Hyperpropreties

Hadar Frenkel, January 13, 2025

Introduction to Temporal Hyperpropreties

Hadar Frenkel, January 13, 2025

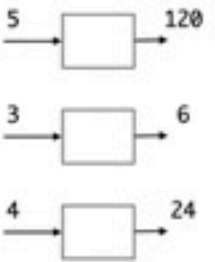
Motivation

Is Our Program Correct?

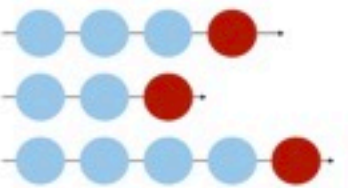
Is Our Program Correct?

Trace Properties
allowed behaviours

Functional correctness



Termination



Correctness is not Enough

Systems can be correct but still not:

Secure Fair Robust

...

Correctness is not Enough

Systems can be correct but still not:

Secure Fair Robust

...

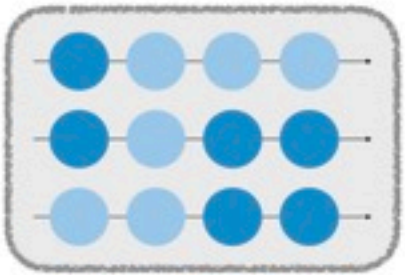
What is the set of properties the system should satisfy?

Hyperproperties

System properties

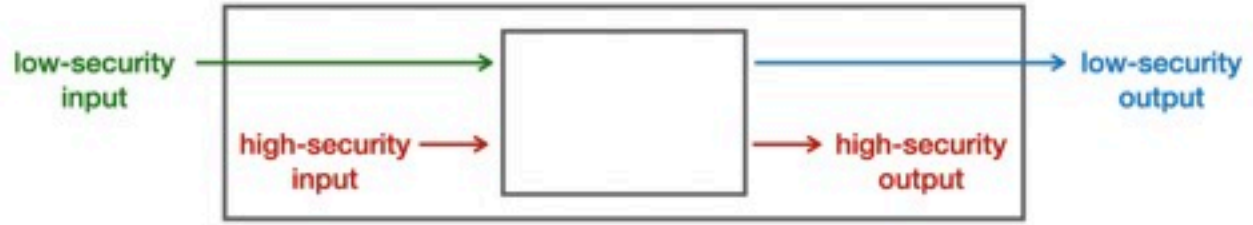
Relate multiple executions

Information-flow security



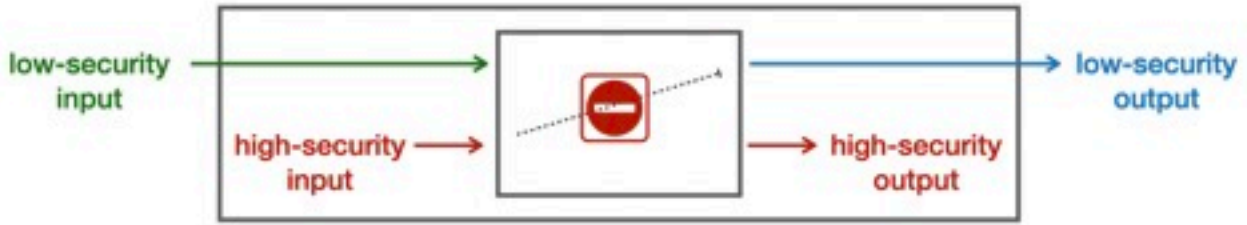
Hyperproperties. Clarkson and Schneider (CSF 2008)

Example: Noninterference



Security Policies and Security Models. Goguen and Meseguer (IEEE 1982)

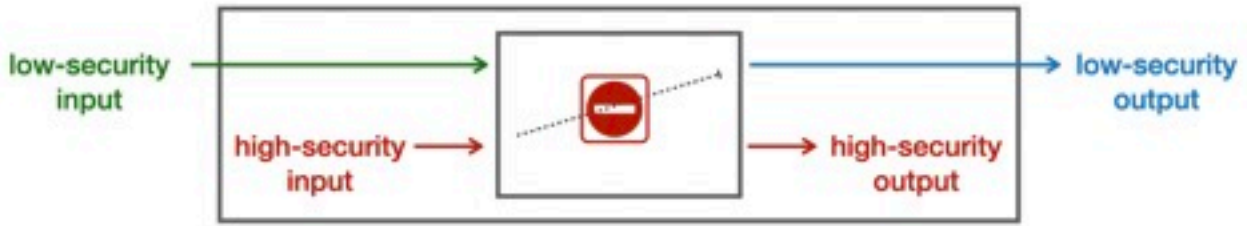
Example: Noninterference



Low-security output can only depend on low-security input

Security Policies and Security Models. Goguen and Meseguer (IEEE 1982)

Example: Noninterference



Low-security output can only depend on low-security input

For every two executions:
if they agree on low-security input then
they agree on low-security output

Security Policies and Security Models. Goguen and Meseguer (IEEE 1982)

Hyperproperties

System properties

Relate multiple executions

Information-flow

Hyperproperties

System properties

Relate multiple executions

Information-flow

Knowledge

Hyperproperties

System properties

Relate multiple executions

Information-flow

Knowledge

Fairness

Hyperproperties

System properties

Relate multiple executions

Information-flow

Knowledge

Fairness

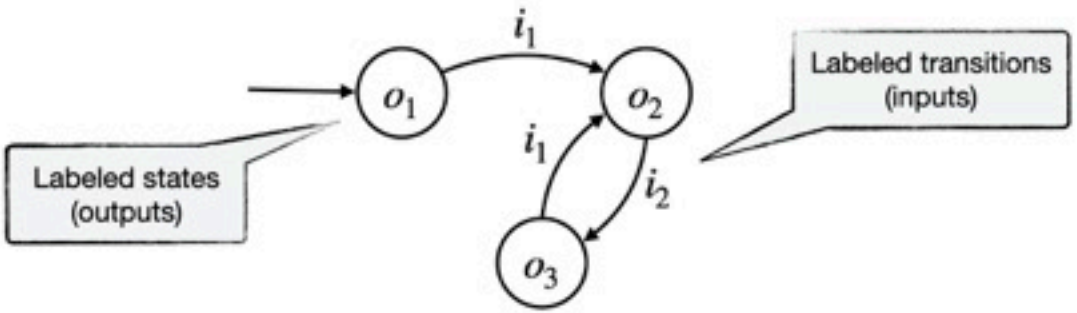
Robustness

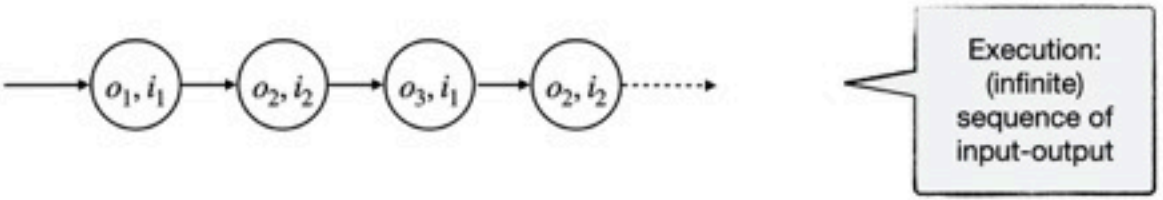
Hyperproperties are important!
Let’s verify them

Outline

- Trace properties & Hyperproperties
- HyperLTL - a logic for temporal hyperproperties
- Some interesting properties in HyperLTL
- HyperLTL model-checking
- HyperLTL satisfiability
- Beyond HyperLTL and security

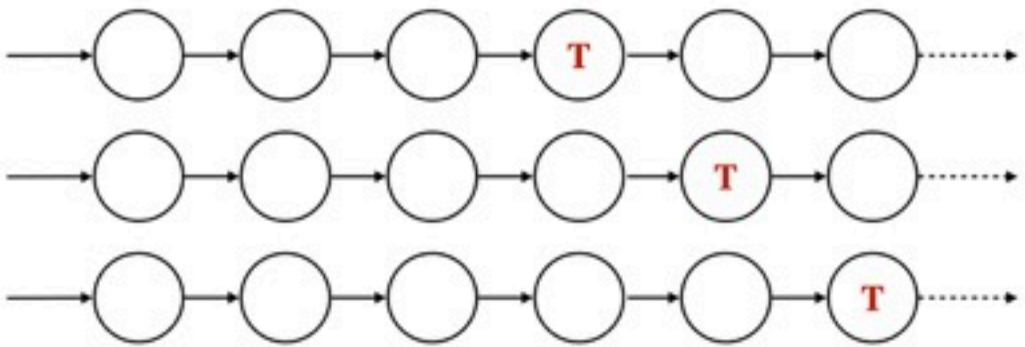
Setting


i_1
o_1
o_2
i_1
i_2
o_3
Labeled states
(outputs)
Labeled transitions
(inputs)


o_1, i_1
o_2, i_2
o_3, i_1
o_2, i_2
Execution:
(infinite)
sequence of
input-output

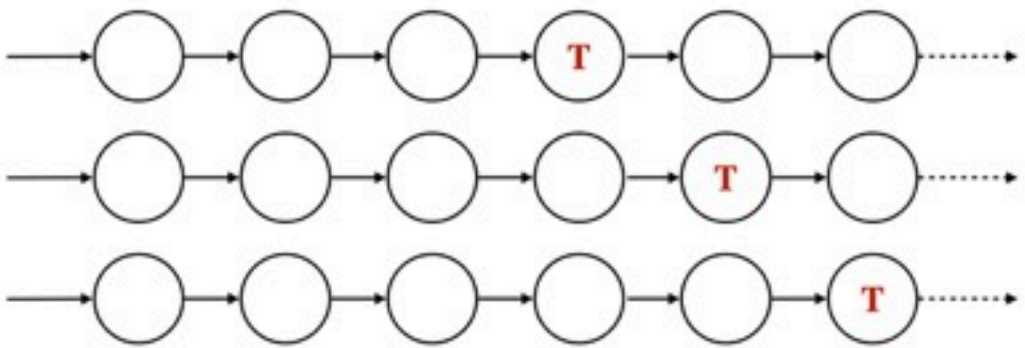
Trace Properties

"All executions reach T"



Trace Properties

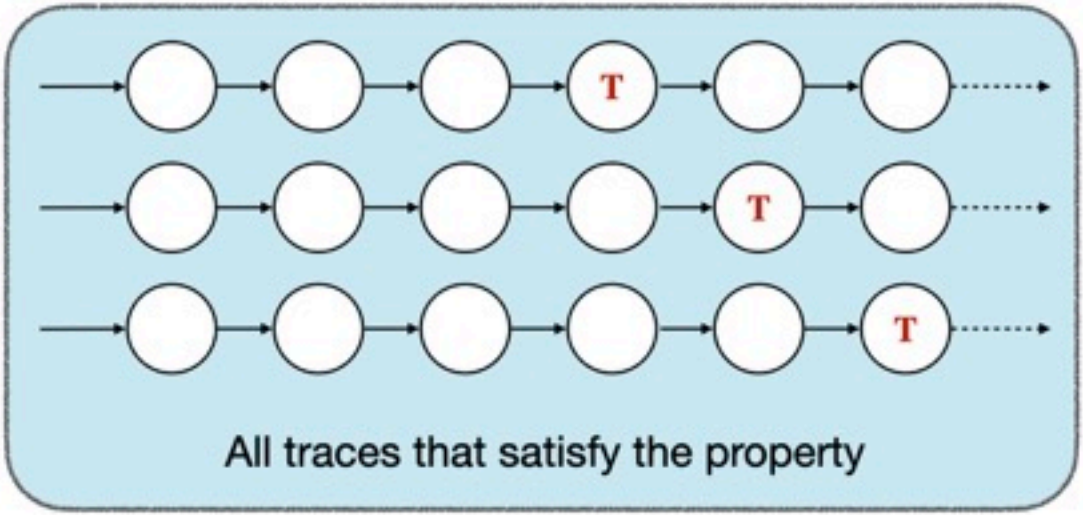
"All executions reach T"



We can look at each trace separately and verify the property

Trace Properties

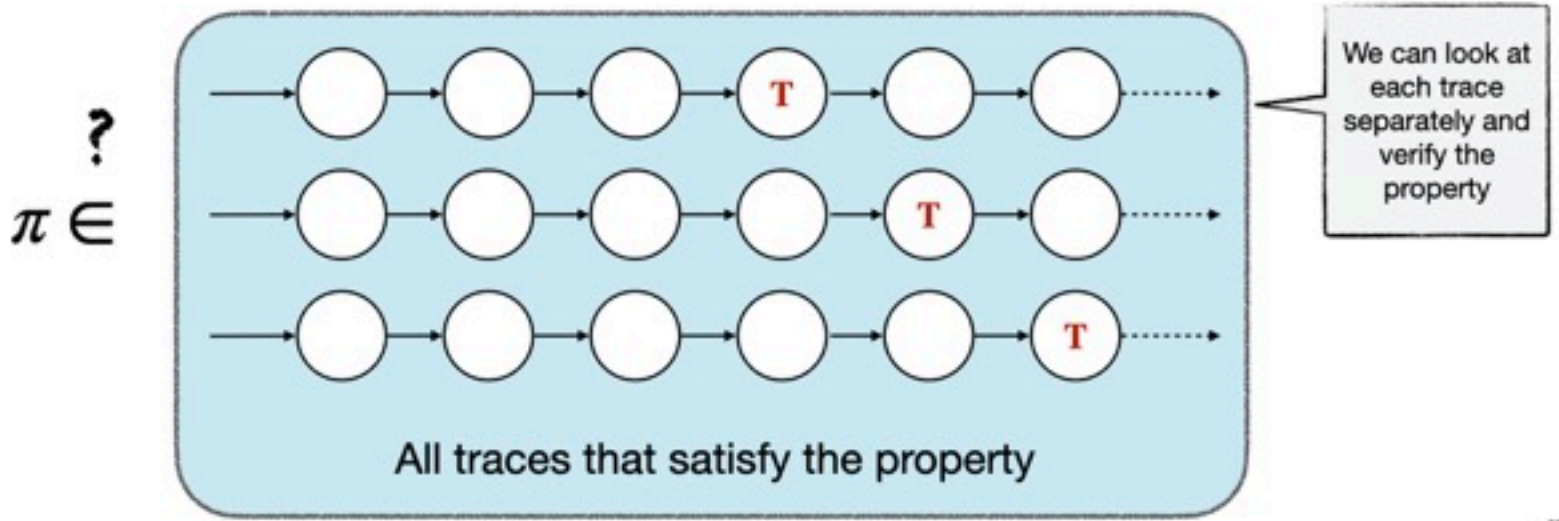
"All executions reach T"



We can look at each trace separately and verify the property

Trace Properties

"All executions reach T"



Hyperproperties

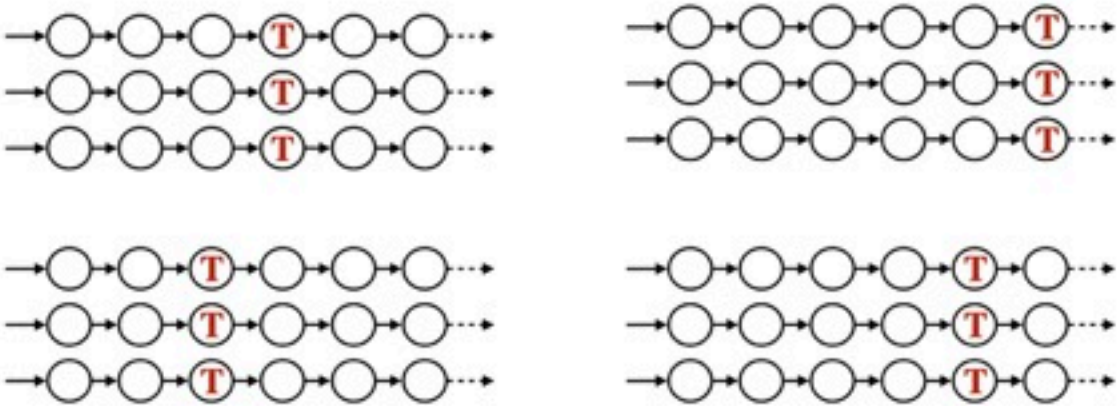
"All executions reach T at the same time"

We have to compare different executions

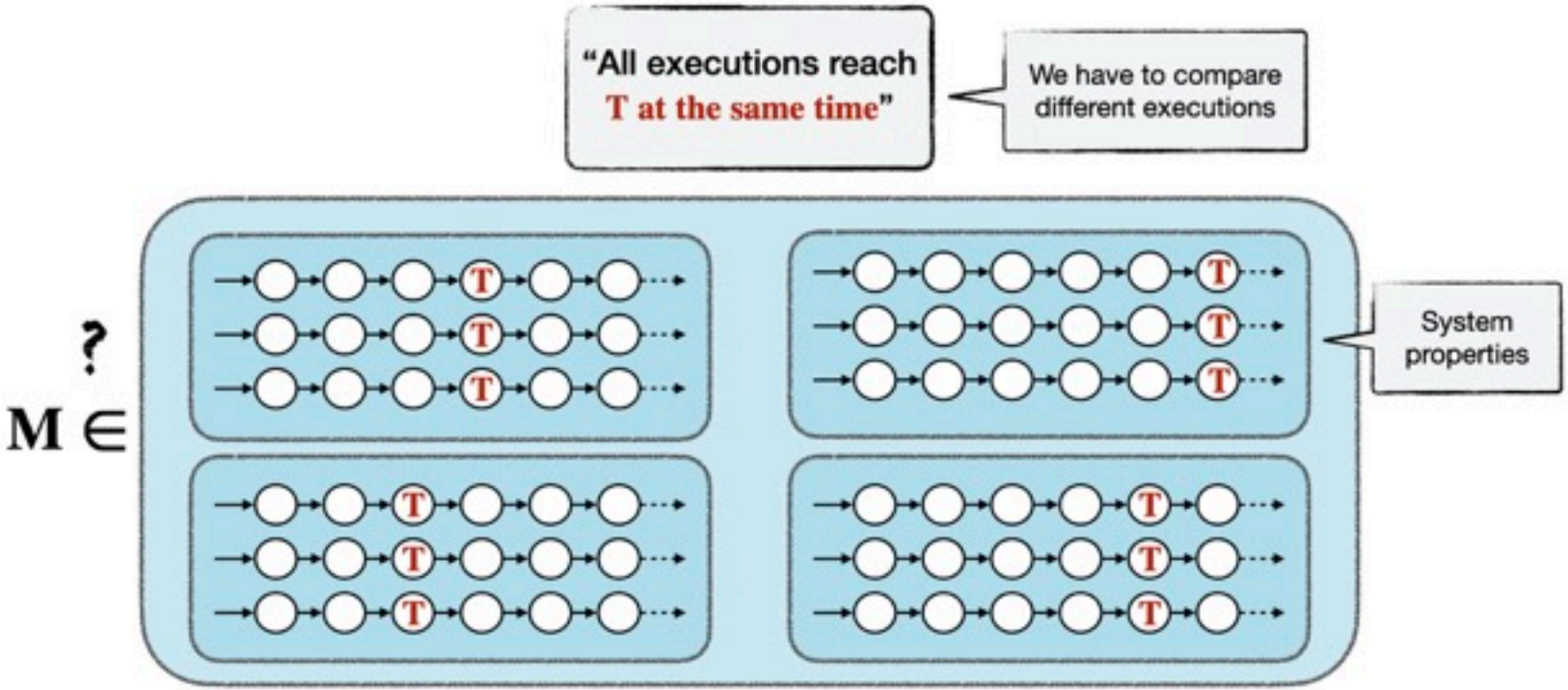
Hyperproperties

"All executions reach T at the same time"

We have to compare different executions

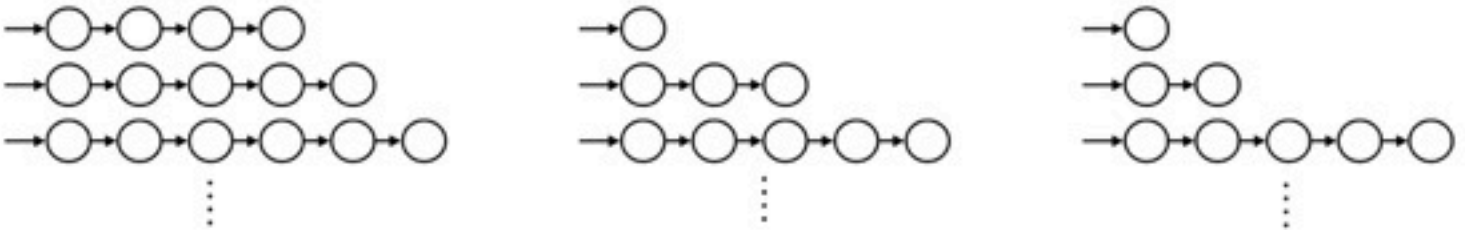


Hyperproperties



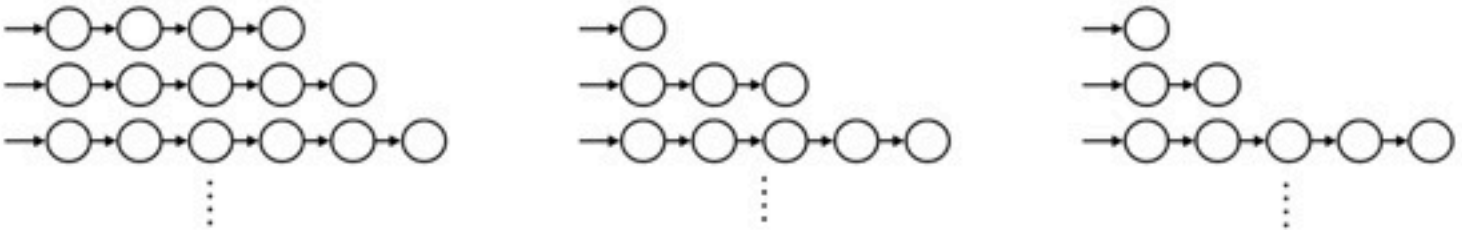
Hyperproperties, Example 2

All infinite languages



Hyperproperties, Example 2

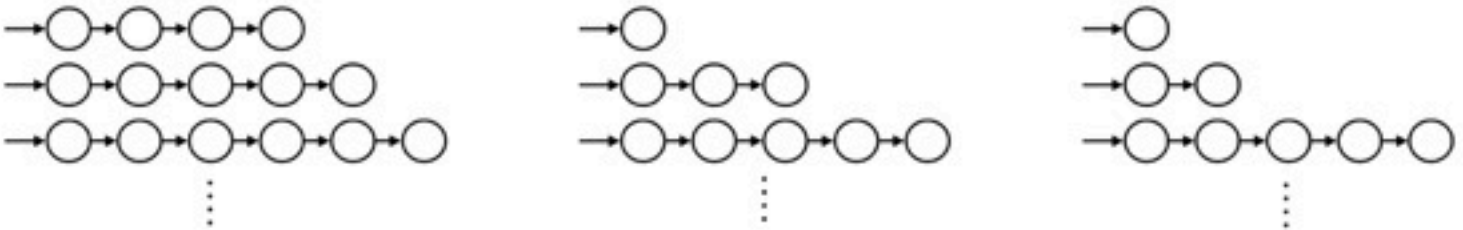
All infinite languages



for every trace
exists a longer trace

Hyperproperties, Example 2

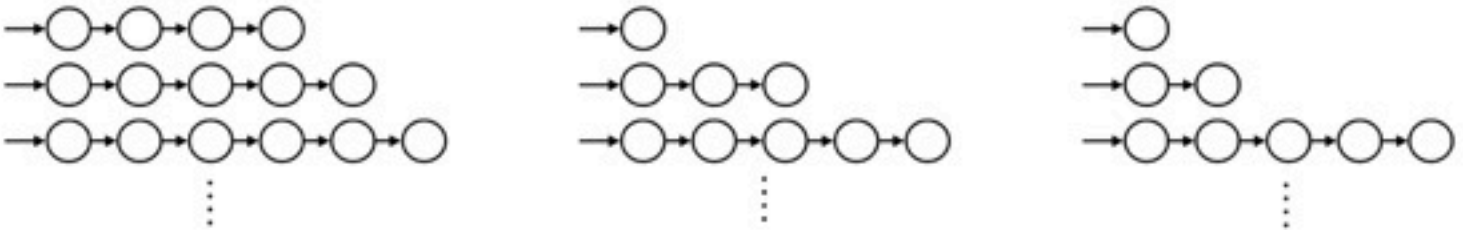
All infinite languages



for every trace
exists a longer trace
&
exists at least one trace

Hyperproperties, Example 2

All infinite languages



for every trace
exists a longer trace
&
exists at least one trace

Quantification over traces!

LTL

$$\psi := a \mid \neg\psi \mid \psi \wedge \psi \mid \bigcirc \psi \mid \psi \cup \psi'$$

ψ holds at the next step

ψ holds until ψ' holds

temporal operators
describe behavior through time:
$\Diamond\psi$ – ψ eventually holds
$\Box\psi$ – ψ holds all the time (globally)

The Temporal Logic of Programs. Pnueli (FOCS 1977).

HyperLTL

$$\psi := a_\pi \mid \neg\psi \mid \psi \wedge \psi \mid \bigcirc \psi \mid \psi \mathbin{U} \psi'$$

$$\varphi := \exists \pi . \varphi \mid \forall \pi . \varphi \mid \psi$$

trace variable

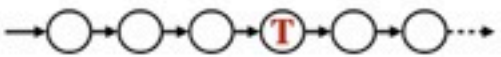
Temporal Logics for Hyperproperties. Clarkson, Finkbeiner, Koleini, Micinski, Rabe, and Sánchez. POST14.

HyperLTL Example

"All executions reach T at the same time"

$$\exists \pi . \neg T_\pi \; U \; (T_\pi \wedge \bigcirc \Box \neg T_\pi)$$

T holds exactly once on π

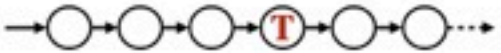


HyperLTL Example

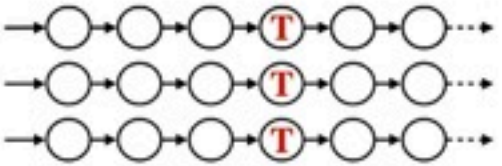
"All executions reach T at the same time"

$\exists \pi . \neg T_\pi \; U \; (T_\pi \wedge \bigcirc \Box \neg T_\pi)$ $\wedge$ $\forall \pi' . \Diamond (T_\pi \wedge T_{\pi'})$

T holds exactly once on π



For all other traces π', T holds at the same time as in π



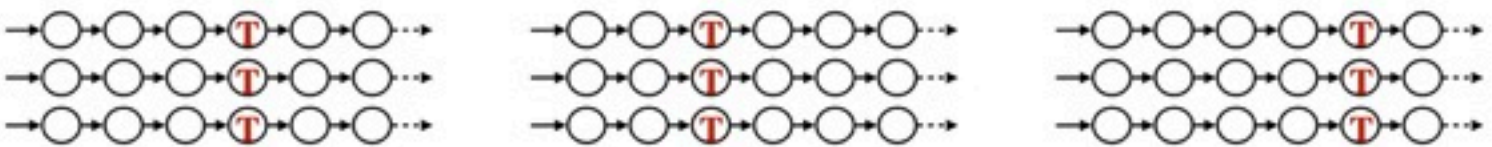
HyperLTL Example

"All executions reach T at the same time"

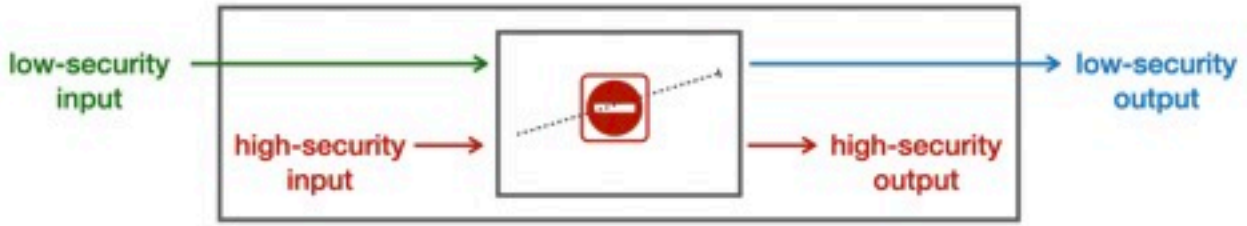
$$\exists \pi . \neg T_\pi \; U \; (T_\pi \wedge \bigcirc \Box \neg T_\pi) \qquad \wedge \qquad \forall \pi' . \Diamond (T_\pi \wedge T_{\pi'})$$

T holds exactly once on π

For all other traces π', T holds at the same time as in π



Example: Noninterference

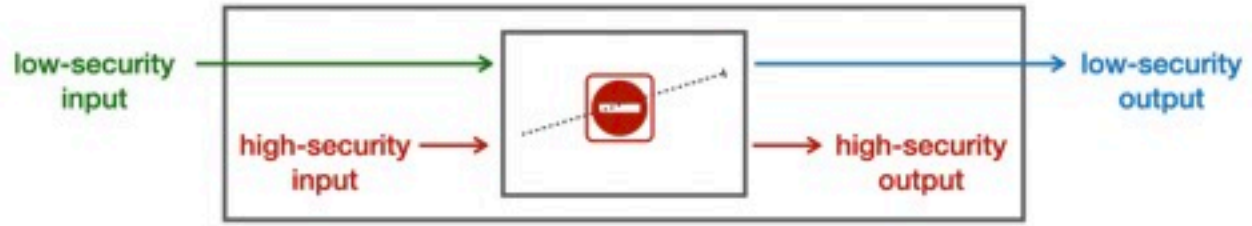


Low-security output can only depend on low-security input

For every two executions:
if they agree on low-security input then
they agree on low-security output

Security Policies and Security Models. Goguen and Meseguer (IEEE 1982)

Example: Noninterference



Low-security output can only depend on low-security input

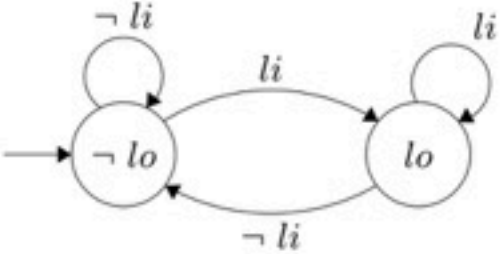
For every two executions:
if they agree on low-security input then
they agree on low-security output

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

Security Policies and Security Models. Goguen and Meseguer (IEEE 1982)

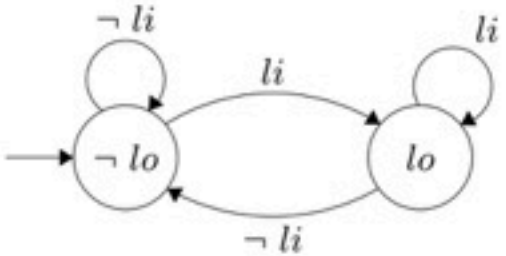
Example: Noninterference

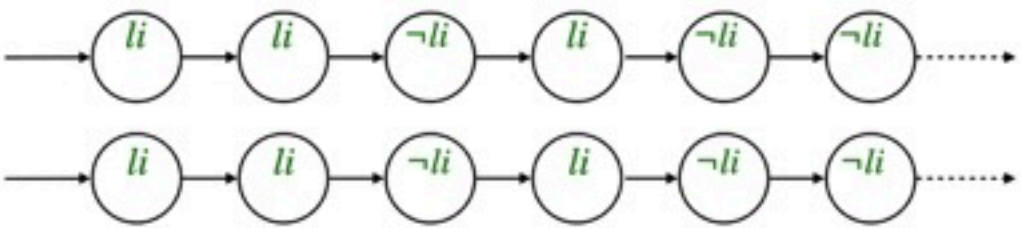
$$\forall \pi \forall \pi' . \square(low - in_{\pi} \leftrightarrow low - in_{\pi'}) \rightarrow \square(low - out_{\pi} \leftrightarrow low - out_{\pi'})$$



Example: Noninterference

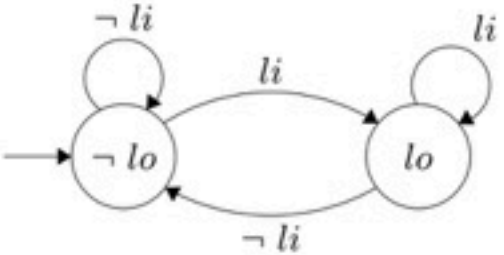
$$\forall \pi \forall \pi' . \square(low - in_{\pi} \leftrightarrow low - in_{\pi'}) \rightarrow \square(low - out_{\pi} \leftrightarrow low - out_{\pi'})$$

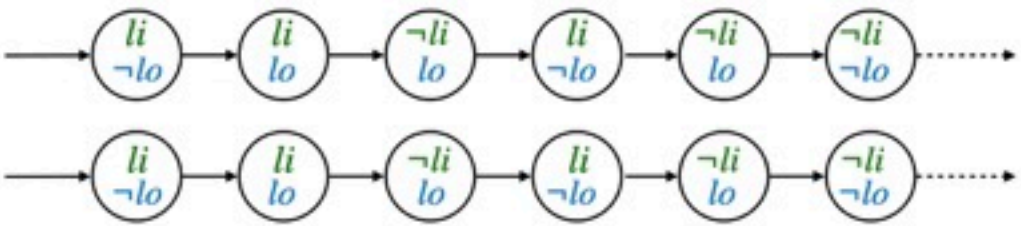




Example: Noninterference

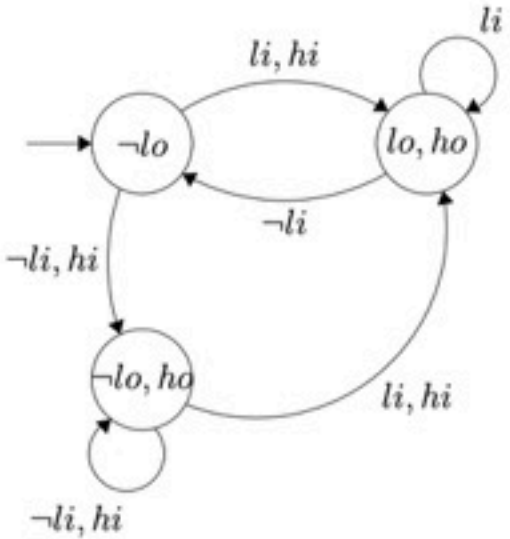
$$\forall \pi \forall \pi' . \square(low - in_{\pi} \leftrightarrow low - in_{\pi'}) \rightarrow \square(low - out_{\pi} \leftrightarrow low - out_{\pi'})$$





Example: Noninterference

$$\forall \pi \forall \pi' . \square(low - in_{\pi} \leftrightarrow low - in_{\pi'}) \rightarrow \square(low - out_{\pi} \leftrightarrow low - out_{\pi'})$$



Systems can be more complex and include high-security variables

But our examples will remain simple

Outline

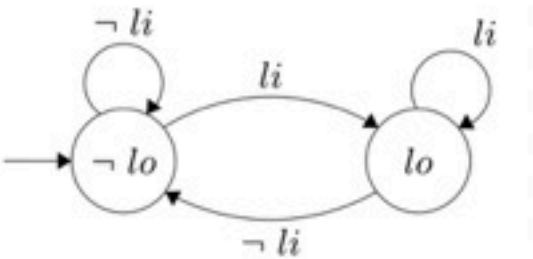
- Trace properties & Hyperproperties
- HyperLTL - a logic for temporal hyperproperties
- Some interesting properties in HyperLTL
- HyperLTL model-checking
- HyperLTL satisfiability
- Beyond HyperLTL and security

Outline

- Trace properties & Hyperproperties
- HyperLTL - a logic for temporal hyperproperties
- Some interesting properties in HyperLTL
- HyperLTL model-checking
  - $\exists^*$-HyperLTL
  - $\forall^*$-HyperLTL
  - Quantifier alternation
- HyperLTL satisfiability
- Beyond HyperLTL and security

Model-Checking ∃*HyperLTL

$$\exists \pi \exists \pi' . \square (low - in_{\pi} \leftrightarrow low - in_{\pi'})$$

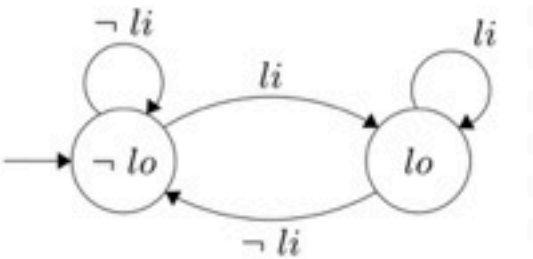


*Algorithms for Model Checking HyperLTL and HyperCTL**. Finkbeiner, Rabe, and Sánchez. CAV 2015.

Model-Checking ∃*HyperLTL

$$\exists \pi \exists \pi' . \square (low - in_{\pi} \leftrightarrow low - in_{\pi'})$$

Look for witness traces

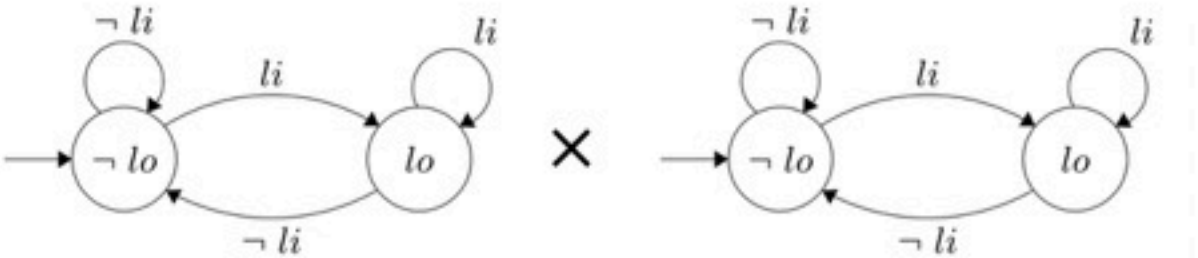


*Algorithms for Model Checking HyperLTL and HyperCTL**. Finkbeiner, Rabe, and Sánchez. CAV 2015.

Model-Checking $\exists^*$HyperLTL

$$\exists \pi \exists \pi' . \square (low - in_{\pi} \leftrightarrow low - in_{\pi'})$$

Look for witness traces in the composed system

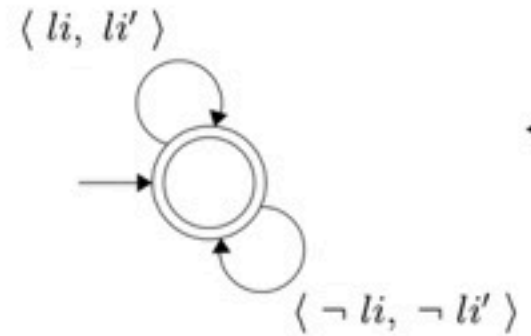


*Algorithms for Model Checking HyperLTL and HyperCTL**. Finkbeiner, Rabe, and Sánchez. CAV 2015.

Model-Checking $\exists^*$HyperLTL

HyperLTL to an Automaton

$$\exists \pi \exists \pi' . \square (low - in_\pi \leftrightarrow low - in_{\pi'})$$



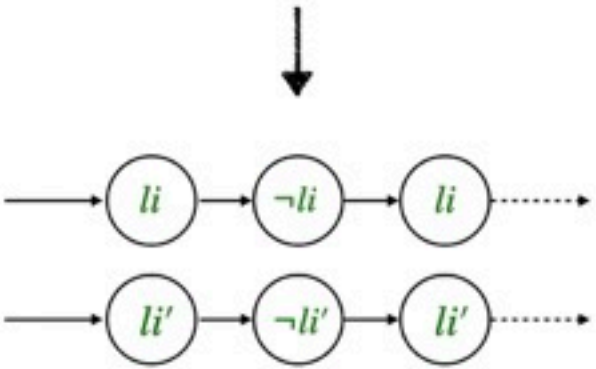
2 quantifiers, alphabet is $\Sigma \times \Sigma$
k quantifiers, alphabet is Σ^k

*Algorithms for Model Checking HyperLTL and HyperCTL**. Finkbeiner, Rabe, and Sánchez. CAV 2015.

Model-Checking $\exists^*$HyperLTL

HyperLTL to an Automaton

$$\langle li, li' \rangle, \langle \neg li, \neg li' \rangle, \langle li, li' \rangle, \cdots$$



2 quantifiers, alphabet is $\Sigma \times \Sigma$
k quantifiers, alphabet is Σ^k

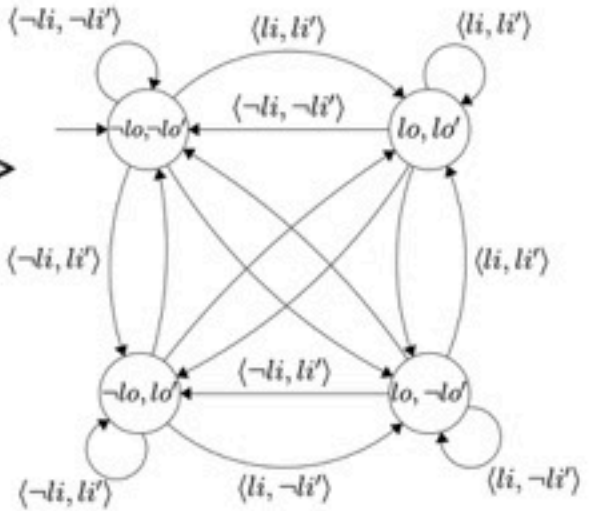
Traces over k-tuples of letters represent k traces

*Algorithms for Model Checking HyperLTL and HyperCTL**. Finkbeiner, Rabe, and Sánchez. CAV 2015.

Model-Checking ∃*HyperLTL

$$\exists \pi \exists \pi' . \square (low - in_{\pi} \leftrightarrow low - in_{\pi'})$$

Self-compose the system (k times)

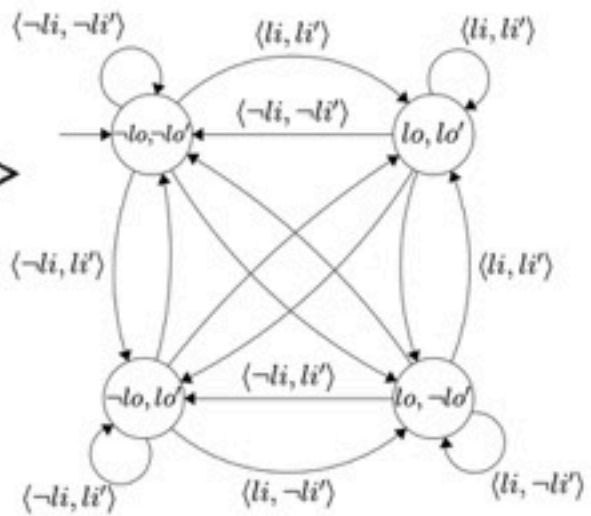


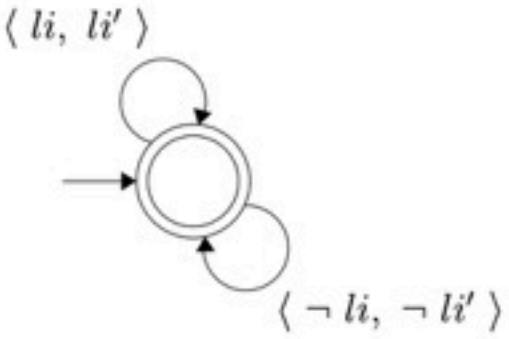
*Algorithms for Model Checking HyperLTL and HyperCTL**. Finkbeiner, Rabe, and Sánchez. CAV 2015.

Model-Checking ∃*HyperLTL

$$\exists \pi \exists \pi' . \square (low - in_{\pi} \leftrightarrow low - in_{\pi'})$$

Self-compose the system (k times)



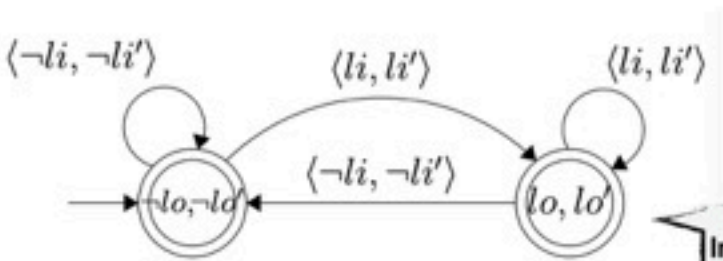


Intersect with the inner formula automaton, to find witness traces

Algorithms for Model Checking HyperLTL and HyperCTL.* Finkbeiner, Rabe, and Sánchez. CAV 2015.

Model-Checking $\exists^*$HyperLTL

$$\exists \pi \exists \pi' . \square (low - in_{\pi} \leftrightarrow low - in_{\pi'})$$



Intersection of the inner formula and the 2-slef composition of the system

Test for emptiness

*Algorithms for Model Checking HyperLTL and HyperCTL**. Finkbeiner, Rabe, and Sánchez. CAV 2015.

Model-Checking $\exists^*$HyperLTL

$$\exists \pi_1 \cdots \exists \pi_k \,.\, \psi$$

1. A_ψ — an automaton for the inner formula ψ, over k-tuples of letters
2. M_k — a k self-composition of the system
3. Intersect $A_\psi \cap M_k$ and check for emptiness
4. A k-trace in $A_\psi \cap M_k$ represents the witness traces $\pi_1, \cdots, \pi_k$

*Algorithms for Model Checking HyperLTL and HyperCTL**. Finkbeiner, Rabe, and Sánchez. CAV 2015.

Outline

- Trace properties & Hyperproperties
- HyperLTL - a logic for temporal hyperproperties
- Some interesting properties in HyperLTL
- HyperLTL model-checking
  - $\exists^*$-HyperLTL
  - $\forall^*$-HyperLTL
  - Quantifier alternation
- HyperLTL satisfiability
- Beyond HyperLTL and security

Model-Checking ∀*HyperLTL

$$\forall \pi \forall \pi' . \square(low - in_{\pi} \leftrightarrow low - in_{\pi'}) \rightarrow \square(low - out_{\pi} \leftrightarrow low - out_{\pi'})$$

Model-Checking $\forall^*$HyperLTL

$$\forall\pi\forall\pi' . \square (low - in_\pi \leftrightarrow low - in_{\pi'})$$

2 quantifiers, alphabet is $\Sigma \times \Sigma$

This time we are looking for **violation**

*Algorithms for Model Checking HyperLTL and HyperCTL**. Finkbeiner, Rabe, and Sánchez. CAV 2015.

Model-Checking ∀*HyperLTL

$$\forall \pi \forall \pi' . \square (low - in_{\pi} \leftrightarrow low - in_{\pi'})$$

2 quantifiers, alphabet is $\Sigma \times \Sigma$

$$\exists \pi \exists \pi' . \Diamond (low - in_{\pi} \leftrightarrow \neg low - in_{\pi'})$$

This time we are looking for **violation**

*Algorithms for Model Checking HyperLTL and HyperCTL**. Finkbeiner, Rabe, and Sánchez. CAV 2015.

Model-Checking ∀*HyperLTL

$$\forall \pi \forall \pi' . \square (low - in_{\pi} \leftrightarrow low - in_{\pi'})$$

2 quantifiers, alphabet is $\Sigma \times \Sigma$

$$\exists \pi \exists \pi' . \Diamond (low - in_{\pi} \leftrightarrow \neg low - in_{\pi'})$$

This time we are looking for **violation**

This looks familiar…

*Algorithms for Model Checking HyperLTL and HyperCTL**. Finkbeiner, Rabe, and Sánchez. CAV 2015.

Model-Checking $\forall^*$HyperLTL

$$\forall \pi_1 \cdots \forall \pi_k \,.\, \psi$$

1. Negate the formula — $\exists \pi_1 \cdots \exists \pi_k \,.\, \neg\psi$
2. $A_{\neg\psi}$ — an automaton for the inner formula $\neg\psi$, over k-tuples of letters
3. M_k — a k self-composition of the system
4. Intersect $A_{\neg\psi} \cap M_k$ and check for emptiness
5. A k-trace in $A_{\neg\psi} \cap M_k$ represents the counterexample traces $\pi_1, \cdots, \pi_k$

*Algorithms for Model Checking HyperLTL and HyperCTL**. Finkbeiner, Rabe, and Sánchez. CAV 2015.

Outline

- Trace properties & Hyperproperties
- HyperLTL - a logic for temporal hyperproperties
- Some interesting properties in HyperLTL
- HyperLTL model-checking
  - $\exists^*$-HyperLTL
  - $\forall^*$-HyperLTL
  - Quantifier alternation
- HyperLTL satisfiability
- Beyond HyperLTL and security

Model-Checking $\forall\exists$HyperLTL

Generalized Noninterference

$$\forall\pi\forall\pi'\exists\pi''.\,\square(high-in_{\pi} \leftrightarrow high-in_{\pi''}) \wedge \square(low-out_{\pi'} \leftrightarrow low-out_{\pi''})$$

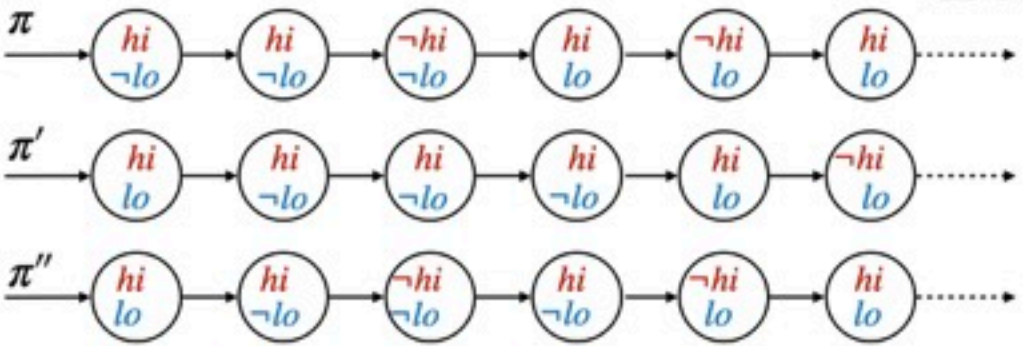
For every two traces
there exists a third trace
that agrees with one on the high-security inputs,
and with the other on the low-security outputs

Model-Checking $\forall\exists$HyperLTL

Generalized Noninterference

$$\forall\pi\forall\pi'\exists\pi''.\,\square(high-in_{\pi} \leftrightarrow high-in_{\pi''}) \wedge \square(low-out_{\pi'} \leftrightarrow low-out_{\pi''})$$

For every two traces
there exists a third trace
that agrees with one on the high-security inputs,
and with the other on the low-security outputs

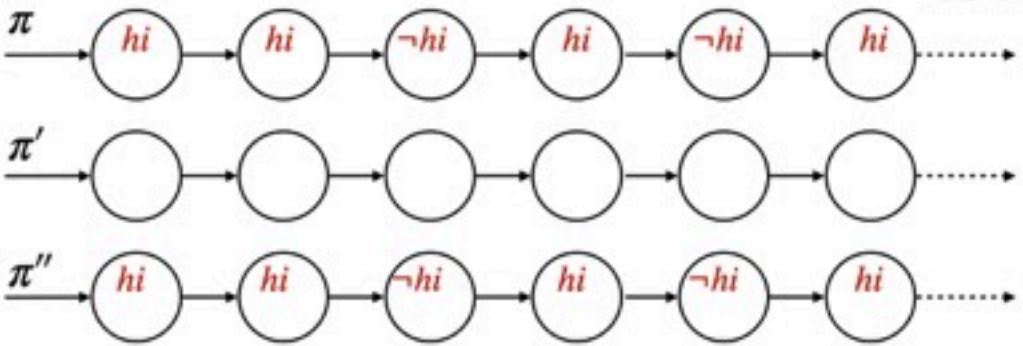


Model-Checking $\forall\exists$HyperLTL

Generalized Noninterference

$$\forall\pi\forall\pi'\exists\pi''.\ \square(high - in_{\pi} \leftrightarrow high - in_{\pi''}) \wedge \square(low - out_{\pi'} \leftrightarrow low - out_{\pi''})$$

For every two traces
there exists a third trace
that agrees with one on the high-security inputs,
and with the other on the low-security outputs

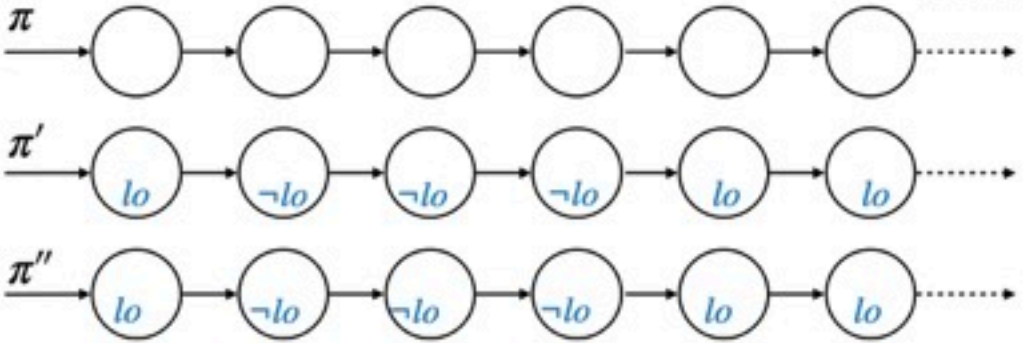


Model-Checking $\forall\exists$HyperLTL

Generalized Noninterference

$$\forall\pi\forall\pi'\exists\pi''.\,\square(high-in_{\pi} \leftrightarrow high-in_{\pi''}) \wedge \square(low-out_{\pi'} \leftrightarrow low-out_{\pi''})$$

For every two traces
there exists a third trace
that agrees with one on the high-security inputs,
and with the other on the low-security outputs

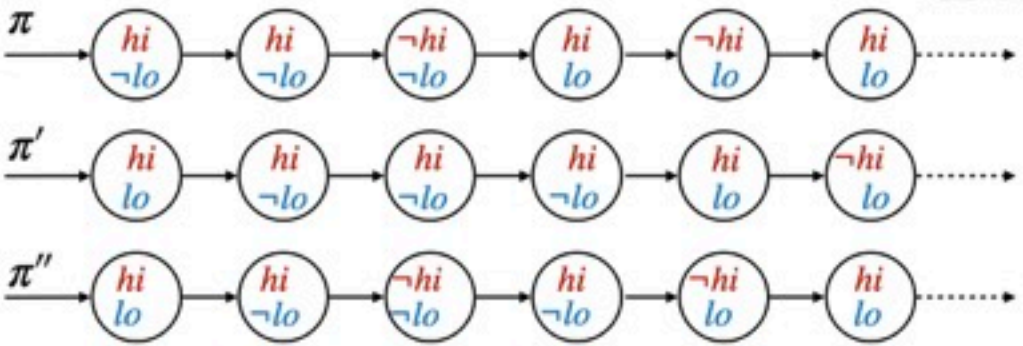


Model-Checking $\forall\exists$HyperLTL

Generalized Noninterference

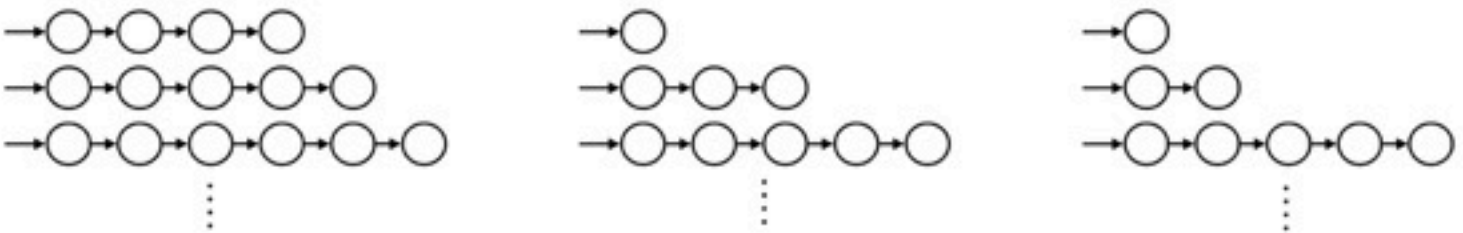
$$\forall\pi\forall\pi'\exists\pi''.\,\square(high-in_{\pi} \leftrightarrow high-in_{\pi''}) \wedge \square(low-out_{\pi'} \leftrightarrow low-out_{\pi''})$$

For every two traces
there exists a third trace
that agrees with one on the high-security inputs,
and with the other on the low-security outputs



Model-Checking $\forall\exists$HyperLTL

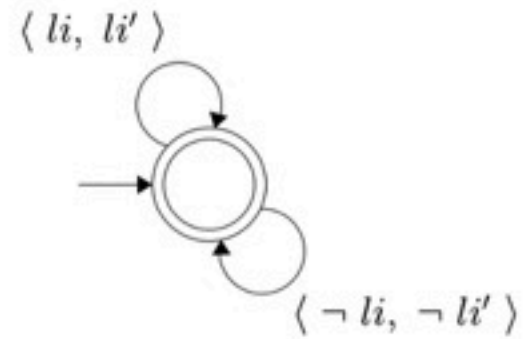
All infinite languages



for every trace
exists a longer trace
&
exists at least one trace

Model-Checking $\forall\exists$HyperLTL

$$\forall\pi\exists\pi' . \square(low - in_{\pi} \leftrightarrow low - in_{\pi'})$$



2 quantifiers, alphabet is $\Sigma \times \Sigma$

Construct automaton for the inner formula

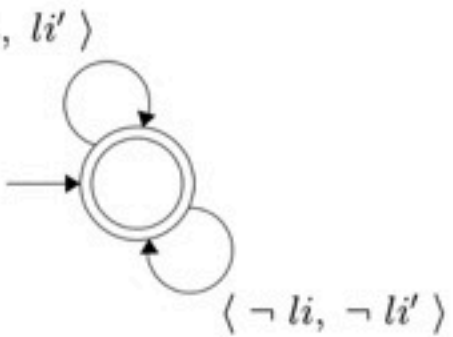
*Algorithms for Model Checking HyperLTL and HyperCTL**. Finkbeiner, Rabe, and Sánchez. CAV 2015.

Model-Checking $\forall\exists$HyperLTL

$$\forall\pi\exists\pi' . \square(low - in_{\pi} \leftrightarrow low - in_{\pi'})$$

We cannot directly use negation since we have two types of quantifiers

Integrate the quantifier into the construction



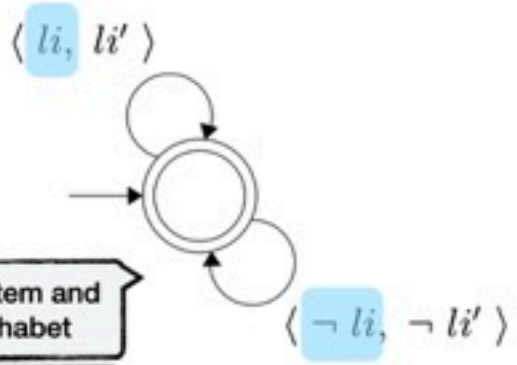
2 quantifiers, alphabet is $\Sigma \times \Sigma$

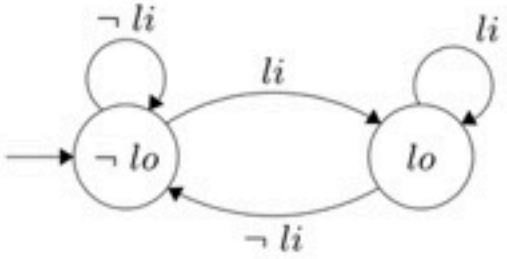
Construct automaton for the inner formula

*Algorithms for Model Checking HyperLTL and HyperCTL**. Finkbeiner, Rabe, and Sánchez. CAV 2015.

Model-Checking $\forall\exists$HyperLTL

$$\forall\pi\exists\pi' . \square (low - in_{\pi} \leftrightarrow low - in_{\pi'})$$





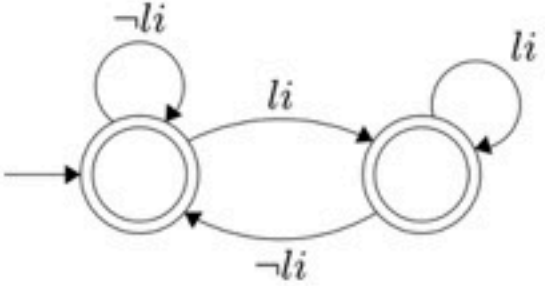
Intersect with the system and then reduce the alphabet

Integrate the quantifier into the construction

*Algorithms for Model Checking HyperLTL and HyperCTL**. Finkbeiner, Rabe, and Sánchez. CAV 2015.

Model-Checking $\forall\exists$HyperLTL

$$\forall\pi\exists\pi' . \square (low - in_{\pi} \leftrightarrow low - in_{\pi'})$$



All traces for which there exists a trace such that...

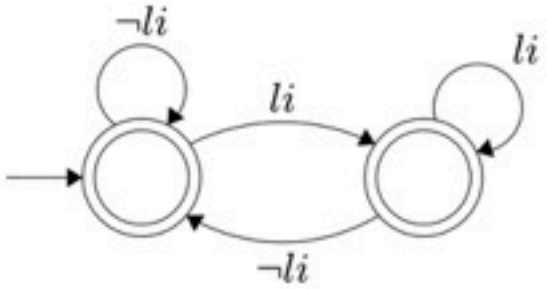
Intersect with the system and then reduce the alphabet

Integrate the quantifier into the construction

*Algorithms for Model Checking HyperLTL and HyperCTL**. Finkbeiner, Rabe, and Sánchez. CAV 2015.

Model-Checking $\forall\exists$HyperLTL

$$\forall\pi\exists\pi' . \square (low - in_{\pi} \leftrightarrow low - in_{\pi'})$$



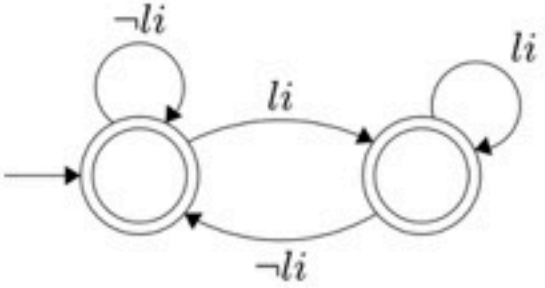
All traces for which there exists a trace such that...

∀ quantifier — looking for a violation

*Algorithms for Model Checking HyperLTL and HyperCTL**. Finkbeiner, Rabe, and Sánchez. CAV 2015.

Model-Checking $\forall\exists$HyperLTL

$$\forall\pi\exists\pi' . \square (low - in_{\pi} \leftrightarrow low - in_{\pi'})$$



All traces for which there exists a trace such that...

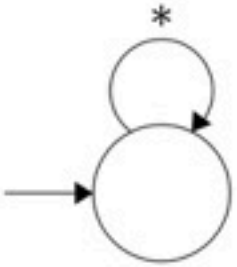
∀ quantifier — looking for a violation

Complement the automaton

*Algorithms for Model Checking HyperLTL and HyperCTL**. Finkbeiner, Rabe, and Sánchez. CAV 2015.

Model-Checking ∀∃HyperLTL

$$\exists \pi \neg \exists \pi' . \square (low - in_{\pi} \leftrightarrow low - in_{\pi'})$$



All traces for which there exists a trace such that...

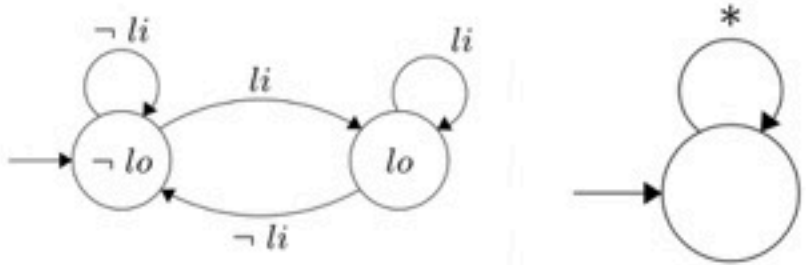
∀ quantifier — looking for a violation

Complement the automaton

*Algorithms for Model Checking HyperLTL and HyperCTL**. Finkbeiner, Rabe, and Sánchez. CAV 2015.

Model-Checking $\forall\exists$HyperLTL

$$\exists\pi \neg \exists\pi' . \square (low - in_{\pi} \leftrightarrow low - in_{\pi'})$$



Intersect with the system to find a violation trace

All traces for which there exists a trace such that...

∀ quantifier — looking for a violation

Complement the automaton

*Algorithms for Model Checking HyperLTL and HyperCTL**. Finkbeiner, Rabe, and Sánchez. CAV 2015.

Model-Checking $\forall\exists$HyperLTL

$$\forall \pi_1 \exists \pi_2 . \psi$$

1. A_ψ — an automaton for the inner formula ψ, over 2-tuples of letters
2. M_k — a k self-composition of the system
3. Intersect $A_\psi \cap M_k$
4. A' — Preform alphabet reduction to receive all traces π_1 for which there exists a trace π_2 for which…
5. Complement A' to find a trace π_1 for which there does not exists π_2
6. Check A' for emptiness

*Algorithms for Model Checking HyperLTL and HyperCTL**. Finkbeiner, Rabe, and Sánchez. CAV 2015.

Model-Checking $\forall\exists$ HyperLTL

$$\forall \pi_1 \exists \pi_2 . \psi$$

1. A_ψ — an automaton for the inner formula ψ, over 2-tuples of letters
2. M_k — a k self-composition of the system
3. Intersect $A_\psi \cap M_k$
4. A' — Preform alphabet reduction to receive all traces π_1 for which there exists a trace π_2 for which…
5. **Complement** A' to find a trace π_1 for which there does not exists π_2
6. Check A' for emptiness

*Algorithms for Model Checking HyperLTL and HyperCTL**. Finkbeiner, Rabe, and Sánchez. CAV 2015.

Model-Checking HyperLTL

$$\forall^* \exists^* \forall^* \exists^* . \psi$$

*Algorithms for Model Checking HyperLTL and HyperCTL**. Finkbeiner, Rabe, and Sánchez. CAV 2015.

Model-Checking HyperLTL

$\forall^* \exists^* \forall^* \exists^* . \psi$

Each alternation requiers complementation

Each alternation adds one exponent

*Algorithms for Model Checking HyperLTL and HyperCTL**. Finkbeiner, Rabe, and Sánchez. CAV 2015.

Model-Checking HyperLTL

$$\forall^* \exists^* \forall^* \exists^* . \psi$$

Each alternation requiers complementation

Each alternation adds one exponent

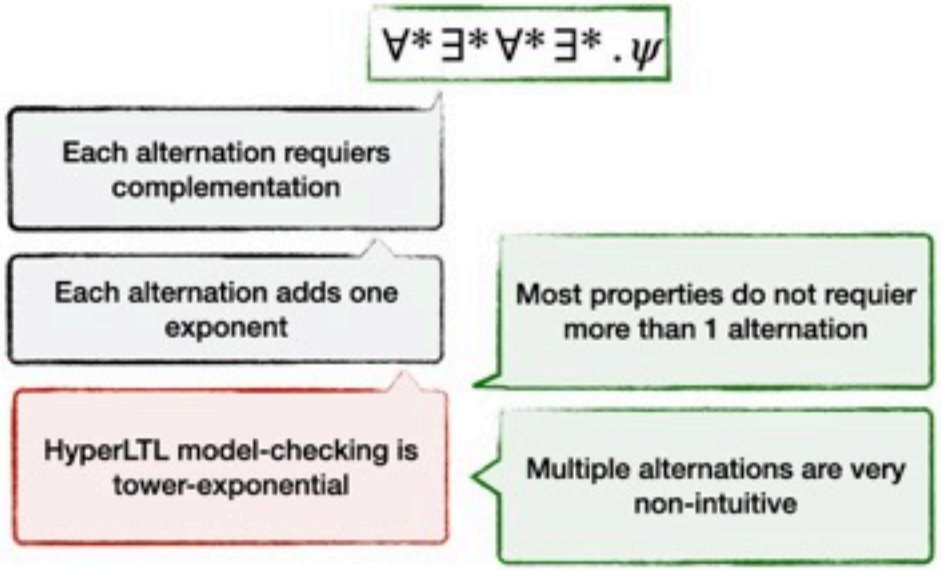
HyperLTL model-checking is tower-exponential

Inner formula into a non-deterministic Büchi automaton:
Exponential in the size of the inner formula

Every alternation:
Exponential in the size of the previous automaton

*Algorithms for Model Checking HyperLTL and HyperCTL**. Finkbeiner, Rabe, and Sánchez. CAV 2015.

Model-Checking HyperLTL



*Algorithms for Model Checking HyperLTL and HyperCTL**. Finkbeiner, Rabe, and Sánchez. CAV 2015.

Demonstration

Hyperproperties are Important But Hard!

Outline

- Trace properties & Hyperproperties
- HyperLTL - a logic for temporal hyperproperties
- Some interesting properties in HyperLTL
- HyperLTL model-checking
  - $\exists^*$-HyperLTL
  - $\forall^*$-HyperLTL
  - Quantifier alternation
- HyperLTL satisfiability
- Beyond HyperLTL and security

Satisfiability of HyperLTL

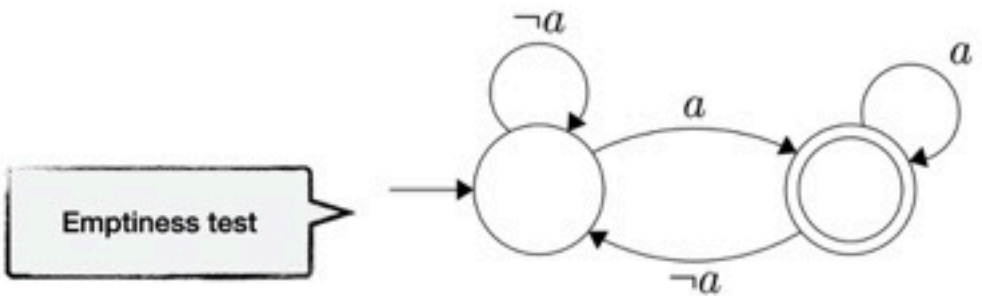
Satisfiability of LTL

Given an LTL formula ψ
is there a trace that satisfies ψ?

Satisfiability of LTL

Given an LTL formula ψ
is there a trace that satisfies ψ?

$\square\Diamond a$



Satisfiability of HyperLTL

Given a HyperLTL formula ψ
is there a set of traces that satisfies ψ?

Satisfiability of HyperLTL

Given a HyperLTL formula ψ
is there a set of traces that satisfies ψ?

Undecidable in general :(

We can encode a PCP
(Post Correspondence Problem)
instance into a $\forall\exists$ HyperLTL formula ψ
such that ψ is satisfiable iff there is a
solution to the PCP instance

Satisfiability of HyperLTL

Given a HyperLTL formula ψ
is there a set of traces that satisfies ψ?

Undecidable in general :(

We can encode a PCP
(Post Correspondence Problem)
instance into a $\forall\exists$ HyperLTL formula ψ
such that ψ is satisfiable iff there is a
solution to the PCP instance

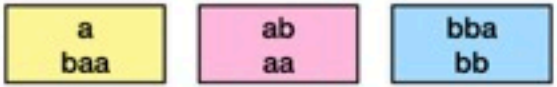
a baa	ab aa	bba bb

Satisfiability of HyperLTL

Given a HyperLTL formula ψ
is there a set of traces that satisfies ψ?

Undecidable in general :(

We can encode a PCP
(Post Correspondence Problem)
instance into a $\forall\exists$ HyperLTL formula ψ
such that ψ is satisfiable iff there is a
solution to the PCP instance



Deciding Hyperproperties. Finkbeiner and Hahn, CONCUR 2016

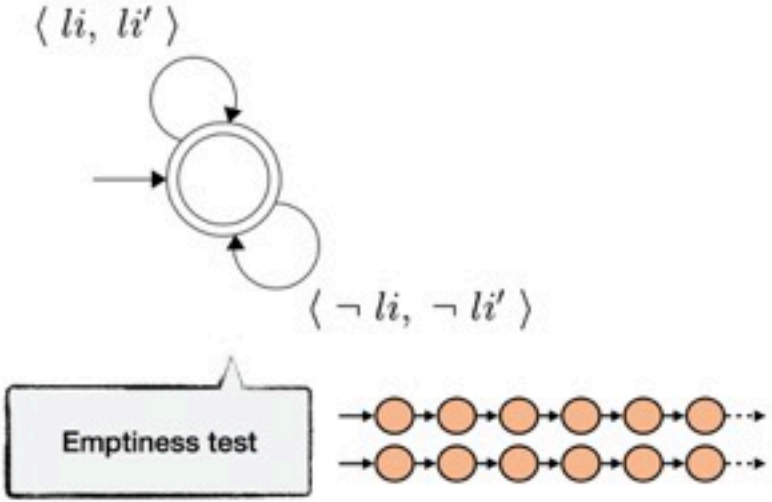
Satisfiability of $\exists^*$HyperLTL

$$\exists \pi \exists \pi' . \square (low - in_\pi \leftrightarrow low - in_{\pi'})$$

Deciding Hyperproperties. Finkbeiner and Hahn, CONCUR 2016

Satisfiability of $\exists^*$HyperLTL

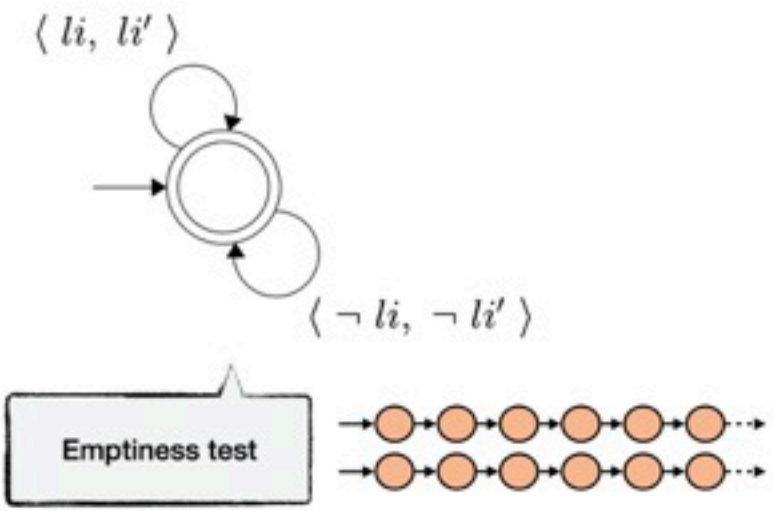
$$\exists \pi \exists \pi' . \square (low - in_{\pi} \leftrightarrow low - in_{\pi'})$$

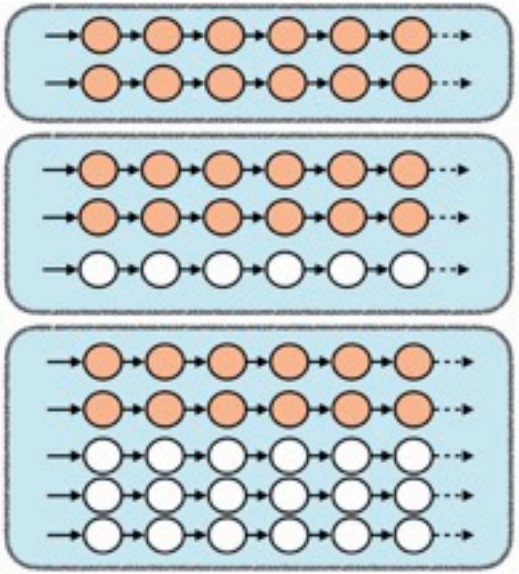


Deciding Hyperproperties. Finkbeiner and Hahn, CONCUR 2016

Satisfiability of $\exists^*$HyperLTL

$$\exists \pi \exists \pi' . \square (low - in_\pi \leftrightarrow low - in_{\pi'})$$





Deciding Hyperproperties. Finkbeiner and Hahn, CONCUR 2016

Satisfiability of $\forall^*$HyperLTL

$$\forall \pi \forall \pi' . \Diamond a_\pi \wedge \Diamond b_{\pi'}$$

If this holds, then it's true also for $\pi = \pi'$

It is enough to consider singelton models

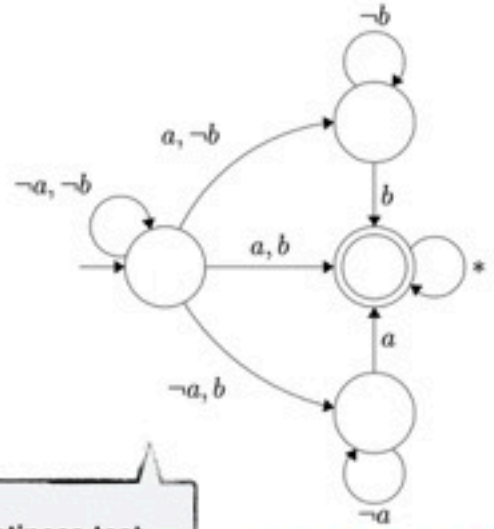
Deciding Hyperproperties. Finkbeiner and Hahn, CONCUR 2016

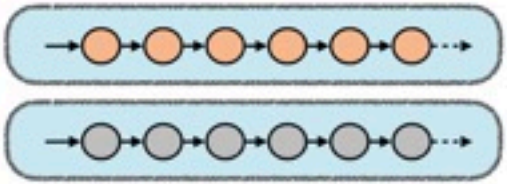
Satisfiability of $\forall^*$HyperLTL

$$\forall \pi \forall \pi' . \Diamond a_\pi \wedge \Diamond b_{\pi'}$$

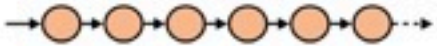
If this holds, then it's true also for $\pi = \pi'$

It is enough to consider singelton models





Emptiness test



Deciding Hyperproperties. Finkbeiner and Hahn, CONCUR 2016

Satisfiability of $\exists^* \forall^*$ HyperLTL

$$\exists \pi \exists \pi' \forall \pi'' . \Diamond a_\pi \wedge \Diamond b_{\pi'} \wedge \Diamond c_{\pi''}$$

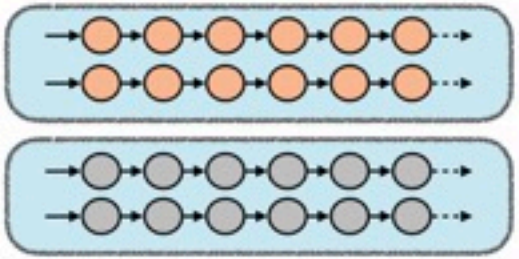
Consider all possible assignments for π''

$$\exists \pi \exists \pi' . (\Diamond a_\pi \wedge \Diamond b_{\pi'} \wedge \Diamond c_\pi)$$

$$\wedge (\Diamond a_\pi \wedge \Diamond b_{\pi'} \wedge \Diamond c_{\pi'})$$

Deciding Hyperproperties. Finkbeiner and Hahn, CONCUR 2016

Satisfiability of $\exists^* \forall^*$ HyperLTL



$$\exists \pi \exists \pi' \forall \pi'' . \Diamond a_\pi \wedge \Diamond b_{\pi'} \wedge \Diamond c_{\pi''}$$

Consider all possible assignments for π''

$$\exists \pi \exists \pi' . (\Diamond a_\pi \wedge \Diamond b_{\pi'} \wedge \Diamond c_\pi)$$

$$\wedge (\Diamond a_\pi \wedge \Diamond b_{\pi'} \wedge \Diamond c_{\pi'})$$

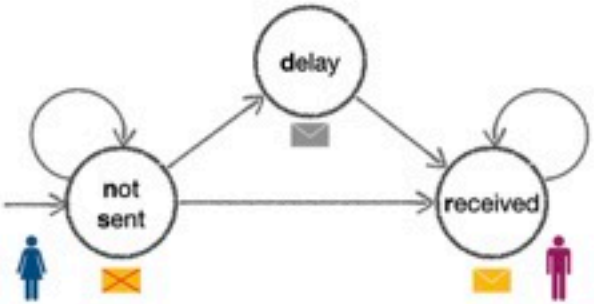
Deciding Hyperproperties. Finkbeiner and Hahn, CONCUR 2016

Outline

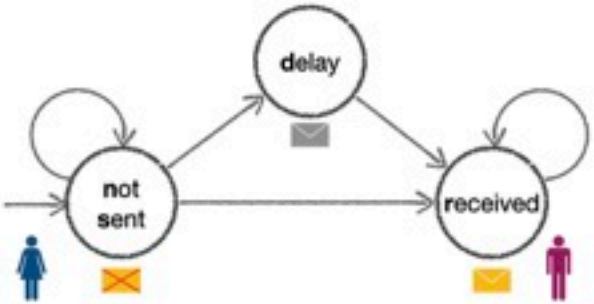
- Trace properties & Hyperproperties
- HyperLTL - a logic for temporal hyperproperties
- Some interesting properties in HyperLTL
- HyperLTL model-checking
  - $\exists^*$-HyperLTL
  - $\forall^*$-HyperLTL
  - Quantifier alternation
- HyperLTL satisfiability
- Beyond HyperLTL and security

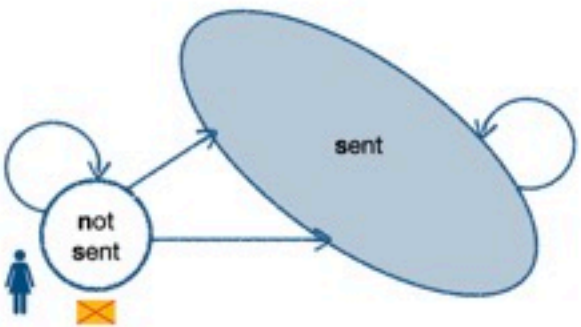
Knowledge and Common Knowledge in Multi-Agent Systems

Example: Knowledge

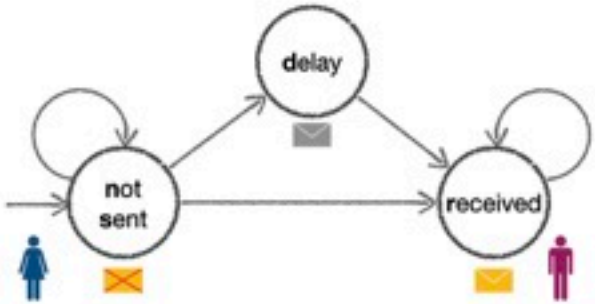


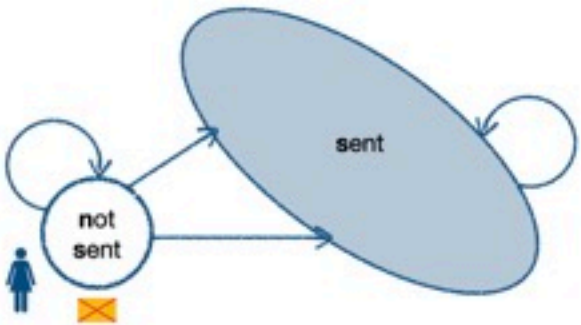
Example: Knowledge

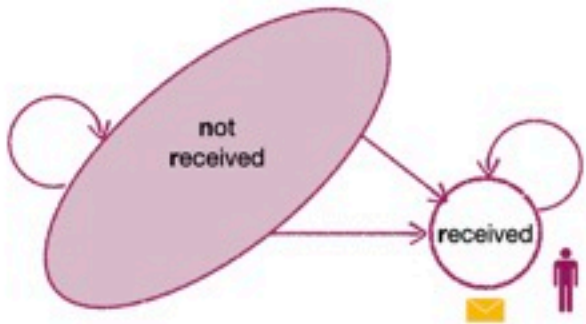




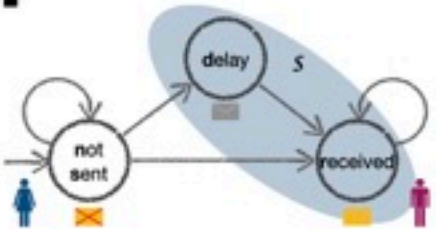
Example: Knowledge

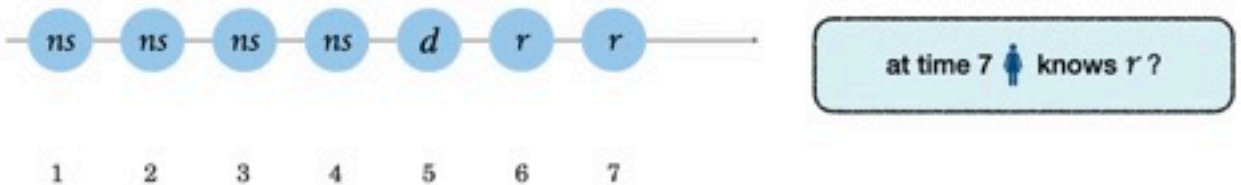




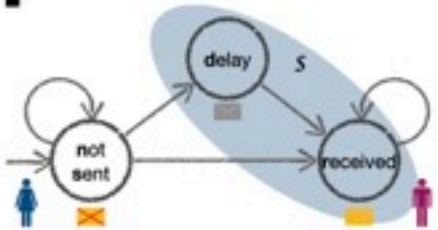


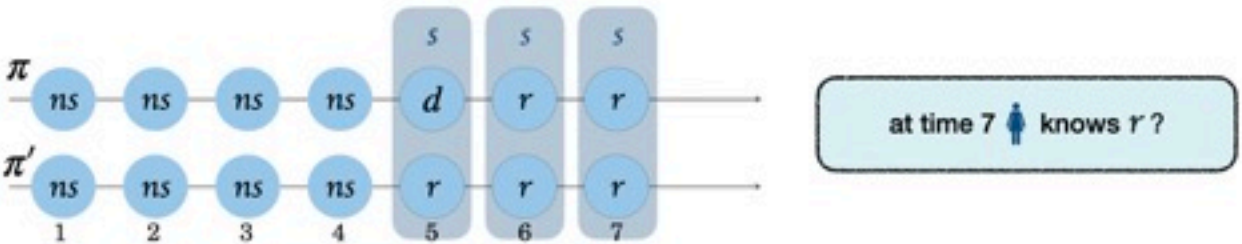
Example: Knowledge



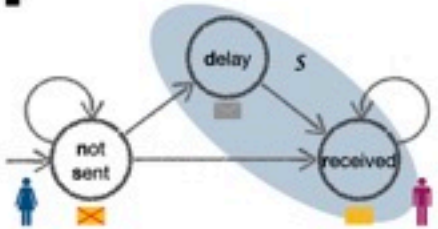


Example: Knowledge





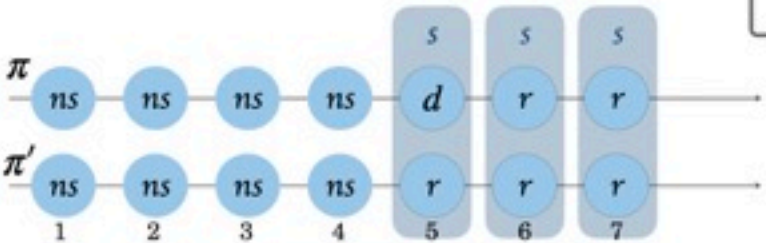
Example: Knowledge



Hyperproperty

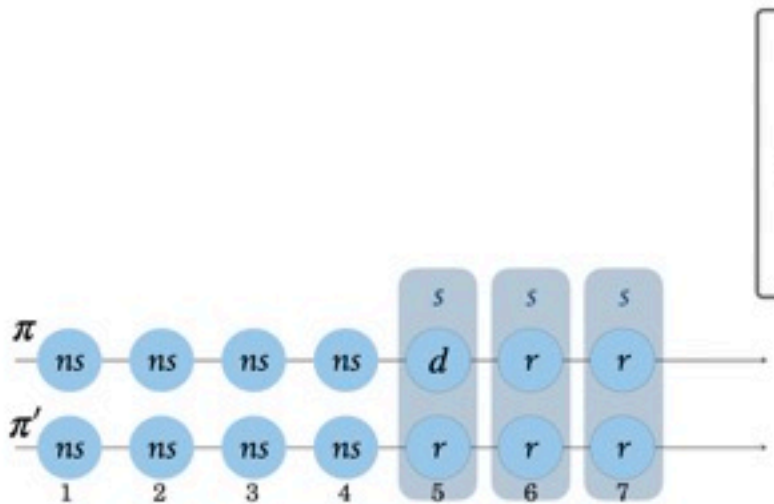
for every trace π'
that 🚹 cannot distinguish from π
until time 7
r holds at time 7

at time 7 🚹 knows r?



Knowledge in HyperLTL

$$\forall \pi \forall \pi' . (\pi \equiv_{\text{Alice}} \pi') \rightarrow \bigcirc^7 r_{\pi'}$$



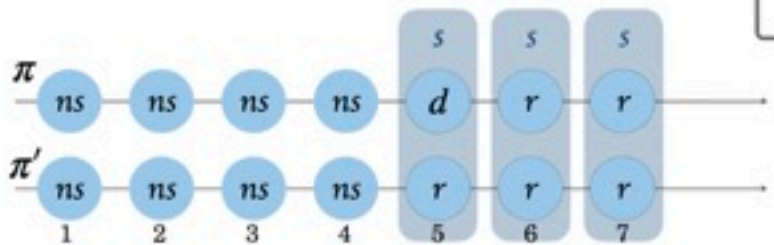
Knowledge in HyperLTL

$$\forall \pi \forall \pi' . (\pi \equiv_{\text{🚹}} \pi') \rightarrow \bigcirc^7 r_{\pi'}$$

$$\bigwedge_{i=1..7} \bigcirc^i \left((ns_\pi \leftrightarrow ns_{\pi'}) \wedge (s_\pi \leftrightarrow s_{\pi'}) \right)$$

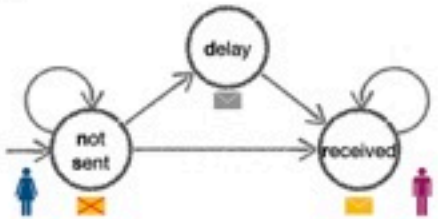
Hyperproperty

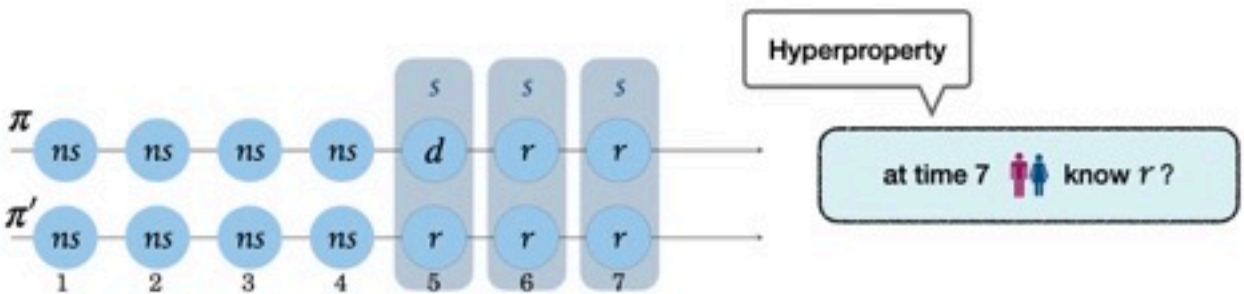
for every trace π'
that 🚹 cannot distinguish from π
until time 7
r holds at time 7



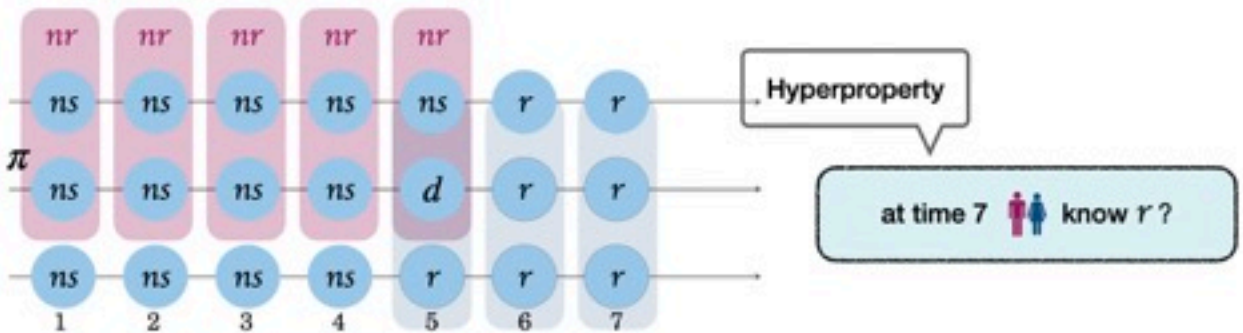
at time 7 🚹 knows r ?

Example: Knowledge

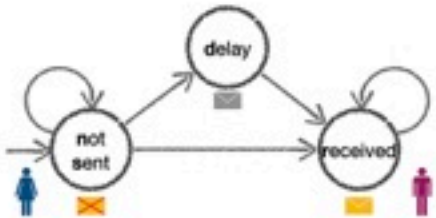




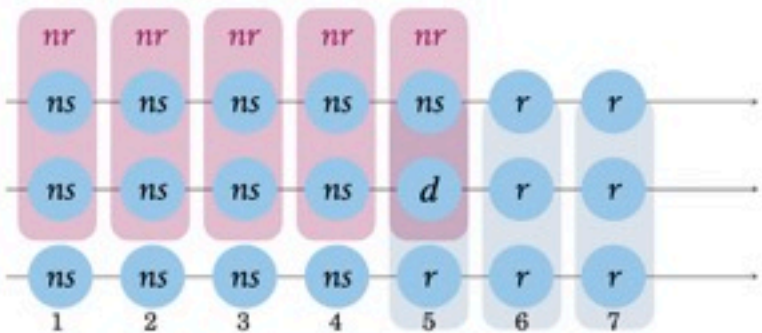
Example: Knowledge



Example: Common Knowledge



r is common knowledge at time 7 among  ?



Common Knowledge

φ common knowledge

$\leftrightarrow$

Reasoning About Knowledge. Fagin, Halpern, Moses, Vardi (MIT Press 1995)

Common Knowledge and Update in Finite Environments. van der Meyden (Inf. Comput. 1998)

Knowledge and common knowledge in a distributed environment. Halpern and Moses (J.ACM 1990)

Common Knowledge

φ common knowledge

first-order trace quantification is not enough

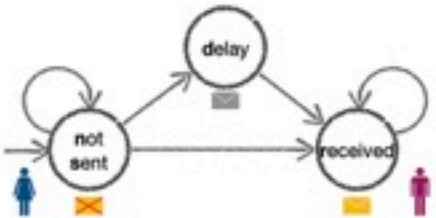
$\leftrightarrow$

know that know that know that

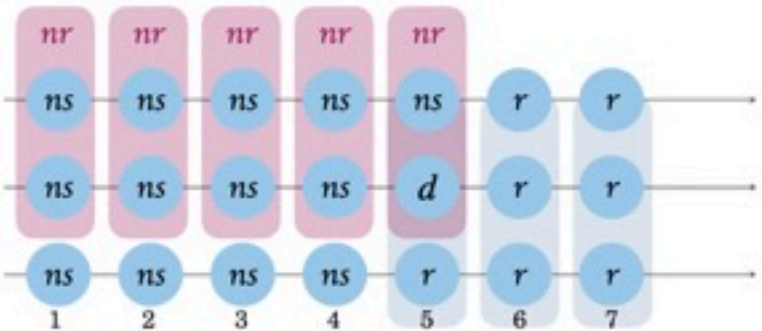
know that know that $\cdots$ φ

Unifying hyper and epistemic temporal logics. Bozzelli, Maubert, Pinchinat (FoSSaCS 2015)

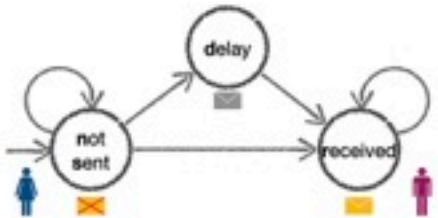
Example: Common Knowledge



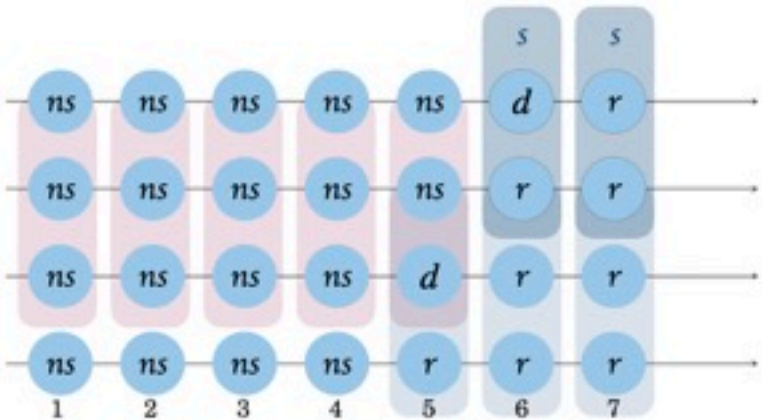
r is common knowledge at time 7 among  ?



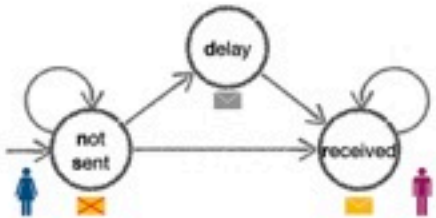
Example: Common Knowledge

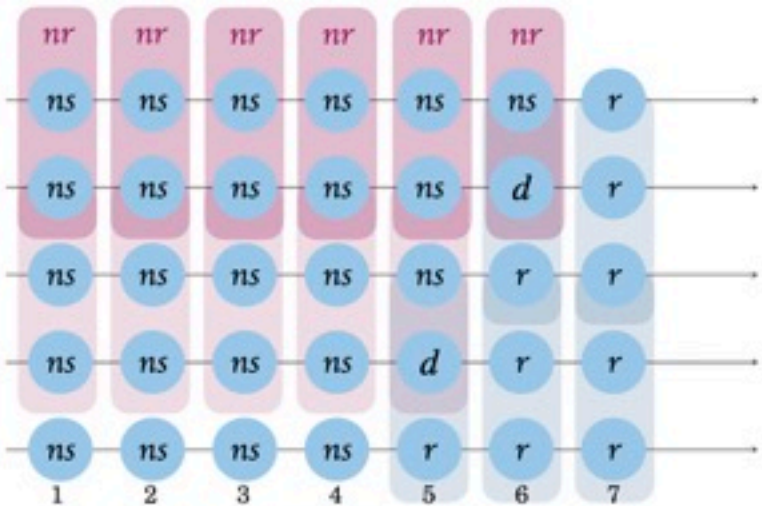


r is common knowledge at time 7 among  ?



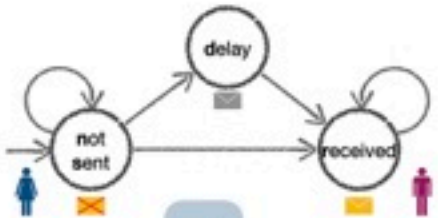
Example: Common Knowledge

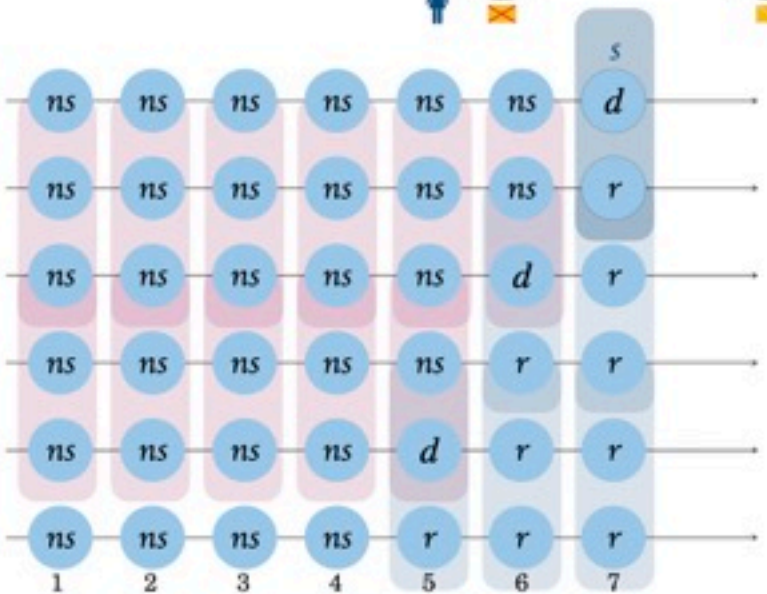




r is common knowledge at time 7 among 👫?

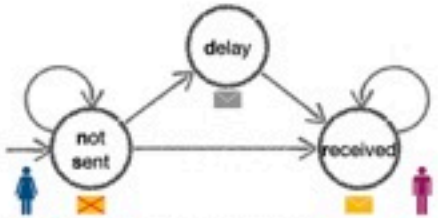
Example: Common Knowledge

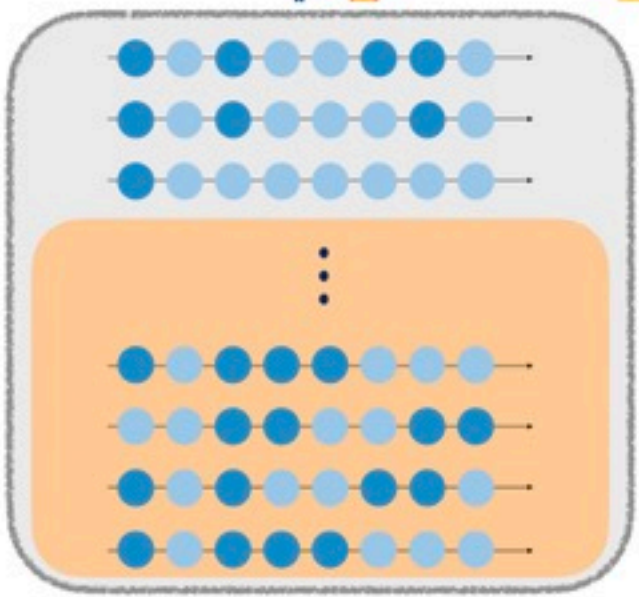




r is common knowledge at time 7 among 👫?

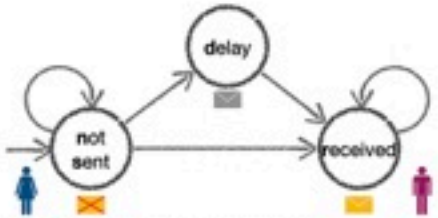
Example: Common Knowledge

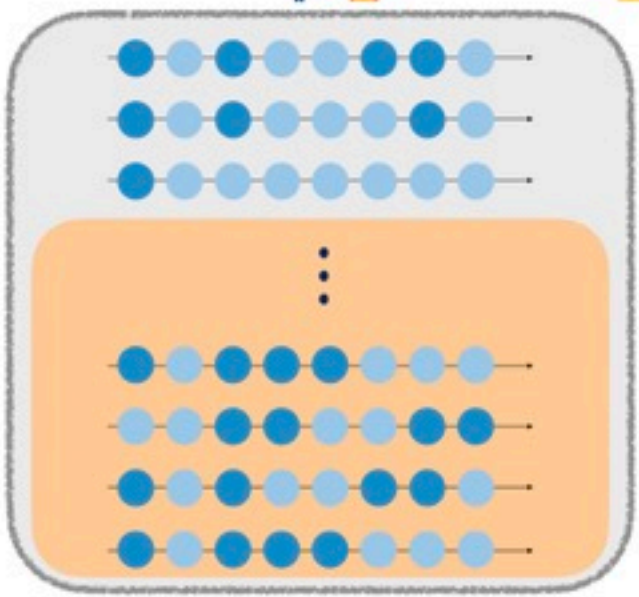




r is common knowledge at time 7 among  ?

Example: Common Knowledge



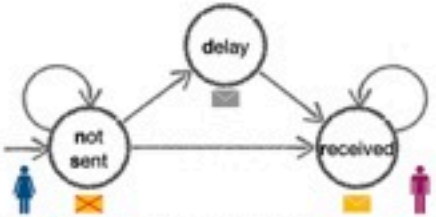


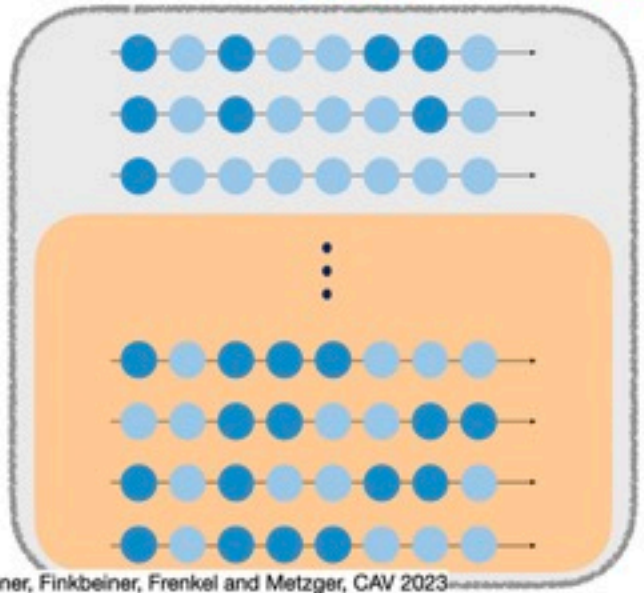
r is common knowledge at time 7 among  ?

1. Define the set X of traces of interest
   all indistinguishable traces

2. Verify the desired property on X
   r eventually holds

Example: Common Knowledge





Hyper^{2}LTL
Quantification over sets of traces

HyperLTL is not enough!

1. Define the set X of traces of interest
   all indistinguishable traces

2. Verify the desired property on X
   r eventually holds

Second-Order Hyperproperties. Beutner, Finkbeiner, Frenkel and Metzger, CAV 2023

Outline

- Trace properties & Hyperproperties
- HyperLTL - a logic for temporal hyperproperties
- Some interesting properties in HyperLTL
- HyperLTL model-checking
  - $\exists^*$-HyperLTL
  - $\forall^*$-HyperLTL
  - Quantifier alternation
- HyperLTL satisfiability
- Beyond HyperLTL and security

Summary

Hyperproperties are important!

Initially for information-flow, but not only

We need new formalisms to express hyperproperties

And verification algorithms…

Summary

Hyperproperties are important!

Initially for information-flow, but not only

We need new formalisms to express hyperproperties

HyperLTL is a logic to express temporal hyperproperties

And verification algorithms…

Summary

Hyperproperties are important!

Initially for information-flow, but not only

We need new formalisms to express hyperproperties

And verification algorithms…

HyperLTL is a logic to express temporal hyperproperties

Captures a wide range of information-flow and security properties

Summary

Hyperproperties are important!

Initially for information-flow, but not only

We need new formalisms to express hyperproperties

And verification algorithms…

HyperLTL is a logic to express temporal hyperproperties

Captures a wide range of information-flow and security properties

Have a decidable (yet costly) model-checking algorithm

Summary

Hyperproperties are important!

Initially for information-flow, but not only

We need new formalisms to express hyperproperties

And verification algorithms…

HyperLTL is a logic to express temporal hyperproperties

Captures a wide range of information-flow and security properties

Have a decidable (yet costly) model-checking algorithm

Can be used in practice when there are not too many quantifier alternations