Concurrent ∀∃-Hyperproperties

Ernst-Rüdiger Olderog

joint work with Bernd Finkbeiner

July 2024

Carl von Ossietzky
Universität
Oldenburg

Hyperproperties

Aim: analyze executions of systems

- trace properties = sets of execution traces

  ... express properties of individual executions,

  e.g. safety:

  $$\forall \pi.\, \Box(out_\pi \neq bad)$$

- hyperproperties = sets of sets of traces [CS10]

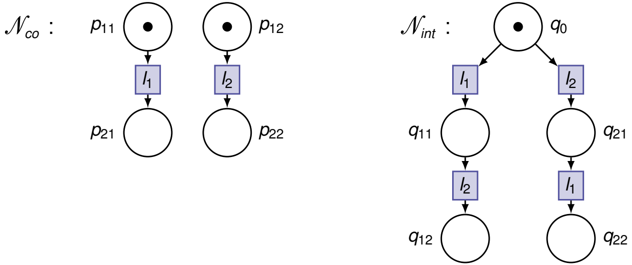
  ... express properties of sets of traces
  by relating different executions,

  e.g. observational determinism:

  $$\forall \pi. \forall \pi'.\, \Box(in_\pi \leftrightarrow in_{\pi'}) \rightarrow \Box(out_\pi \leftrightarrow out_{\pi'})$$

Hyperproperties refer to traces,
which represent concurrency by an interleaving semantics.

We model systems by Petri nets, which represent concurrency using partial orders.



This brings us to concurrent hyperproperties.

Concurrent Traces = Pomsets

Let Σ be a set of labels. A Σ-labeled partially ordered set is a triple $(X, <, \ell)$ where $<$ is a partial order on X and $\ell : X \to \Sigma$ is a labeling function.

A partially ordered multiset (pomset) over Σ is an isomorphy class of Σ-labeled partial ordered sets, denoted as $[(X, <, \ell)]$ [Pra85].

A totally ordered multiset (tomset) is a pomset where $<$ is a total order.

Terminology:

- traces = tomsets over Σ
- trace property = set of traces
- hyperproperty = set of sets of traces
- concurrent traces = pomsets over Σ
- concurrent trace property = set of concurrent traces
- concurrent hyperproperty = set of sets of concurrent traces

$\mathbb{T}(\Sigma)$ = set of all concurrent traces over Σ.

Information Flow Property

Every pair of concurrent traces agrees on the occurrence of the *low-security* events, independent on any other event.

Let Σ_{low} = set of *low-security* events.

The requirement is formalized as the concurrent hyperproperty

$$H_1 = \{\, T \subseteq \mathbb{T}(\Sigma) \mid \forall\, [(X,<,\ell)],\, [(X',<',\ell')] \in T.\ \exists \text{ bijection } f : X_{low} \to X'_{low}.\ \forall x \in X_{low}.\ \ell'(f(x)) = \ell(x) \,\}$$

where

$$X_{low} = \{x \in X \mid \ell(x) \in \Sigma_{low}\}$$
$$X'_{low} = \{x \in X' \mid \ell'(x) \in \Sigma_{low}\}$$

The behavior observable by a *low-security* observer must not change when all *high-security* inputs are removed.

Let $\Sigma = \Sigma_{low} \cup \Sigma_{high}$.

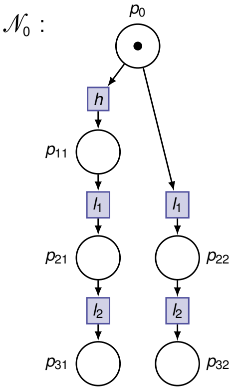
This requirement is formalized with quantifier alternation :

$$H_2 = \{\ T \subseteq \mathbb{T}(\Sigma) \ \mid \ \forall [(X, <, \ell)] \in T.\ \exists [(X', <', \ell')] \in T.\ \exists \text{ bijection } f : X_{low} \to X'_{low}.\ (\ \forall x \in X_{low}.\ \ell'(f(x)) = \ell(x) \wedge \forall x, y \in X_{low}.\ f(x) <' f(y) \Leftrightarrow x < y\ ) \wedge \forall x \in X'.\ \ell'(x) \notin \Sigma_{high}\ \},$$

where X_{low} and X'_{low} are as before.

Example for Noninference

Consider low-security actions l_1 and l_2, and high-security action h in



Low-security behavior must not change when all high-security actions are removed.

Testing of Petri Nets

Idea: testing of processes due to De Nicola and Hennessy [DH84]:
interaction of a nondet. process with a user (test) ,
may and must testing.

Here, a test is a Petri net, extended by a set of successful places.
Graphically, we mark these places by ✓ .

To perform a test $\mathscr{T}$ on a given Petri net $\mathscr{N}$,
we consider the parallel composition $\mathscr{N} \,\|\, \mathscr{T}$.

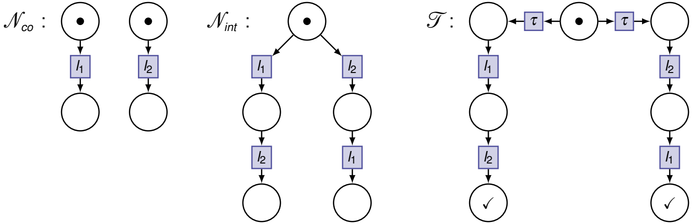
A run $\rho = (\mathscr{N}_R, f)$ of $\mathscr{N} \,\|\, \mathscr{T}$ is deadlock free if it is infinite;
it terminates successfully if it is finite and
all places of $\mathscr{T}$ inside the parallel composition without causal successor are marked with ✓ .

May and Must

A net $\mathscr{N}$ may pass a test $\mathscr{T}$ if there exists a maximal run of $\mathscr{N} \| \mathscr{T}$ which is deadlock free or terminates successfully.

A net $\mathscr{N}$ must pass a test $\mathscr{T}$ if all maximal runs of $\mathscr{N} \| \mathscr{T}$ are deadlock free or terminate successfully.

Example. Tests can distinguish runs of concurrent and interleaved systems:



The run of $\mathscr{N}_{co}$ must pass $\mathscr{T}$ and the runs of $\mathscr{N}_{int}$ may pass $\mathscr{T}$.

Related: causal testing of event structures by Goltz and Wehrheim [GW96].

Checking Hyperproperties

To check a hyperproperty relating two concurrent traces of a system $\mathcal{N}_0$, we investigate maximal runs $\rho = (\mathcal{N}, f)$ and $\rho' = (\mathcal{N}', f')$ of $\mathcal{N}_0$, where $\mathcal{N}$ and $\mathcal{N}'$ are causal nets of $\mathcal{N}_0$,
but in $\mathcal{N}'$ every action u of $\mathcal{N}_0$ is relabled into a primed copy u' .

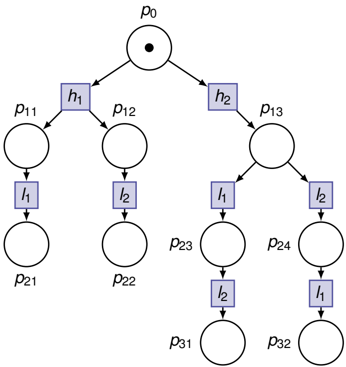
To represent the hyperproperty (with two quantifiers), we test

$$\mathcal{Q}\,\rho.\,\mathcal{Q}'\,\rho'.\ \ \mathcal{N} \parallel \mathcal{N}'\ \mathit{m}\ \ \text{pass}\ \mathcal{T},$$

where $\mathcal{Q}, \mathcal{Q}' \in \{\exists, \forall\}$ and $\mathit{m} \in \{\text{may, must}\}$.

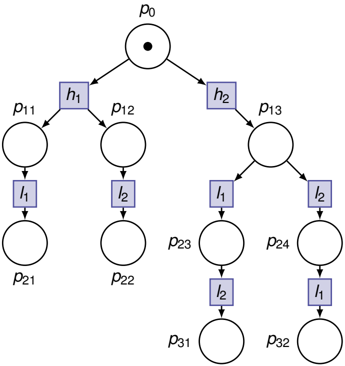
For H_1 consider net $\mathcal{N}_C$

$\mathcal{N}_C$:

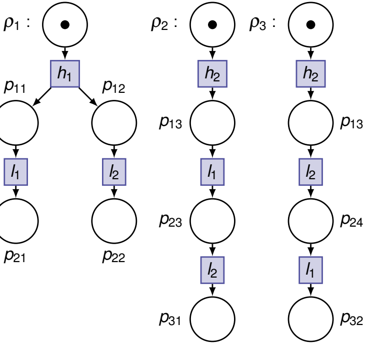


Net $\mathcal{N}_C$ has three maximal runs

$\mathcal{N}_C$:



Three maximal runs:



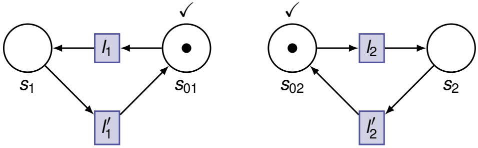
Corresponding traces π_1, π_2, π_3 ignore the places.

Testing Concurrent Hyperproperty H_1

Now we check the concurrent hyperproperty H_1:

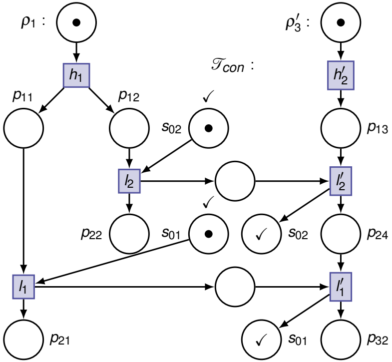
> every pair of concurrent traces π and π'
> agrees on occurrence of low-security events l_1 and l_2.

To this end, we use the concurrent test $\mathscr{T}_{con}$:



Outcome of Concurrent Test $\mathscr{T}_{con}$

The outcome of testing ρ_1 and ρ_3 of $\mathscr{N}_{\mathrm{C}}$:



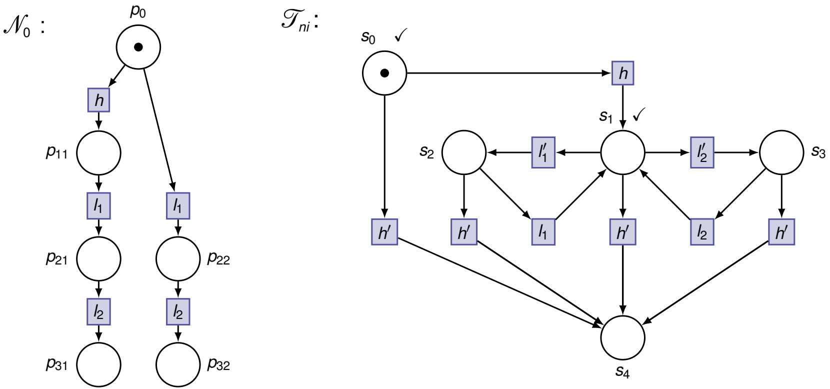
We conclude that $\rho_1 \,\|\, \rho_3$ must pass $\mathscr{T}_{con}$. In general, we have

$$\forall \rho, \rho' \,.\, \mathscr{N} \,\|\, \mathscr{N}' \text{ must pass } \mathscr{T}_{con}.$$

This shows that the system $\mathscr{N}_{\mathrm{C}}$ satisfies H_1 .

Testing Noninference H_2

Consider low-security actions l_1 and l_2, and high-security action h.



The low-security behavior must not change when all high-security actions are removed. This is tested as follows:

$$\forall \rho.\ \exists \rho'.\ \mathcal{N} \parallel \mathcal{N}' \ \text{ must pass } \mathcal{T}_{ni}\,.$$

Decidability Results

... on model checking finite Petri nets against concurrent hyperproperties:

Property class		Model checking problem
$\forall$	must	decidable [FO23]
$\exists$	may	**decidable** (Theorem 1)
$\forall$	may	undecidable [FO23]
$\exists$	must	**undecidable** (Theorem 2)
$\forall\exists / \exists\forall$	must/may	**undecidable** (Corollary 1)

Decidability

Theorem 1. Existential may testing is decidable .

Proof. This testing of a finite, safe net $\mathcal{N}_0$ is of the form

$$(*) \qquad \exists \rho_1, \cdots, \exists \rho_k.\ \mathcal{N}_1 \parallel \cdots \parallel \mathcal{N}_k \ \text{ may pass } \mathcal{T}.$$

We can equivalently refer to copies $\mathcal{N}_{0,1}, \ldots, \mathcal{N}_{0,k}$ of $\mathcal{N}_0$, and check

$$\mathcal{N} = \mathcal{N}_{0,1} \parallel \cdots \parallel \mathcal{N}_{0,k} \parallel \mathcal{T},$$

for the following properties:

(1) the unfolding of $\mathcal{N}$ is infinite

or (2) $\exists M \in \mathit{reach}(\mathcal{N}).\ \forall p \in \mathit{Pl}_{\mathcal{T}} \cap M.$
$p \in \checkmark \wedge \neg \exists t \in \longrightarrow .\ \mathit{pre}(t) \subseteq M.$

Both properties are decidable.

Undecidability

Theorem 2. Existential must testing is undecidable .

Proof. We reduce the infinite Post Correspondence Problem (ω-PCP) to existential must testing.

Illustration for ω-PCP over alphabet $\{a, b\}$. As input I, consider the lists (u_1, u_2, u_3) and (v_1, v_2, v_3), where

$$u_1 = b,\ u_2 = b,\ u_3 = aba \quad \text{and} \quad v_1 = ba,\ v_2 = aba,\ v_3 = b.$$

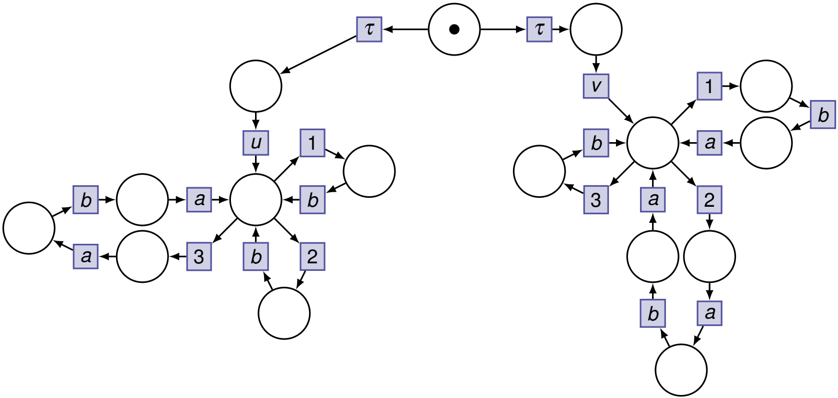
The ω-PCP with this input is solvable by the infinite (ω-regular) correspondence $1 \cdot (3 \cdot 2)^\omega$ because

$$\begin{array}{rcl} u_1\, u_3\, u_2\, u_3\, u_2 \cdots & = & b \,|\, aba \,|\, b \,|\, aba \,|\, b \cdots \\ v_1\, v_3\, v_2\, v_3\, v_2 \cdots & = & ba \,|\, b \,|\, aba \,|\, b \,|\, aba \cdots \end{array}$$

The input I has no solution as a normal, finite PCP.

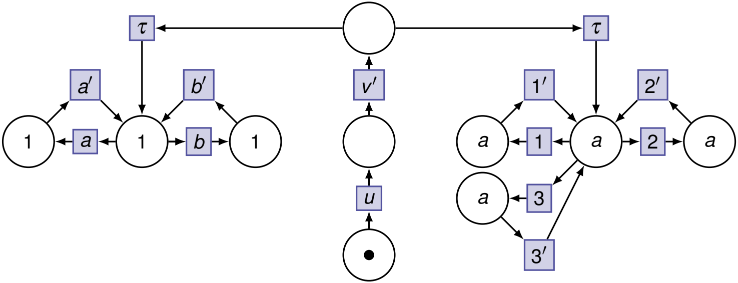
Simulating the Input I

Petri net $\mathcal{N}_I$ simulating the input I of the ω-PCP:



Test $\mathscr{T}_\omega$

... for checking whether two runs of $\mathscr{N}_I$
simulate a correspondence of the ω-PCP:



Quantifier Alternation

Corollary 1.
For a single quantifier alternation we can extend the above results:

1. $\forall\exists$ may testing and $\exists\forall$ may testing are undecidable .
2. $\forall\exists$ must testing and $\exists\forall$ must testing are undecidable .

Theorem 3.
Consider $\forall\exists$ may testing of a finite net $\mathscr{N}_0$ of the form

$$(*) \qquad \forall \rho . \exists \rho' . \mathscr{N} \| \mathscr{N}' \text{ may pass } \mathscr{T},$$

where $\mathscr{N}$ and $\mathscr{N}'$ are the nets belonging to the runs ρ and ρ' of $\mathscr{N}_0$ and where $\alpha(\mathscr{T}) = \alpha(\mathscr{N}) \cup \alpha(\mathscr{N}')$.

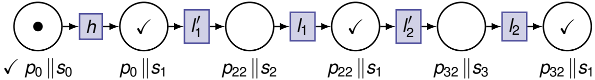
Suppose the parallel composition $\mathscr{N}' \| \mathscr{T}$ yields a deterministic net.

Then it is decidable whether $(*)$ holds.

Example for Theorem 3

Consider $\mathscr{N}_0$ and test $\mathscr{T}_{ni}$ for noninference.
Let $\mathscr{N}'$ be the net of the run to the right of $\mathscr{N}_0$.

Then $\mathscr{N}' \,||\, \mathscr{T}_{ni}$ is deterministic:



Conclusion

Summary:

Model checking is undecidable for concurrent hyperproperties that combine existential and universal quantification.

This result is in contrast to standard (non-concurrent) hyperproperties, for example specified by HyperLTL [CFK$^+$14].

For HyperLTL model checking remains decidable for arbitrary quantifier alternations, but with nonelementary complexity [FRS15].

Future work:

Generalize Theorem 3 to some nondeterministic testers.

References I

Michael R. Clarkson, Bernd Finkbeiner, Masoud Koleini, Kristopher K. Micinski, Markus N. Rabe, and César Sánchez.
Temporal logics for hyperproperties.
In Martín Abadi and Steve Kremer, editors, *Principles of Security and Trust – Third Intern. Conf., POST 2014, Held as Part of ETAPS 2014, Proc.*, volume 8414 of *LNCS*, pages 265–284. Springer, 2014.

Michael R. Clarkson and Fred B. Schneider.
Hyperproperties.
J. Comput. Secur., 18(6):1157–1210, 2010.

R. De Nicola and M. Hennessy.
Testing equivalences for processes.
TCS, 34:83–134, 1984.

Bernd Finkbeiner and Ernst-Rüdiger Olderog.
Concurrent hyperproperties.
In Jonathan P. Bowen, Qin Li, and Qiwen Xu, editors, *Theories of Programming and Formal Methods - Essays Dedicated to Jifeng He on the Occasion of His 80th Birthday*, volume 14080 of *LNCS*, pages 211–231. Springer, 2023.
Open access.

Bernd Finkbeiner, Markus N. Rabe, and César Sánchez.
Algorithms for model checking HyperLTL and HyperCTL*.
In Daniel Kroening and Corina S. Pasareanu, editors, *Computer Aided Verification – 27th Intern. Conf., CAV 2015, Proc., Part I*, volume 9206 of *LNCS*, pages 30–48. Springer, 2015.

Ursula Goltz and Heike Wehrheim.
Causal testing.
In Wojciech Penczek and Andrzej Szalas, editors, *Mathematical Foundations of Computer Science 1996, 21st Intern. Symp., Proc.*, volume 1113 of *LNCS*, pages 394–406. Springer, 1996.

References II

John McLean.
A general theory of composition for trace sets closed under selective interleaving functions.
In *1994 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 79–93. IEEE Computer Society, 1994.

Vaughan R. Pratt.
The pomset model of parallel processes: Unifying the temporal and the spatial.
In Stephen D. Brookes, A. W. Roscoe, and Glynn Winskel, editors, *Seminar on Concurrency, Carnegie-Mellon University*, volume 197 of *LNCS*, pages 180–196. Springer, 1985.