Canonical Representations of *k*-Safety Hyperproperties

Reactive Systems Group

Bernd Finkbeiner, Lennart Haas, Hazem Torfah
Saarland University

This paper

Representing
hyperproperties such as noninterference, observational determinism etc.
canonically as **automata**

- Foundation for automata-based **analysis** techniques such as **monitoring**,
- and for **constructive** techniques such as **learning**.

Hyperproperties

Sets

Hyperproperty
= a **set of sets of traces**
[Clarkson & Schneider '10]

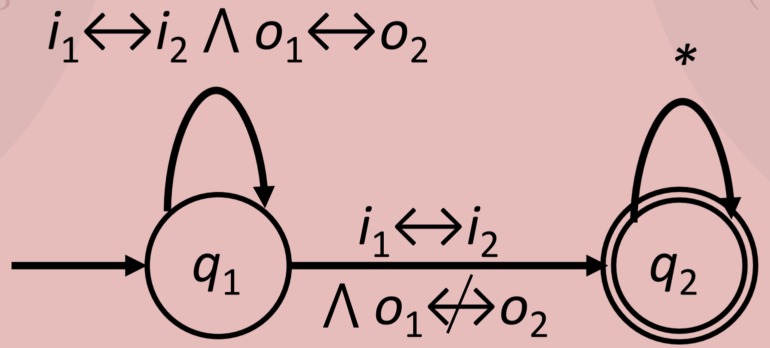
$$\{T \subseteq \Sigma^\omega \mid \forall t, t' \in T.\ t =_i t' \Rightarrow t =_o t'\}$$

Logic

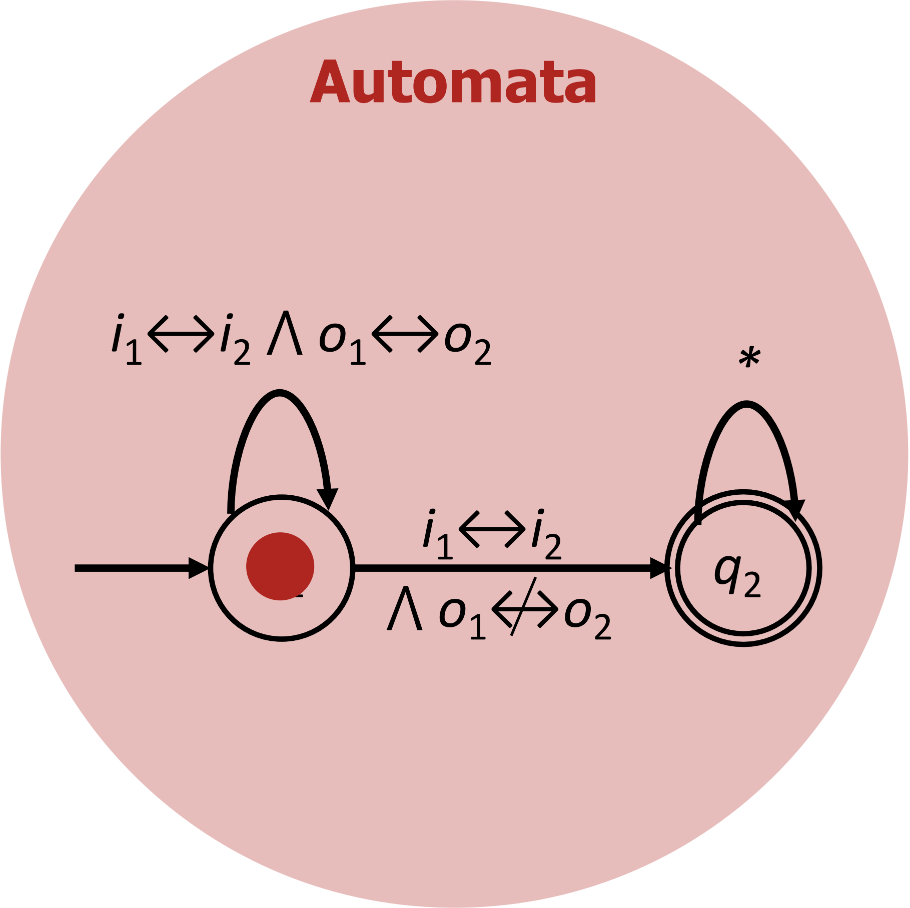
HyperLTL
= LTL + **trace quantifiers**

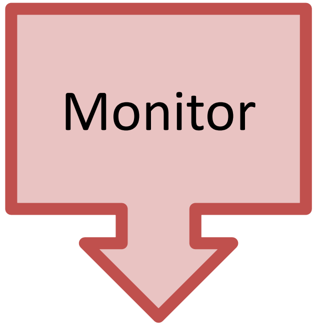
$$\forall \pi, \pi'.\ (o_\pi \leftrightarrow o_{\pi'}) \text{ W-Until } (i_\pi \not\leftrightarrow i_{\pi'})$$

Automata



Monitoring



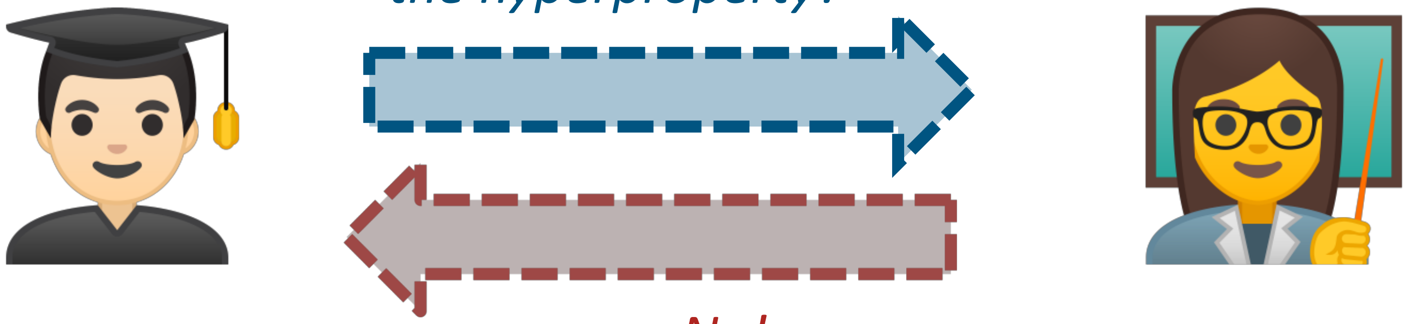


i	T	T	T	T	F	T	T	
o	T	T	T	T	T	F	T	…

i	T	T	T	T	F	T	F	
o	T	T	F	F	T	T	T	…

Automata provide an **operational semantics** for monitoring

Learning



Student

Does **not** know hyperproperty

Asks queries to teacher

Learns automaton from answers

Teacher

Knows hyperproperty

Answers queries

Can be human or automated

(e.g., HyperLTL tool)

Automata provide a **canonical representation** for learning

Bad prefixes

i	T	T	T	T	F	T	T	…
o	T	T	T	T	T	F	T	

i	T	T	T	T	F	T	F	…
o	T	T	F	F	T	T	T	

i	F	F	T	F	T	F	T	…
o	T	F	F	T	T	T	T	

i	T	F	T	T	F	T	T	…
o	T	F	F	T	T	F	T	

⋮

Bad prefixes

i	T	T	T
o	T	T	T

i	T	T	T
o	T	T	F

A **bad prefix** of a hyperproperty *H* is a **finite set** of **finite traces** such that **every extension** to a set of infinite traces **violates *H***.

A hyperproperty *H* is **safety** if every trace set that **violates** *H* has a **bad prefix**.

A hyperproperty *H* is ***k*-safety** if every trace set that violates *H* has a bad prefix **of size *k*.**

Representing sets of traces

{

i	T	T	T
o	T	T	T

i	T	T	T
o	T	T	F

}

T_1

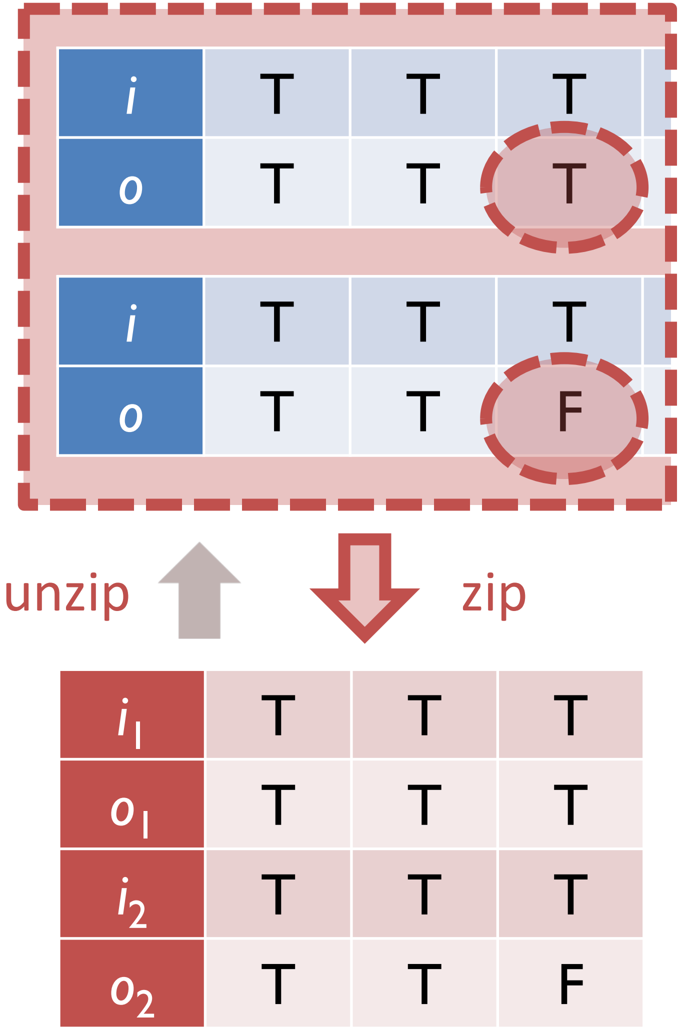
i	T	T	T	
o	T	T	T	

T_2

i	T	T	T	
o	T	T	F	

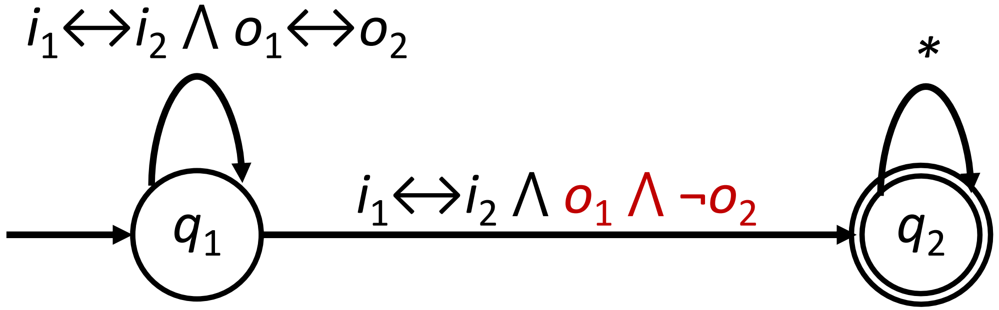
i_1	T	T	T
o_1	T	T	T
i_2	T	T	T
o_2	T	T	F

Representing hyperproperties



A **bad-prefix automaton** of a *k*-safety hyperproperty *H* is an automaton *A* over **finite words of *k*‘-tuples** such that, for every set of traces *T*,

T violates *H* iff $L(A)$ contains a word $\boldsymbol{\sigma}$ s.t. unzip($\boldsymbol{\sigma}$) is a prefix of *T*



Horizontal tightness

i	T	T	T	T	F	T	T	…
o	T	T	T	T	T	F	T	

i	T	T	T	T	F	T	F	…
o	T	T	F	F	T	T	T	

i_1	T	T	T
o_1	T	T	T
i_2	T	T	T
o_2	T	T	F

i_1	T	T	T	T
o_1	T	T	T	T
i_2	T	T	T	T
o_2	T	T	F	F

Vertical tightness

i	T	T	T	T	F	T	T	…
o	T	T	T	T	T	F	T	

i	T	T	T	T	F	T	F	…
o	T	T	F	F	T	T	T	

i	F	F	T	F	T	F	T	…
o	T	F	F	T	T	T	T	

i	T	F	T	T	F	T	T	…
o	T	F	F	T	T	F	T	

⋮

i_1	T	T	T
o_1	T	T	T
i_2	T	T	T
o_2	T	T	F

i_1	T	T	T
o_1	T	T	T
i_2	T	T	T
o_2	T	T	F
i_3	F	F	T
o_3	T	F	F

Tight automata

i	T	T	T	T	F	T	T	…
o	T	T	T	T	T	F	T	

i	T	T	T	T	F	T	F	…
o	T	T	F	F	T	T	T	

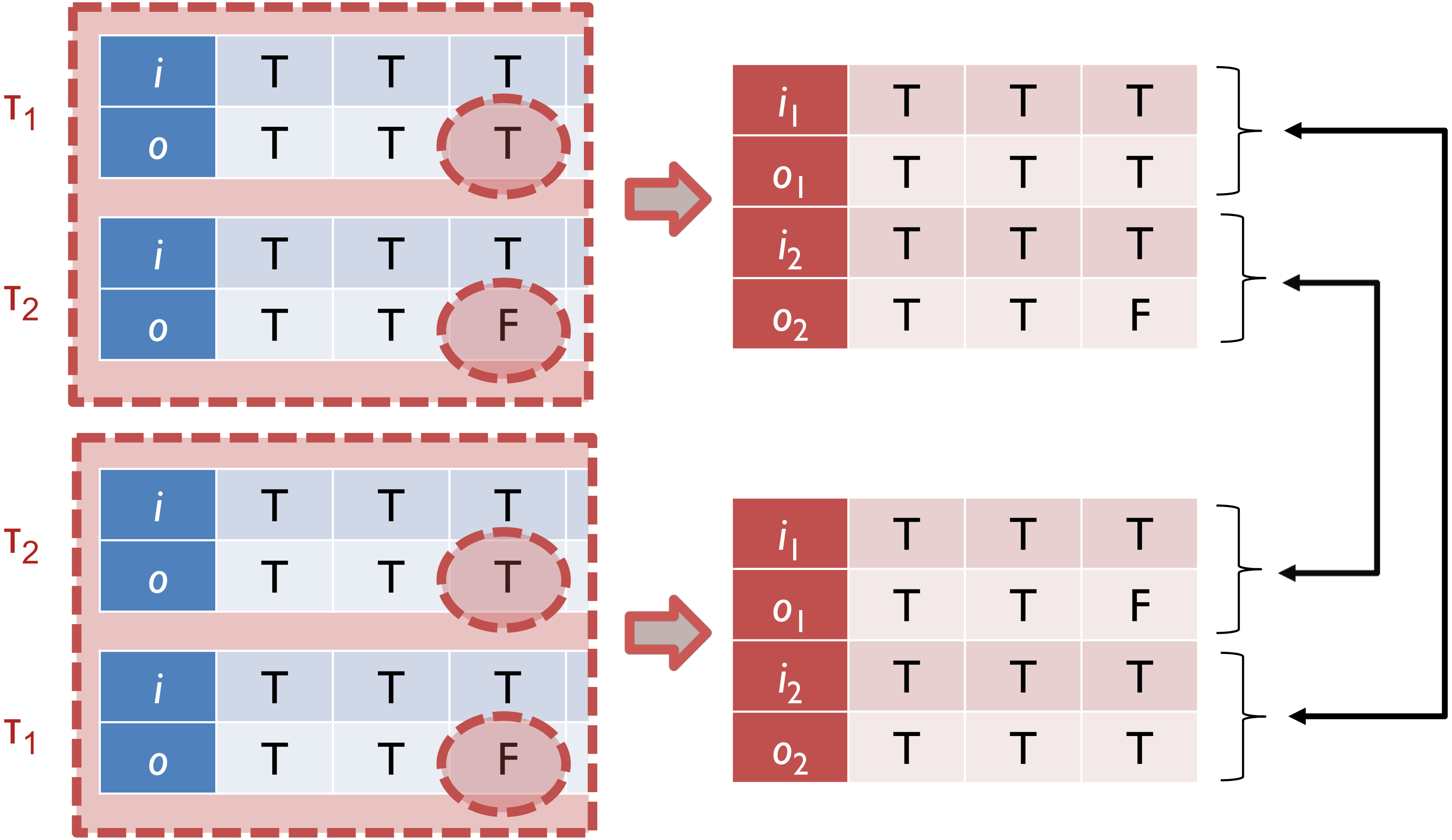
i	F	F	T	F	T	F	T	…
o	T	F	F	T	T	T	T	

i	T	F	T	T	F	T	T	…
o	T	F	F	T	T	F	T	

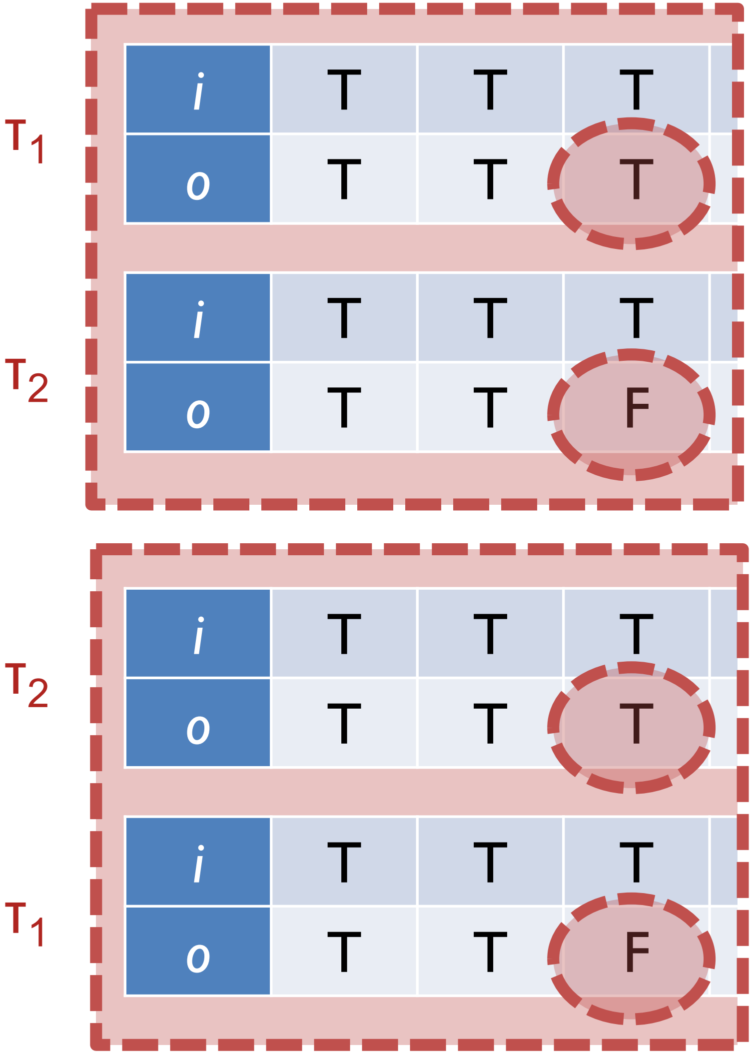
⋮

A *k*-bad-prefix automaton is **tight** iff it accepts **some representation** of **each bad prefix** of size $\leqq k$

Permutation completeness



Permutation completeness



A *k*-bad-prefix automaton is **tight**
iff it accepts **some representation**
for **each bad prefix** of size ≦ *k*

A *k*-bad-prefix automaton is **permutation-complete**
iff it accepts **every representation**
for **each bad prefix** it accepts.

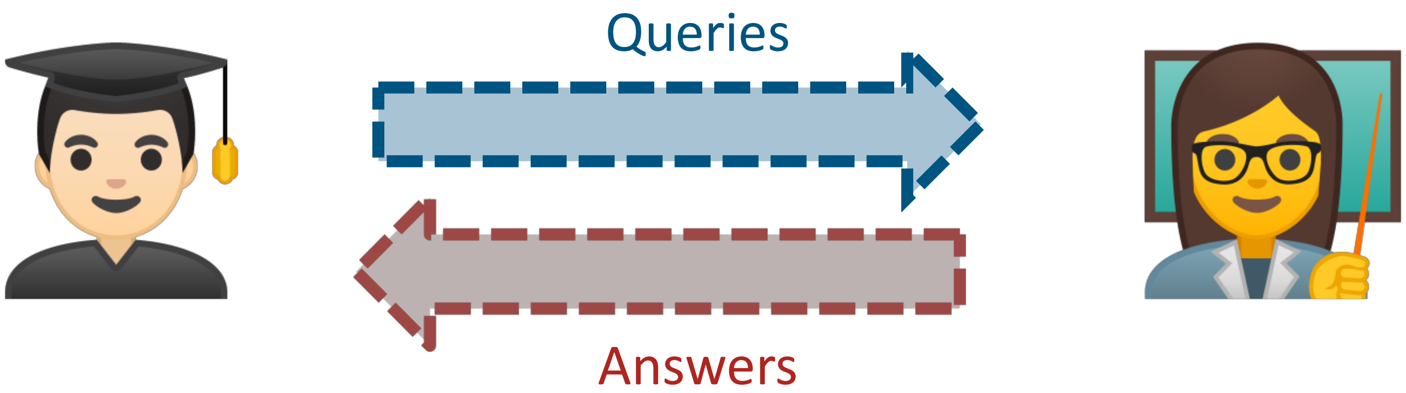
Theorem:
A minimal deterministic **tight permutation-complete** *k*-bad-prefix automaton is a **canonical representation** for (regular) *k*-safety hyperproperties.

Direct automata constructions

Construction:
For a k-safety hyperproperty S, we can construct
a **deterministic**, **tight**, and **permutation-complete** bad-prefix automaton of size

- **polynomial in** $|A|$ and **doubly exponential in** $k \cdot \log(k)$ if S is given as a **deterministic** bad-prefix automaton A
- **exponential in** $|A|$ and **doubly exponential in** $k \cdot \log(k)$ if S is given as a **nondeterministic** bad-prefix automaton A

Learning regular languages: L*



Teacher answers two types of queries:

- **Membership queries**
  Is the word σ in the language?

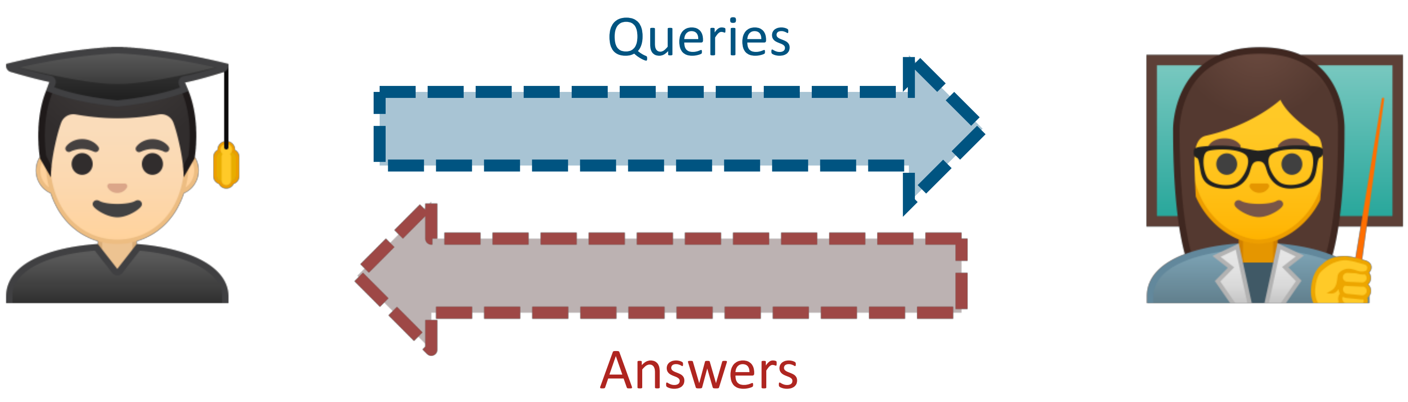
  Yes/No

- **Equivalence queries:**
  Does the automaton A recognize the language?

  Yes/No: counterexample

[Angluin 1987]

Learning hyperproperties: L* Hyper



Teacher answers two types of queries:

- **Membership queries**
  Does the trace set T satisfy the hyperproperty?
  Yes/No
- **Equivalence queries:**
  Is the automaton A a bad-prefix automaton for the hyperproperty?
  Yes/No: counterexample

Observation tables

Separating sequences $E \subseteq \Sigma^*$

		ϵ	$\neg a$
Accessing sequences $S \subseteq \Sigma^*$	ϵ	0	1
	$\neg a$	1	1
	a	0	0
	$a \cdot \neg a$	0	0
$S \cdot \Sigma$	$\neg a \cdot a$	1	1
	$\neg a \cdot \neg a$	1	1
	$a \cdot a$	0	0
	$a \cdot \neg a \cdot a$	0	0
	$a \cdot \neg a \cdot \neg a$	0	0

States

Observation table stores results of membership queries

accessing seq · separating seq

E.g. $\neg a \cdot \neg a$ is in the language

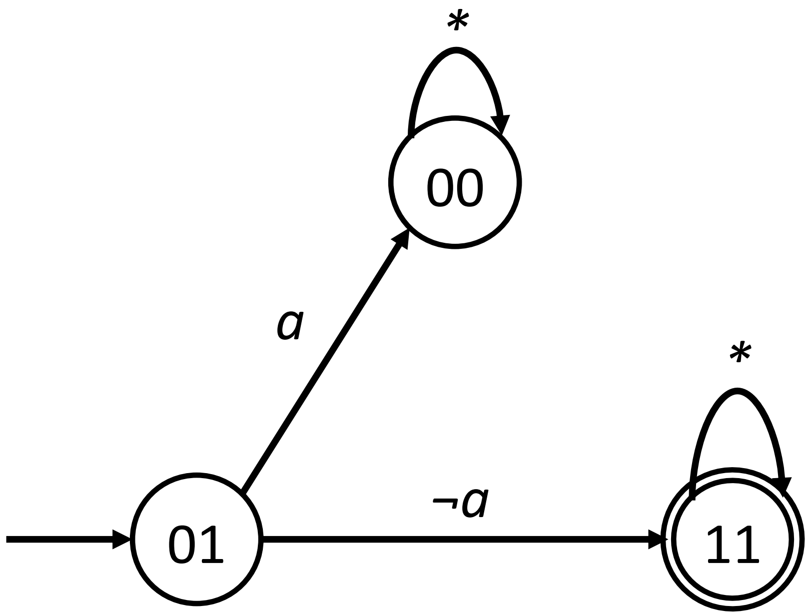
[Angluin 1987]

Observation tables

Separating sequences $E \subseteq \mathbf{\Sigma}^*$

		ϵ	$\neg a$
Accessing sequences $S \subseteq \mathbf{\Sigma}^*$	ϵ	0	1
	$\neg a$	1	1
	a	0	0
	$a \cdot \neg a$	0	0
$S \cdot \mathbf{\Sigma}$	$\neg a \cdot a$	1	1
	$\neg a \cdot \neg a$	1	1
	$a \cdot a$	0	0
	$a \cdot \neg a \cdot a$	0	0
	$a \cdot \neg a \cdot \neg a$	0	0

States



[Angluin 1987]

Observation tables

Separating sequences $E \subseteq \Sigma^*$

Accessing sequences $S \subseteq \Sigma^*$

	ϵ	$\neg a$
ϵ	0	1
$\neg a$	1	1
a	0	0
$a \cdot \neg a$	0	0
$\neg a \cdot a$	1	1
$\neg a \cdot \neg a$	1	1
$a \cdot a$	0	0
$a \cdot \neg a \cdot a$	0	0
$a \cdot \neg a \cdot \neg a$	0	0

$S \cdot \Sigma$

States

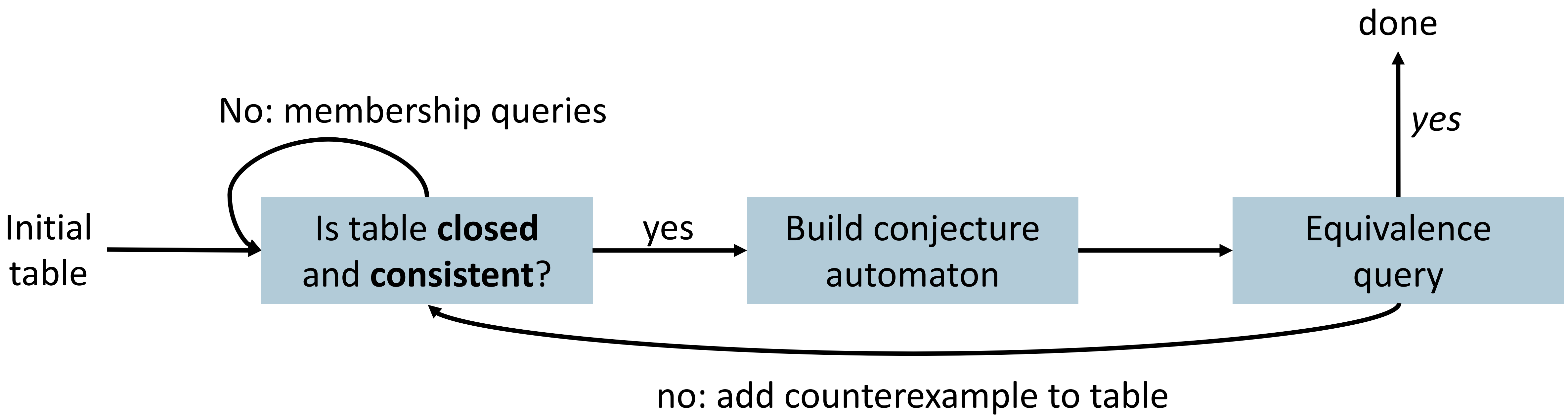
Closedness check:
For all $t \in S$, $e \in \Sigma$,
there is a $t' \in S$.
state($t \cdot e$)=state(t') ?

Consistency check:
For all $t, t' \in S$, $e \in \Sigma$.
state(t)=state(t') ⇒
state($t \cdot e$)=state($t' \cdot e$) ?

A **closed** and **consistent** observation table defines a **deterministic automaton.**

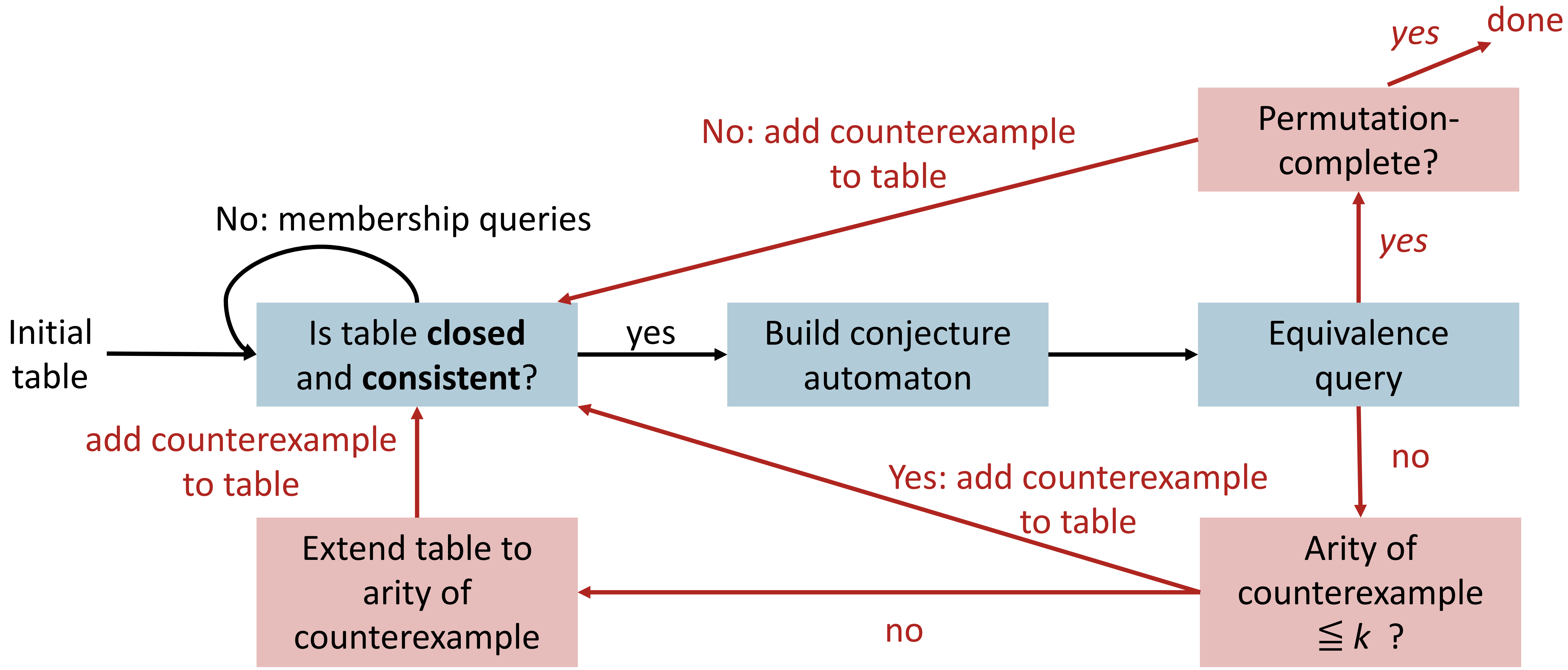
[Angluin 1987]

Learning regular languages: L*



[Angluin 1987]

Learning hyperproperties: L* Hyper

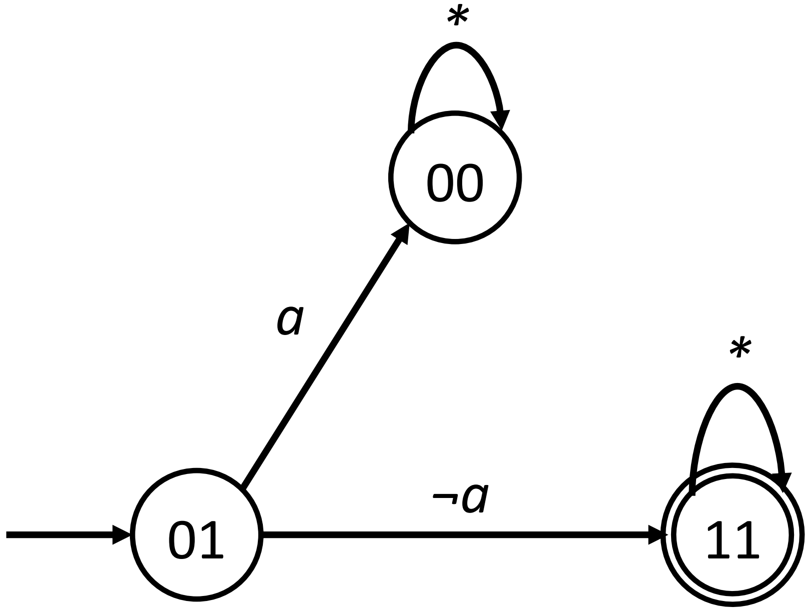


Example

$$\forall \pi, \pi'.\ a_\pi \land \text{ Globally } (a_\pi \leftrightarrow a_{\pi'})$$

Table for arity 1:

	ϵ	$\neg a$
ϵ	0	1
$\neg a$	1	1
a	0	0
$a \cdot \neg a$	0	0
$\neg a \cdot a$	1	1
$\neg a \cdot \neg a$	1	1
$a \cdot a$	0	0
$a \cdot \neg a \cdot a$	0	0
$a \cdot \neg a \cdot \neg a$	0	0



Equivalence query: Counterexample of arity 2, $\{a \cdot \neg a,\ a \cdot a\}$

Extending the arity

Table for arity 1:

$E \subseteq (\mathbf{\Sigma}^1)^*$

	ϵ	$\neg a$
ϵ	0	1
$\neg a$	1	1
a	0	0
$a \cdot \neg a$	0	0
$\neg a \cdot a$	1	1
$\neg a \cdot \neg a$	1	1
$a \cdot a$	0	0
$a \cdot \neg a \cdot a$	0	0
$a \cdot \neg a \cdot \neg a$	0	0

Rows ϵ through $a \cdot \neg a$: $S \subseteq (\mathbf{\Sigma}^1)^*$

Rows $\neg a \cdot a$ through $a \cdot \neg a \cdot \neg a$: $S \cdot \mathbf{\Sigma}^1$

$E \subseteq (\mathbf{\Sigma}^2)^*$

Table for arity 2:

Repeat last tuple element to turn 1-tuple into 2-tuple

Extending the arity

Table for arity 1:

$E \subseteq (\Sigma^1)^*$

	ϵ	$\neg a$
ϵ	0	1
$\neg a$	1	1
a	0	0
$a \cdot \neg a$	0	0
$\neg a \cdot a$	1	1
$\neg a \cdot \neg a$	1	1
$a \cdot a$	0	0
$a \cdot \neg a \cdot a$	0	0
$a \cdot \neg a \cdot \neg a$	0	0

$S \subseteq (\Sigma^1)^*$ (rows ϵ, $\neg a$, a, $a \cdot \neg a$); $S \cdot \Sigma^1$ (rows $\neg a \cdot a$ … $a \cdot \neg a \cdot \neg a$)

Table for arity 2:

$E \subseteq (\Sigma^2)^*$	ϵ	$(\neg a, \neg a)$

ϵ
$(\neg a,\neg a)$
$\epsilon \cdot (a,a)$
$(a,a) \cdot (\neg a,\neg a)$
$(\neg a,\neg a) \cdot (a,a)$
$(\neg a,\neg a) \cdot (\neg a,\neg a)$
$(a,a) \cdot (a,a)$
$(a,a) \cdot (\neg a,\neg a) \cdot (a,a)$
$(a,a) \cdot (\neg a,\neg a) \cdot (\neg a,\neg a)$

$S' \subseteq (\Sigma^2)^*$ (first four rows); $S' \cdot \Sigma^2$ (last five rows)

element
to 2-tuple

Extending the arity

Table for arity 1:

0	1
1	1
0	0
0	0
1	1
1	1
0	0
0	0
0	0
ϵ	$(\neg a, \neg a)$

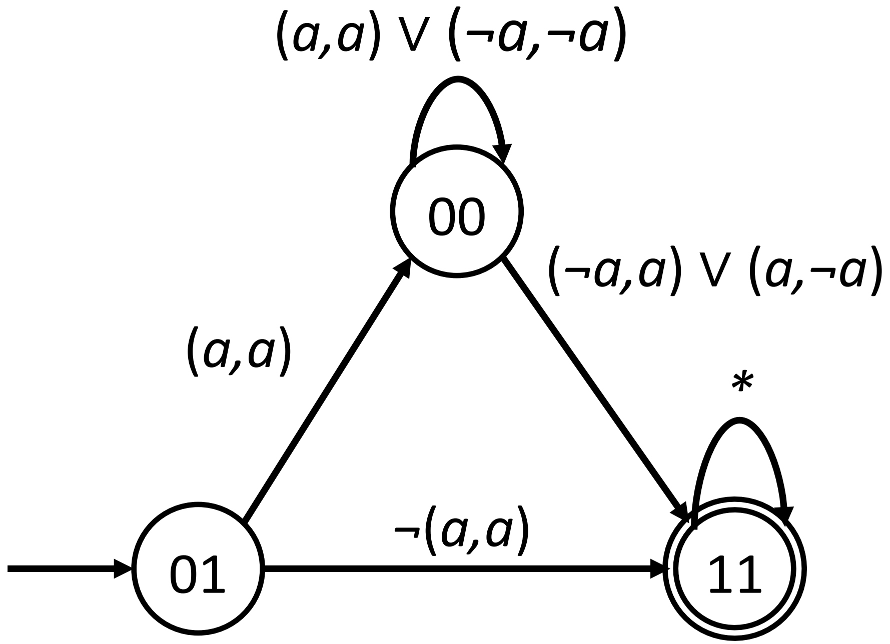
Table for arity 2:

ϵ
$(\neg a,\neg a)$
$\epsilon \cdot (a,a)$
$(a,a) \cdot (\neg a,\neg a)$
$(\neg a,\neg a) \cdot (a,a)$
$(\neg a,\neg a) \cdot (\neg a,\neg a)$
$(a,a) \cdot (a,a)$
$(a,a) \cdot (\neg a,\neg a) \cdot (a,a)$
$(a,a) \cdot (\neg a,\neg a) \cdot (\neg a,\neg a)$

Closed and consistent observation table with added counterexample

	ϵ	(¬a,¬a)
ϵ	0	1
(¬a,¬a)	1	1
(a,a)	0	0
(a,a) · (¬a,¬a)	0	0
(a,¬a)	1	1
(¬a,a)	1	1
(¬a,¬a) · (*)	1	1
(a,a) · (a,a)	0	0
(a,a) · (¬a,¬a) · (a,a)	0	0
(a,a) · (¬a,¬a) · (¬a,a)	1	1
(a,a) · (¬a,¬a) · (a,¬a)	1	1
(a,a) · (¬a,¬a) · (¬a,¬a)	0	0
(a,a) · (¬a,a) · (*)	1	1

$$\forall \pi, \pi'.\ a_\pi \wedge \text{ Globally } (a_\pi \leftrightarrow a_{\pi'})$$



Complexity

Theorem:
L* Hyper learns a **minimal, deterministic**, and **permutation-complete** automaton A for a k-safety hyperproperty in

- **polynomial time** in $|A|$
- **polynomial time** in the length of the longest counterexample, and
- **exponential time** in k

Application: Learning Automata for HyperLTL

Membership queries for a set T of traces of length n and
a hyperproperty given as a universal safety HyperLTL formula $\boldsymbol{\varphi}$ with k quantifiers
can be answered in

- **polynomial time** in $\boldsymbol{n}$ and
- **polynomial space** in $|\boldsymbol{\varphi}|$ and $\boldsymbol{k \cdot \log(|T|)}$

Equivalence queries for a deterministic automaton **A** and
a hyperproperty given as a universal safety HyperLTL formula $\boldsymbol{\varphi}$ with k quantifiers
can be solved in

- **polynomial time** in $|\boldsymbol{A}|$,
- **exponential time** in $|\boldsymbol{\varphi}|$, and
- **exponential space** in $\boldsymbol{k}$.

Conclusions

- Automata provide a **canonical representation** for *k*-safety hyperproperties

- **Direct constructions** for tight permutation-complete automata
- **L* Hyper** learns minimal deterministic automata **in polynomial time** in the size of the automaton
- Learning is an **output-sensitive** approach

- **Challenge: Beyond *k*-safety**