall} \mid {s_1}^{\exists}\ \{\ q_1\ \} && \text{(by GUARD and STEP)} \\ \Longleftarrow\ & (s_1, s_1, q_1) \in H && \text{(by CYCLE)} \end{aligned}$$

△

5.3. Extension to Programs

The coinductive approach to the verification of $\forall\exists$-hyperproperties has been formalized in the Rocq proof assistant. This enables using the proof assistant as an IDE to develop large proofs. It also gives confidence in the correctness of the approach: a full formal proof of the soundness of the deductive system is provided, and every proof developed using the system can be machine-checked.

Unlike the simplified version of the deductive system we presented here, the Rocq formalization also supports reasoning about arbitrary transition systems, including systems with an infinite state-space. In particular, this enables to verify programs written in any programming language, so long as a formal operational semantics is provided. For example, [Correnson and Finkbeiner 2025] presents a version of the deductive system targeting the verification of imperative programs with non-determinism and I/O.

The system presented in this paper and the Rocq formalization also differ in the way safety specifications are handled. Here, in the interest of simplicity, we presented a *semantic* approach based on a DSA for the LTL body ψ. In the Rocq formalization, we instead use a more *syntactic* approach based on the notion of *property derivatives*. In essence, derivatives interpret the subformulas of ψ as the state space of a DSA for ψ itself. This technique enables a more syntax-driven style of reasoning, as is more commonly found in program logics.

6. CONCLUSIONS

The effective resolution of quantifier alternations is a fundamental challenge in the verification of hyperliveness properties. With automata-theoretic model checking, game-based approximation, and deductive reasoning via coinduction, we have surveyed three fundamental ideas for addressing this challenge.

While the automata-theoretic approach is faithful to the original semantics of the logic, the game-based approach, which also underlies the deductive reasoning, introduces a new interpretation of the quantifiers where we *constructively* find a strategy that resolves the existential choice based on the preceding universal choices. In a model checking approach, this change greatly simplifies the verification problem; in a deductive proof system, it furthermore provides an opportunity for the user to support the verification effort by providing the strategy as part of the proof.

Recently, the game-based view has also lead to the development of new logics that distinguish explicitly between quantification over traces and quantification over strategies. HyperATL* [**?**] is an extension of computation tree logic with path variables and strategy quantifiers that can express can express *strategic hyperproperties*, such as that the scheduler in a concurrent system has a strategy to avoid information leakage. Hyper Strategy Logic (HyperSL) [Beutner and Finkbeiner 2024a] is an even more expressive strategy logic where the outcome of multiple strategy profiles can be compared by relating multiple paths. This allows for reasoning about concepts like quantitative Nash equilibria, optimal adversarial planning, and strategies with imperfect information.

By distinguishing quantification over traces from quantification over strategies, the strategic hyperlogics improve the semantic precision in the specification of hyperliveness properties. When integrated into deductive proof systems, such logics would also allow for approximating one type of quantifier by another as an explicit step in the proof, instead of as an implicit assumption that underlies the proof system.

ACKNOWLEDGMENTS

This work was partially supported by the European Research Council (ERC) grant HYPER (No. 101055412).

REFERENCES

Martín Abadi and Leslie Lamport. 1991. The Existence of Refinement Mappings. *Theor. Comput. Sci.* 82, 2 (1991), 253–284. DOI: http://dx.doi.org/10.1016/0304-3975(91)90224-P

Christel Baier, Norine Coenen, Bernd Finkbeiner, Florian Funke, Simon Jantsch, and Julian Siber. 2021. Causality-Based Game Solving. In *International Conference on Computer Aided Verification, CAV 2021 (Lecture Notes in Computer Science)*, Vol. 12759. Springer, 894–917. DOI: http://dx.doi.org/10.1007/978-3-030-81685-8_42

Christel Baier and Joost-Pieter Katoen. 2008. *Principles of model checking*. MIT press.

Gilles Barthe, Pedro R. D'Argenio, and Tamara Rezk. 2004. Secure Information Flow by Self-Composition.. In *CSFW*. 100–114.

Gilles Barthe, Pedro R. D'Argenio, and Tamara Rezk. 2011. Secure information flow by self-composition. *Math. Struct. Comput. Sci.* 21, 6 (2011). DOI: http://dx.doi.org/10.1017/S0960129511000193

Jan Baumeister, Norine Coenen, Borzoo Bonakdarpour, Bernd Finkbeiner, and César Sánchez. 2021. A Temporal Logic for Asynchronous Hyperproperties. In *International Conference on Computer Aided Verification, CAV 2021 (Lecture Notes in Computer Science)*, Vol. 12759. Springer, 694–717. DOI: http://dx.doi.org/10.1007/978-3-030-81685-8_33

Raven Beutner and Bernd Finkbeiner. 2021. A Temporal Logic for Strategic Hyperproperties. In *32nd International Conference on Concurrency Theory, CONCUR 2021, August 24-27, 2021, Virtual Conference (LIPIcs)*, Serge Haddad and Daniele Varacca (Eds.), Vol. 203. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 24:1–24:19. DOI: http://dx.doi.org/10.4230/LIPIcs.CONCUR.2021.24

Raven Beutner and Bernd Finkbeiner. 2022a. Prophecy Variables for Hyperproperty Verification. In *Computer Security Foundations Symposium, CSF 2022*. DOI: http://dx.doi.org/10.1109/CSF54842.2022.00030

Raven Beutner and Bernd Finkbeiner. 2022b. Software Verification of Hyperproperties Beyond k-Safety. In *International Conference on Computer Aided Verification, CAV 2022*. DOI: http://dx.doi.org/10.1007/978-3-031-13185-1_17

Raven Beutner and Bernd Finkbeiner. 2023. AutoHyper: Explicit-State Model Checking for HyperLTL. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2023*. DOI: http://dx.doi.org/10.1007/978-3-031-30823-9_8

Raven Beutner and Bernd Finkbeiner. 2024a. Hyper Strategy Logic. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS '24)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 189–197.

Raven Beutner and Bernd Finkbeiner. 2024b. Non-deterministic Planning for Hyperproperty Verification. In *International Conference on Automated Planning and Scheduling, ICAPS 2024*. AAAI Press, 25–30. DOI: http://dx.doi.org/10.1609/ICAPS.V34I1.31457

Raven Beutner and Bernd Finkbeiner. 2025. AutoHyper: leveraging language inclusion checking for hyperproperty model-checking. *International Journal on Software Tools for Technology Transfer* (2025), 1–17. DOI: http://dx.doi.org/10.1007/s10009-025-00801-5

Tewodros A. Beyene, Swarat Chaudhuri, Corneliu Popeea, and Andrey Rybalchenko. 2014. A constraint-based approach to solving games on infinite graphs. In *Symposium on Principles of Programming Languages, POPL 2014*. ACM, 221–234. DOI: http://dx.doi.org/10.1145/2535838.2535860

Michael R. Clarkson, Bernd Finkbeiner, Masoud Koleini, Kristopher K. Micinski, Markus N. Rabe, and César Sánchez. 2014. Temporal Logics for Hyperproperties. In *International Conference on Principles of Security and Trust, POST 2014*. DOI: http://dx.doi.org/10.1007/978-3-642-54792-8_15

Michael R. Clarkson and Fred B. Schneider. 2008. Hyperproperties. In *2008 21st IEEE Computer Security Foundations Symposium*. 51–65. DOI: http://dx.doi.org/10.1109/CSF.2008.7

Lorenzo Clemente and Richard Mayr. 2019. Efficient reduction of nondeterministic automata with application to language inclusion testing. *Log. Methods Comput. Sci.* 15, 1 (2019). DOI: http://dx.doi.org/10.23638/LMCS-15(1:12)2019

Norine Coenen, Bernd Finkbeiner, César Sánchez, and Leander Tentrup. 2019a. Verifying Hyperliveness. In *Computer Aided Verification - 31st International Conference, CAV 2019, Part I (Lecture Notes in Computer Science)*, Isil Dillig and Serdar Tasiran (Eds.), Vol. 11561. Springer, 121–139. DOI: http://dx.doi.org/10.1007/978-3-030-25540-4_7

Norine Coenen, Bernd Finkbeiner, César Sánchez, and Leander Tentrup. 2019b. Verifying Hyperliveness. In *International Conference on Computer Aided Verification, CAV 2019*. DOI: http://dx.doi.org/10.1007/978-3-030-25540-4_7

Arthur Correnson and Bernd Finkbeiner. 2025. Coinductive Proofs for Temporal Hyperliveness. *Proc. ACM Program. Lang.* 9, POPL, Article 53 (Jan. 2025), 28 pages. DOI: http://dx.doi.org/10.1145/3704889

Pedro R. D'Argenio, Gilles Barthe, Sebastian Biewer, Bernd Finkbeiner, and Holger Hermanns. 2017. Is Your Software on Dope? - Formal Analysis of Surreptitiously "enhanced" Programs. In *European Symposium on Programming, ESOP 2017*. DOI: http://dx.doi.org/10.1007/978-3-662-54434-1_4

Kyveli Doveri, Pierre Ganty, and Nicolas Mazzocchi. 2022. FORQ-Based Language Inclusion Formal Testing. In *International Conference on Computer Aided Verification, CAV 2022*. DOI: http://dx.doi.org/10.1007/978-3-031-13188-2_6

Kyveli Doveri, Pierre Ganty, Francesco Parolini, and Francesco Ranzato. 2021. Inclusion Testing of Büchi Automata Based on Well-Quasiorders. In *International Conference on Concurrency Theory, CONCUR 2021*. DOI: http://dx.doi.org/10.4230/LIPIcs.CONCUR.2021.3

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. 2022. From Spot 2.0 to Spot 2.10: What's New?. In *International Conference on Computer Aided Verification, CAV 2022*. DOI: http://dx.doi.org/10.1007/978-3-031-13188-2_9

Marco Faella and Gennaro Parlato. 2023. Reachability Games Modulo Theories with a Bounded Safety Player. In *Conference on Artificial Intelligence, AAAI 2023*. AAAI Press, 6330–6337. DOI: http://dx.doi.org/10.1609/AAAI.V37I5.25779

R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. 2003. *Reasoning about Knowledge*. MIT Press, Cambridge, Mass.

Azadeh Farzan and Zachary Kincaid. 2018. Strategy synthesis for linear arithmetic games. *Proc. ACM Program. Lang.* 2, POPL (2018), 61:1–61:30. DOI: http://dx.doi.org/10.1145/3158149

Bernd Finkbeiner, Markus N. Rabe, and César Sánchez. 2015a. Algorithms for Model Checking HyperLTL and HyperCTL*. In *International Conference on Computer Aided Verification, CAV 2015*. DOI: http://dx.doi.org/10.1007/978-3-319-21690-4_3

Bernd Finkbeiner, Markus N. Rabe, and César Sánchez. 2015b. Algorithms for Model Checking HyperLTL and HyperCTL*. In *CAV 2015 (Part I) (Lecture Notes in Computer Science)*, Daniel Kroening and Corina S. Pasareanu (Eds.), Vol. 9206. Springer, 30–48. DOI: http://dx.doi.org/10.1007/978-3-319-21690-4_3

Joseph A. Goguen and José Meseguer. 1982. Security Policies and Security Models. In *IEEE Symposium on Security and Privacy*. 11–20.

J. Y. Halpern and Y. Moses. 1990. Knowledge and Common Knowledge in a Distributed Environment. *J. ACM* 37, 3 (1990), 549–587.

Philippe Heim and Rayna Dimitrova. 2024. Solving Infinite-State Games via Acceleration. *Proc. ACM Program. Lang.* 8, POPL (2024), 1696–1726. DOI: http://dx.doi.org/10.1145/3632899

Chung-Kil Hur, Georg Neis, Derek Dreyer, and Viktor Vafeiadis. 2013. The power of parameterization in coinductive proof. In *Proceedings of the 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '13)*. Association for Computing Machinery, New York, NY, USA, 193–206. DOI: http://dx.doi.org/10.1145/2429069.2429093

Yong Li, Andrea Turrini, Xuechao Sun, and Lijun Zhang. 2020. Proving Non-inclusion of Büchi Automata Based on Monte Carlo Sampling. In *International Symposium on Automated Technology for Verification and Analysis, ATVA 2020*. DOI: http://dx.doi.org/10.1007/978-3-030-59152-6_26

Daryl McCullough. 1988. Noninterference and the composability of security properties. In *Proc. IEEE Symposium on Security and Privacy*. 177–186.

John McLean. 1994. A General Theory of Composition for Trace Sets Closed Under Selective Interleaving Functions. In *Proc. IEEE Symposium on Security and Privacy*. 79–93.

Luan Viet Nguyen, James Kapinski, Xiaoqing Jin, Jyotirmoy V. Deshmukh, and Taylor T. Johnson. 2017. Hyperproperties of Real-Valued Signals. In *Proceedings of the 15th ACM-IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE '17)*. Association for Computing Machinery, New York, NY, USA, 104–113. DOI: http://dx.doi.org/10.1145/3127041.3127058

Markus N. Rabe. 2016. *A temporal logic approach to Information-flow control*. Ph.D. Dissertation. Saarland University.

Tachio Terauchi and Alexander Aiken. 2005. Secure Information Flow as a Safety Problem. In *Static Analysis, 12th International Symposium, SAS 2005, London, UK, September 7-9, 2005, Proceedings*. 352–367. DOI: http://dx.doi.org/10.1007/11547662_24

Adam Walker and Leonid Ryzhyk. 2014. Predicate abstraction for reactive synthesis. In *Formal Methods in Computer-Aided Design, FMCAD 2014*. IEEE, 219–226. DOI: http://dx.doi.org/10.1109/FMCAD.2014.6987617

S. Zdancewic and A. C. Myers. 2003. Observational Determinism for Concurrent Program Security. In *Proc. of CSFW'03*.

Conference Reports

JOOST-PIETER KATOEN, Aachen University
katoen@cs.rwth-aachen.de



This edition of the SIGLOG Newsletter features a conference report on the 40th issue of the IEEE/ACM Annual Conference on Logic in Computer Science (LICS). LICS is the flagship conference of SIGLOG and its 2025-edition was held in Singapore.

I hope the LICS report is an enjoyable and informative read. I'd like to thank the LICS 2025 PC co-chairs Barbara König and Lars Birkedal for providing their reflections on LICS 2025. LICS 2026 will return to Europe and will be part of the Federated Logic Conference (FLOC) in Lisbon.

In case you want to report on an event that you visited and that might be relevant to the SIGLOG community, please reach out to me.

Enjoy the summer.

Report on Logic in Computer Science (LICS '25) conference

Lars Birkedal,
Aarhus University, Denmark



Barbara König,
University of Duisburg-Essen, Germany



Th 40th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), took place at the National University of Singapore (NUS), Singapore, from June 23 to June 26, 2025. LICS 2025 was co-located with the Logic Mentoring Workshop that ran in parallel on June 23.

The conference was hosted in the School of Computing (SoC) at NUS where we were also provided with delicious coffee and lunch and where the reception took place on the first day of the conference. The Banquet was served in the NUSS Kent Ridge Guild House and the conference concluded with an excursion to the Night Safari.

The LICS 2023 program included invited talks by Anuj Dawar (University of Cambridge, UK) on notions of width for logical formulae and and Hongseok Yang (School of Computing, KAIST, Korea) on Razborov's Flag Algebras as a computational method for solving problems in extremal combinatorics, as well as invited tutorials by Rustan Leino (Amazon Web Services) on an industrial perspective on tools for software verification and Christine Tasson (Sorbonne Université, France) on linear logic and probabilistic programming. The program further included 66 contributed papers which were selected from 215 submissions. LICS 2025 also hosted a poster session on Wednesday afternoon.

As usual, LICS 2025 was also the occasion to award several recognitions. This year the Kleene Award for Best Student Paper, sponsored by the European Association for Theoretical Computer Science (EATCS), went to the paper *"The Steenrod squares via unordered joins"* by Axel Ljungström and David Wärn.

Starting in 2021, around 10% of accepted LICS papers are selected as *distinguished papers*. In addition to the Kleene award winner the PC has chosen the following papers:

— *Pushdown Model Checking above the Cubic Bottleneck* by A. R. Balasubramanian, Dmitry Chistikov and Rupak Majumdar
— *Effectful Mealy Machines: Bisimulation and Trace* by Filippo Bonchi, Elena Di Lavore and Mario Román
— *The commutativity problem for effective varieties of formal series, and applications* by Lorenzo Clemente
— *Ordinal Exponentiation in Homotopy Type Theory* by Tom de Jong, Nicolai Kraus, Fredrik Nordvall Forsberg and Chuangjie Xu

— *The Identity Problem in virtually solvable matrix groups over algebraic numbers* by Ruiwen Dong and Corentin Bodart
— *The Yoneda embedding in simplicial type theory* by Daniel Gratzer, Jonathan Weinberger and Ulrik Buchholtz
— *The Blurred Drinker Paradox: Constructive Reverse Mathematics of the Downward Löwenheim-Skolem Theorem* by Dominik Kirst and Haoyi Zeng
— *Minimal History-Deterministic Co-Büchi Automata: Congruences and Passive Learning* by Christof Löding and Igor Walukiewicz
— *Hofmann–Streicher Lifting of Fibred Categories* by Andrew Slattery and Jonathan Sterling
— *Characterization and Decidability of FC-Definable Regular Languages* by Sam M. Thompson, Nicole Schweikardt and Dominik D. Freydenberger

To acknowledge the long-term, foundational nature of papers appearing in LICS, the Test-of-Time Award is given annually to the paper or papers that were published twenty years ago and that have "stood the test of time". This year, the Test-of-Time Award committee, consisting of Sandra Kiefer, Prakash Panangaden (chair) and Peter Selinger, selected two outstanding papers from the 20th Annual IEEE Symposium on Logic in Computer Science, Chicago, USA, 2005:

— *"On the Decidability of Metric Temporal Logic"* by Joël Ouaknine and James B. Worrell
— *"Existential Positive Types and Preservation under Homomorphisms"* by Benjamin Rossman

The recipients of the Test-of-Time Award gave talks about their work in the awards session.

For the first time the conference awarded a *distinguished reviewer* award that goes to Mario Román for his outstanding contributions to the reviewing process of LICS 2025.

Warmest congratulations to the authors of all distinguished works.

A heartfelt thanks goes to everyone who contributed to the success of LICS 2025. In particular to program committee members for their commitment, hard work, and flexibility in taking on last-minute tasks due to unexpected circumstances. Our warm thanks to the 302 external reviewers, to all the authors who submitted papers, as well as to the invited and tutorial speakers.

Special thanks go to Frank Stephan for his work as Conference Chair and local organizer and to his team, in particular to Ho Siew Foong. Additionally, we thank Igor Walukiewicz as General Chair, Valentin Blot as Workshop Chair, and Sam Staton as Publicity and Proceedings Chair.

The National University of Singapore hosted LICS 2025 at the School of Computing and the LICS 2025 organisers want to thank the NUS and SoC for all the help provided. The organisers of the Logic Mentoring Workshop appreciate the financial support provided by the National Science Foundation and Jane Street Capital.



Fig. 1. Gardens by the Bay, Singapore



Fig. 2. Impression from the Night Safari

CFP

SIGLOG MONTHLY 263

July 2025

ELLI ANASTASIADI, Aalborg University, SE



July 2025 edition of SIGLOG Monthly, featuring deadlines, calls and community announcements.

Past Issues - How to submit an announcement

1. TABLE OF CONTENTS

— DEADLINES (Section 2)

— CALLS

  — SIR@IWCS2025 (CALL FOR PAPERS) (Section 3)

  — PETER ACZEL MEMORIAL CONFERENCE & BRITISH LOGIC COLLOQUIUM 2025 (CALL FOR PAPERS) (Section 4)

  — REACTS'25 (CALL FOR PAPERS) (Section 5)

  — CPP 2026 (CALL FOR PAPERS) (Section 6)

  — FoSSaCS 2026 (CALL FOR PAPERS) (Section 7)

  — ICLP 2025 (CALL FOR PARTICIPATION) (Section 8)

  — WoLLIC 2025 (CALL FOR PARTICIPATION) (Section 9)

  — CONFEST 2025 (CALL FOR PARTICIPATION) (Section 10)

— JOB ANNOUNCEMENTS

  — Professor position (Section 11)

2. DEADLINES

ICLP 2025:	Jul 06, 2025 (Submission deadline)
SIR@IWCS2025:	Jul 14, 2025 (Paper deadline)
CSL 2026:	Jul 15, 2025 (Abstract), Jul 21, 2025 (Paper)
CONFEST 2025:	Jul 26, 2025 (Early registration)
PETER ACZEL MEMORIAL CONFERENCE & BRITISH LOGIC COLLOQUIUM 2025:	Jul 31, 2025 (Abstracts deadline)
REACTS'25:	Aug 13, 2025 (Abstract), Aug 20, 2025 (Paper)
CPP 2026:	Sep 05, 2025 (Abstract Submission Deadline), Sep 12, 2025 (Paper Submission Deadline)
FoSSaCS 2026:	Oct 16, 2025 (Submission deadline)
FM 2026:	Nov 25, 2025 (Abstract Submission), Dec 02, 2025 (Full Paper Submission)

3. SIR@IWCS2025: FIRST WORKSHOP ON SEMANTICS FOR INTERDISCIPLINARY RESEARCH

Dusseldorf, Germany
September 24 2025
https://team.inria.fr/semagramme/first-workshop-on-semantics-for-interdisciplinary-research/
https://openreview.net/group?id=inria.fr/INRIA/SC3A9magramme/2025/SIR01
CALL FOR PAPERS

— In recent years, Natural Language Processing (NLP) has increasingly intersected with the humanities and social sciences, offering new methodologies for analyzing textual data, interpreting meaning, and modelling language-based phenomena. The potential for multi-disciplinary research using NLP methods is particularly great in computational semantics (CS), as its ability to process and represent meaning opens up innovative pathways for researchers in history, philosophy, literary studies, political science, etc. This workshop aims to explore how semantic models and tools can be leveraged to tackle traditional and emerging questions in the Humanities in a broader sense (Social Sciences, Law, Economics, Management, Literature, Languages, Art, . . .). A major theme of SIR is the role of semantics in NLP applied to the humanities (both statistical and symbolic approaches).

— IMPORTANT DATES

Paper deadline:	July 14th (anywhere on earth)
Notification:	August 25th (anywhere on earth)
Camera Ready:	September 10th (anywhere on earth)
Workshop:	September 24th (anywhere on earth)

— Submission Information

Papers should describe original research and must not exceed 4 pages (with an extra page in the camera ready version for accepted papers). Papers should be submitted no later than 14 July 2025 (anywhere on earth). Accepted papers will be published in the conference proceedings in the ACL Anthology. For inclusion in the proceedings, at least one author must register to the conference and present the paper in person. Submissions should be fully anonymous to ensure double-blind reviewing. Submission via : https://openreview.net/group?id=inria.fr/INRIA/SC3A9magramme/2025/SIR01 . The workshop follow the IWCS 2025 template see the workshop web page.

— ORGANISERS

— Maxime Amblard, Université de Lorraine

— Ellen Breitholtz, Gothenburg University

— CONTACT

maxime.amblard@univ-lorraine.fr and ellen.breitholtz@ling.gu.se

4. PETER ACZEL MEMORIAL CONFERENCE & BRITISH LOGIC COLLOQUIUM 2025

10-12 September 2025, Manchester, UK
https://sites.google.com/view/blc2025/home
CALL FOR PAPERS

— The Peter Aczel Memorial Conference and the British Logic Colloquium 2025 meeting will be held at the University of Manchester (UK) on 10, 11, 12 September 2025.

— Registration is open at: https://estore.manchester.ac.uk/conferences-and-events/faculty-of-science-engineering/department-of-mathematics/departments-of-mathematics/british-logic-colloquium-2025-and-peter-aczel-memorial-conference

— The programme committee invites abstracts for Contributed Talks. Students and early-career researchers are especially encouraged to present their work.

— IMPORTANT DATES

Abstracts deadline: Jul 31, 2025

— Invited speakers for the Peter Aczel memorial conference (10 September):

— Steve Awodey (Carnegie Mellon University),

— Jouko Vaananen (University of Helsinki),

— Rosalie Iemhoff (Utrecht University),

— Andrew Swan (University of Ljubljana),

— Invited speakers for the BLC meeting (11-12 September) are:

— Mirna Dzamonja (Paris),

— Fairouz Kamareddine (Heriot-Watt),

— Fraser MacBride (Manchester),

— Paul-André Melliès (Paris),

— Paula Quinon (Lund/Warsaw),

— Katrin Tent (Münster),

— Frank Wolter (Liverpool)

— Additional information is available from: https://sites.google.com/view/blc2025/home

5. REACTS'25: INTERNATIONAL WORKSHOP ON RECONFIGURABLE TRANSITION SYSTEMS: SEMANTICS, LOGICS AND APPLICATIONS

https://reacts-workshop.github.io/2025/
Tuesday, 10-11 November 2025
Toledo, Spain
Satellite event of SEFM 2025 (https://sefm-conference.github.io/sefm2025)
CALL FOR PAPERS

— OVERVIEW

Reconfigurable Transition Systems (RTS) are dynamic relational structures (graphs) that evolve along its execution, in the sense that their accessibility relation, their set of nodes or their labelling change when their edges are crossed. These structures have proven to be suitable to compactly represent complex reactive and reconfigurable behaviours. Namely, the ability of reacting or readapting under the influence of certain events is a very distinctive feature of many diverse situations and objects. An autonomous vehicle that changes its route due to a new strike occurring, the behaviour of a software component after a memory disposal, or a DNA mutation as the result of a viral infection, are different examples that witness the importance of modelling about changes in a determined situation. Practical user cases have aroused the interest of the logic community in the study of variants of RTS, by developing formal methods to properly reason about such situations.

This workshop aims to bring together the whole community of researchers working on different ways to model reconfigurable and reactive systems from a formal perspective. This includes theoretical approaches (like hybrid logics, reactive frames, model-update logics, and topological and algebraic semantics), or formalisms designed for specific purposes (like separation logic in software verification, dynamic epistemic logic in AI planning, and others). Also, our goal is to devise novel approaches and potential applications, and share a common perspective on the discipline.

— SUBMISSION GUIDELINES

Authors are invited to submit, via CMT, research contributions or experience reports (https://cmt3.research.microsoft.com/reacts2025). All papers should be written in English and prepared using the specific LNCS templates available at http://www.springer.de/comp/lncs/authors.html. There are two categories of submissions:

— FULL PAPERS up to 12 pages (excluding references) – to present original research and the analysis, interpretation and validation of the research findings.

— SHORT PRESENTATIONS up to 4 pages (excluding references) – to present work in progress and preliminary results.

Both kinds of submissions allow system descriptions, to present a new tool, a new tool component or novel extensions to an existing tool aiming at supporting open community approaches, or the use/customisation of an existing tool in the context of RTS. Accepted full papers will be included in the workshop programme and will appear in the workshop LNCS proceedings. Accepted short presentations will be included in the pre-proceeding (available online before the Workshop) but not published in the LNCS proceedings.

— PROGRAM CO-CHAIRS

— José Proença, University of Porto (Portugal)

— Umberto Rivieccio, Universidad Nacional de Educación a Distancia (Spain)

— PUBLICATION

Accepted full papers will be published by Springer in a volume of Lecture Notes in Computer Science (http://www.springer.com/lncs), which will collect contributions to some workshops co-located with SEFM 2025. Condition for inclusion in proceedings is that at least one of the co-authors has presented the paper at the Workshop. Similarly to the last edition of ReacTS, we plan to invite authors of selected contributions to submit extended versions to a special issue, e.g. to the Journal of Applied Logics.

— IMPORTANT DATES

Abstract submission:	Aug 13, 2025
Paper submission:	Aug 20, 2025
Author notification:	Sep 22, 2025
Workshop:	Nov 10-11, 2025

— CONTACT

If you have any problems or questions, please contact us via e-mail at: jose.proenca@fc.up.pt / umberto@fsof.uned.es

6. CPP 2026: CERTIFIED PROGRAMS AND PROOFS

https://popl26.sigplan.org/home/CPP-2026
Rennes, France
12-13 January 2025
CALL FOR PAPERS

— Certified Programs and Proofs (CPP) is an international conference on practical and theoretical topics in all areas that consider formal verification and certification as an essential paradigm for their work. CPP spans areas of computer science, mathematics, logic, and education. CPP 2026 (https://popl26.sigplan.org/home/CPP-2026) will be held on 12-13 January 2025 and will be co-located with POPL 2026 in Rennes, France. CPP 2026 is sponsored by ACM SIGPLAN, in cooperation with ACM SIGLOG. CPP 2026 will welcome contributions from all members of the community. The CPP 2026 organizers will strive to enable both in-person and remote participation, in cooperation with the POPL 2026 organizers.

— IMPORTANT DATES

Abstract Submission Deadline:	Sep 05, 2025
Paper Submission Deadline:	Sep 12, 2025
Notification (tentative):	Nov 13, 2025
Camera Ready Deadline (tentative):	Dec 01, 2025
Conference:	Jan 12-13, 2026

Deadlines expire at the end of the day, anywhere on earth. Abstract and submission deadlines are strict and there will be no extensions.

— DISTINGUISHED PAPER AWARDS

Around 10

— For a complete list of topics of interest, and detailed submission instructions, please visit the conference website.

— ORGANIZERS

— Kathrin Stark, Heriot-Watt University (conference co-chair)

— Yannick Zakowski, ENS Lyon (conference co-chair)

— Nikhil Swamy, Microsoft Research (PC co-chair)

— Nicolas Tabareau, Inria (PC co-chair)

For any questions please contact the two PC chairs:

— Nikhil Swamy nswamy@microsoft.com

— Nicolas Tabareau nicolas.tabareau@inria.fr

7. FOSSACS 2026: 29TH INTERNATIONAL CONFERENCE ON FOUNDATIONS OF SOFTWARE SCIENCE AND COMPUTATION STRUCTURES (FOSSACS 2026)

Part of the International Joint Conferences On Theory and Practice of Software (ETAPS 2026)
https://etaps.org/2026/conferences/fossacs/
April 11-16, 2026
Turin, Italy
CALL FOR PAPERS

— GENERAL

FoSSaCS seeks original papers on foundational research with a clear significance for software science. The conference invites submissions on theories and methods to support the analysis, integration, synthesis, transformation, and verification of programs and software systems.

— PROCEEDINGS

The FoSSaCS 2026 proceedings will be published in Springer LNCS as gold open access.

— SUBMISSIONS

FoSSaCS solicits just a single paper category: research papers. The page limit is 18 pp excluding references, respecting Springer's LNCS format at the submission time. Additional material (no page limit) can be placed in a clearly marked appendix, at the end of the paper. The papers can be submitted at https://easychair.org/conferences/?conf=fossacs2026 . FoSSaCS 2026 will adopt a double-blind reviewing process, in line with the other ETAPS conferences. PC members will be allowed to submit up to one paper. These submissions will be held to a higher standard - for example, PC submissions will not be part of the final vote.

— ARTIFACT EVALUATION

FoSSaCS 2026 will have a post-paper-acceptance voluntary artifact evaluation. Authors will be encouraged to submit artifacts for evaluation after paper notification. The outcome will not alter the paper's acceptance decision. Guillermo Alberto Perez (University of Antwerp) will act as the FoSSaCS artifact evaluation chair. Detailed information will shortly appear at https://etaps.org/2026/conferences/ae-esop-fase-fossacs

— IMPORTANT DATES

Submission deadline:	Oct 16, 2025
Rebuttal:	Dec 08, 2025
Paper notification:	Dec 22, 2025
Voluntary artifact submission deadline:	Jan 08, 2026
Paper final version:	Jan 22, 2026
Artifact notification:	Feb 12, 2026
Main Conference:	Apr 13–16, 2026

— PC CHAIRS

— Nathalie Bertrand (Inria Rennes, France)

— Stefan Milius (Friedrich-Alexander Universität Erlangen-Nürnberg, Germany)

8. ICLP 2025: 41ST INTERNATIONAL CONFERENCE ON LOGIC PROGRAMMING

University of Calabria, Rende, Italy
September 12-19, 2025
CALL FOR PARTICIPATION

— We are pleased to announce the availability of student grants to support participation in ICLP 2025, generously provided by the Artificial Intelligence Journal (AIJ), the Italian Association for Artificial Intelligence (AIxIA), and the Gruppo Ricercatori e Utenti Logic Programming (GULP). We encourage all eligible students to apply!

— IMPORTANT DATES

Submission deadline:	Jul 06, 2025
Notification of acceptance:	Jul 13, 2025

— Who can apply?

Grants are available to Master's and PhD students and are conditional upon:

— A certificate of student status

— A statement confirming the lack of sufficient funds to attend ICLP 2025

— What is covered?

The grants will partially cover registration fees and/or shared accommodation in university residences or nearby lodging facilities. The applicant can indicate the type of support he needs. AIJ offers multiple grants, which will be provided as direct discounts on conference services. AixIA offers two grants, up to 400 EUR each, which will be provided via reimbursement handled by the association after the conference. GULP offers two grants, up to 260 EUR each, which will be provided via reimbursement handled by the association after the conference. For US-based students, an NSF-sponsored grant may become available later. Applicants should indicate their preference for a specific grant type in the application form.

— Evaluation Criteria

Applications will be evaluated based on:

— Whether the student is an author of a paper accepted at any ICLP-related event

— The submission timestamp

— Whether the applicant is a member of AIxIA or GULP Associations

— The personal preference for a specific grant

Under no circumstances can a grant be awarded to an applicant who is not enrolled in a recognized study program (Master's level – ISCED Level 7 or PhD level – ISCED Level 8 or equivalent), or who is unable to provide valid proof of student status. The committee will maximise student participation given the available funds. If the NSF-sponsored grant becomes available, it will be allocated to eligible students based in the United States.

To apply, please complete the application form by July 6th by filling in the following form: https://forms.gle/8w39MENkZ9nYbjo38

9. WOLLIC 2025: 31ST WORKSHOP ON LOGIC, LANGUAGE, INFORMATION AND COMPUTATION

14-17 July 2025
Porto, Portugal
https://wollic2025.github.io/
CALL FOR PARTICIPATION

— REGISTRATION

Registration is open in: https://forms.gle/GKzvG4vTCEcxhgpS6

— Late registration:

— Regular: 360 euros

— Student: 310 euros

— INVITED SPEAKERS

— Tobias Kappé (Leiden University): On propositional program equivalence.

— Daniela Petrişan (IRIF, Université de Paris): Functorial Mealy machines.

— ACCEPTED PAPERS

The list of accepted papers can be found at https://wollic2025.github.io/accepted/

— WoLLIC is an annual international forum on interdisciplinary research involving formal logic, computing and programming theory, and natural language and reasoning. Each meeting includes invited talks and tutorials as well as contributed papers. The thirty-first WoLLIC will be held at the University of Porto, Portugal, 14-17 July 2025. WoLLIC is a series of workshops which started in 1994 with the aim of fostering interdisciplinary research in pure and applied logic. The idea is to have a forum which is large enough in the number of possible interactions between logic and the sciences related to information and computation, and yet is small enough to allow for concrete and useful interaction among participants.

— SCIENTIFIC SPONSORSHIP

— Interest Group in Pure and Applied Logics (IGPL)

— The Association for Logic, Language and Information (FoLLI)

— Association for Symbolic Logic (ASL)

— European Association for Theoretical Computer Science (EATCS)

— European Association for Computer Science Logic (EACSL)

— Sociedade Brasileira de Lógica (SBL)

— Sociedade Portuguesa de Lógica (SPL)

10. CONFEST 2025: CONCUR, FMICS, QEST_FORMATS

https://conferences.au.dk/confest2025
CONCUR, FMICS, QEST+FORMATS, and six co-located workshops
August 25-30, 2025
Aarhus, Denmark
CALL FOR PARTICIPATION

— OVERVIEW

— 8 invited talks

— 75 conference paper presentations

— 6 co-located workshops

The early registration deadline is July 26, 2025.

We are excited to invite you to register for CONFEST 2025, which will host three major international conferences:

— CONCUR 2025: 36th International Conference on Concurrency Theory

— FMICS 2025: 30th International Conference on Formal Methods for Industrial Critical Systems

— QEST+FORMATS 2025: Joint International Conference on Quantitative Evaluation of Systems and Formal Modeling and Analysis of Timed Systems

These events will take place in Aarhus, Denmark, from August 25 to August 30, offering a fantastic opportunity to follow the latest advancements, and network with researchers and practitioners in these fields. For more information about the conferences and the venue, please visit: https://conferences.au.dk/confest2025

— Registration

The Early Registration deadline is

Early registration: Jul 26, 2025

via: https://conferences.au.dk/confest2025/registration

Some hotel booking codes are available here: https://conferences.au.dk/confest2025/hotel-accommodations . Other accommodation options are available here: https://www.visitaarhus.com/aarhus/where-sleep/hotels

— Invited Speakers

— Alessandro Abate, U of Oxford, UK. Title: Neural synthesis for verification and control of stochastic systems - certificates and abstractions

— Christel Baier, TU Dresden, Germany. Title: Linear Temporal Logic with Standpoint Modalities

— Lu Feng, University of Virginia, USA. Title: Runtime Safety for Learning-Enabled Cyber-Physical Systems: From Predictive Monitoring to Adaptive Shielding

— Arnd Hartmanns, U of Twente, NL. Title: Sound and Modest Approaches to Quantitative Model Checking from Sea to Space

— Chris Heunen, U of Edinburgh, UK. Title: Towards categorical quantum concurrency theory

— Christoph Matheja, U of Oldenburg, Germany and DTU Denmark. Title: Automating Proof Rules for Probabilistic Programs

— Ina Schieferdecker, Independent Researcher, Germany. Title: Empowering Testing with AI - Navigating the growing field of research on AI for software testing

— Jiri Srba, Aalborg University, Denmark. Title: On-the-Fly Verification: Advancements in Dependency Graphs

— Workshops

— BMQL 2025 - 1st IW on Behavioural Metrics and Quantitative Logics

— Express/SOS 2025 - combined IW on Expressiveness in Concurrency and Structural Operational Semantics

— FMQC 2025 - IW on Formal Methods in Quantum Computing

— PFQA 2025 - Colloquium on Principles of Formal Quantitative Analysis

— Radical 2025 - 4th IW on Recent Advances in Concurrency and Logic

— SynCoP 2025 - 10th IW on Synthesis of Complex Parameters

We hope to meet you in Aarhus this summer!

11. PROFESSOR POSITION: ROSKILDE UNIVERSITY, DENMARK

JOB ANNOUNCEMENT

— Roskilde University, Denmark, invites applications for a Professor of Computer Science or Associate Professor on the promotion programme of Computer Science.

— We are seeking applicants with a background in Computer Science and a strong research profile in Data Science, preferably within the domain of Machine Learning. We are particularly interested in applicants with research in one or more of the following subject areas: (1) explainable AI, (2) the combination machine learning and symbolic reasoning, (3) language models or Natural Language Processing, and (4) cybersecurity.

— Application deadline: September 7th, 2025.

— For more information, see: https://candidate.hr-manager.net/ApplicationInit.aspx?cid=1310&ProjectId=1477

Links: SIGLOG website, LICS website, SIGLOG Monthly