Active Learning of Deterministic Timed Automata with Myhill-Nerode Style Characterization

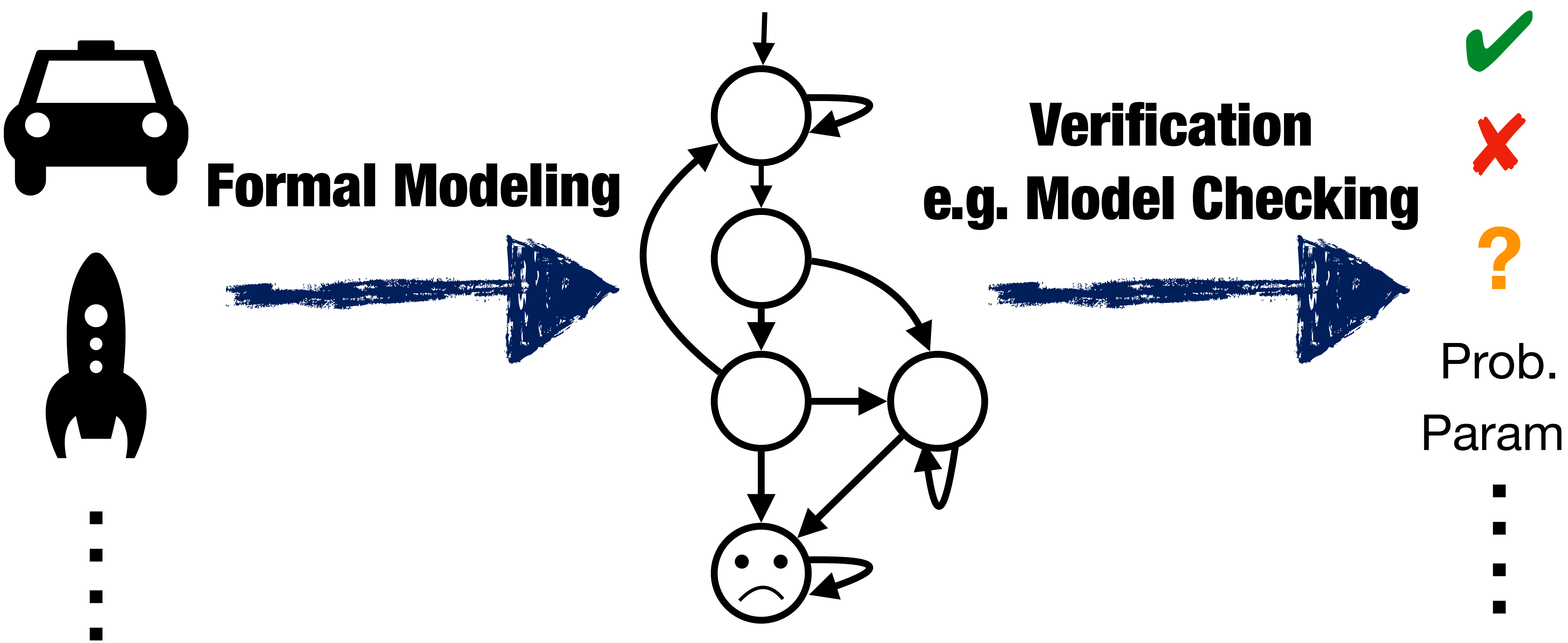
Masaki Waga

Kyoto University

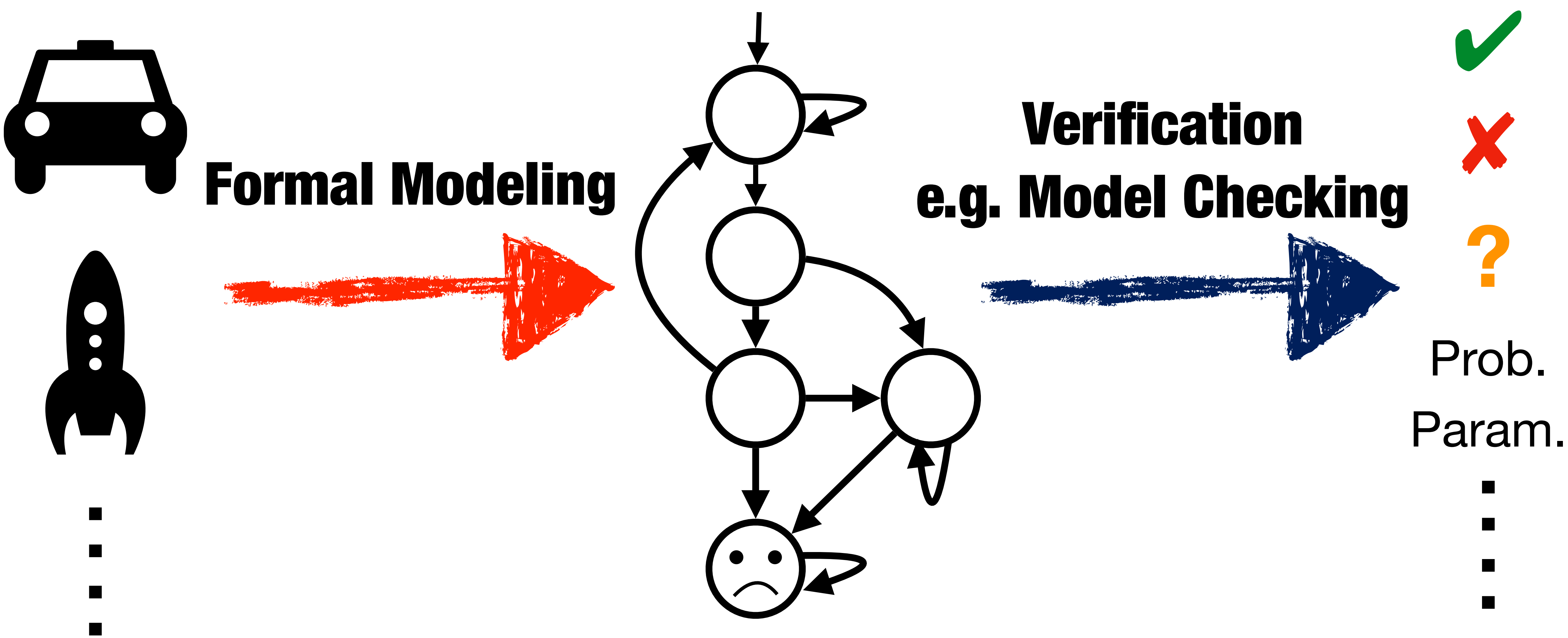
21st September 2023, FORMATS 2023
Originally presented at CAV 2023

This work is partially supported by JST ACT-X Grant No. JPMJAX200U, JST PRESTO Grant No. JPMJPR22CA, JST CREST Grant No. JPMJCR2012, and JSPS KAKENHI Grant No. 22K17873.

Formal modeling is essential

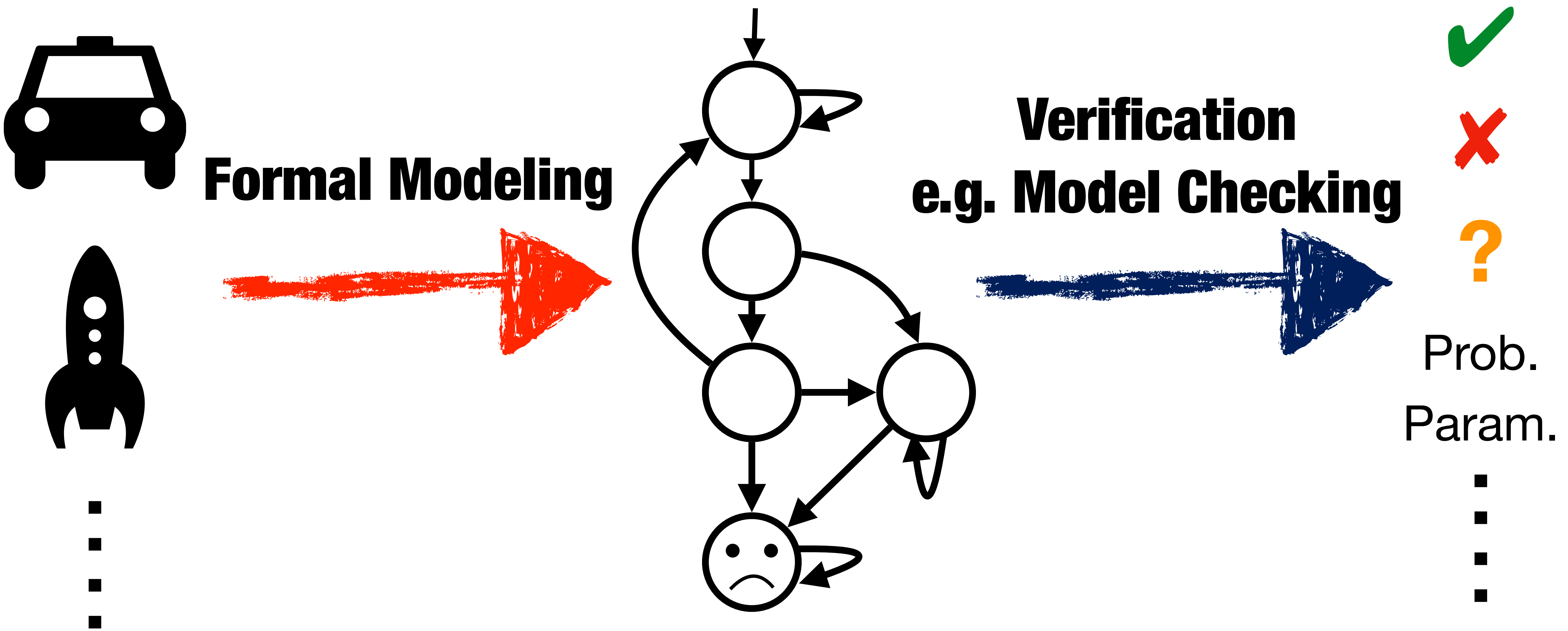


Formal modeling is essential but usually difficult



Formal modeling is essential but usually difficult

Q. How about automatically learn a formal model?



Active DFA Learning

[Angluin, Inform Comput'87]

Membership

Equivalence

Questions: $w \stackrel{?}{\in} L_{\mathrm{target}}$, $\mathscr{L}(\mathscr{A}_{\mathrm{hyp}}) \stackrel{?}{=} L_{\mathrm{target}}$

Answers: Yes, No, Evidence $w \in \mathscr{L}(\mathscr{A}_{\mathrm{hyp}}) \triangle L_{\mathrm{target}}$

Active DFA Learning

[Angluin, Inform Comput'87]

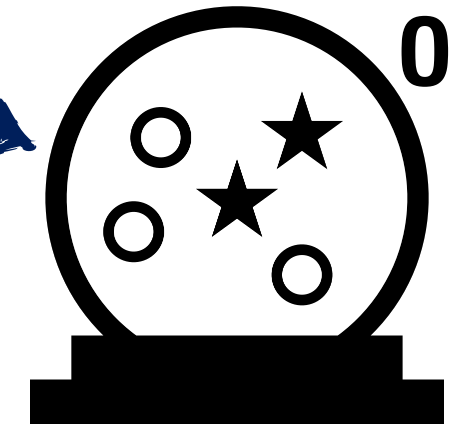
Membership

Equivalence

Questions: $w \overset{?}{\in} L_{\mathrm{target}}$, $\mathscr{L}(\mathscr{A}_{\mathrm{hyp}}) \overset{?}{=} L_{\mathrm{target}}$

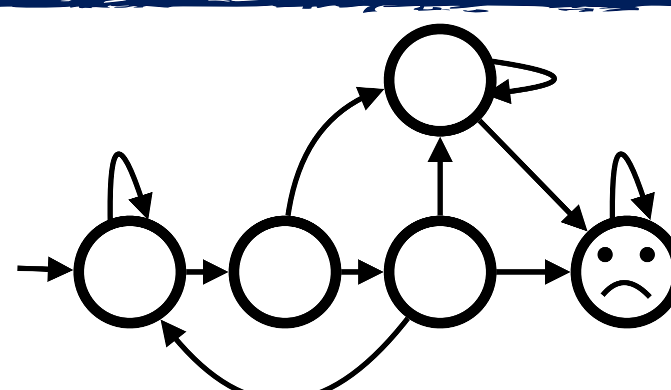
Learner

Oracle



Answers: Yes, No, Evidence $w \in \mathscr{L}(\mathscr{A}_{\mathrm{hyp}}) \bigtriangleup L_{\mathrm{target}}$

is modeled by



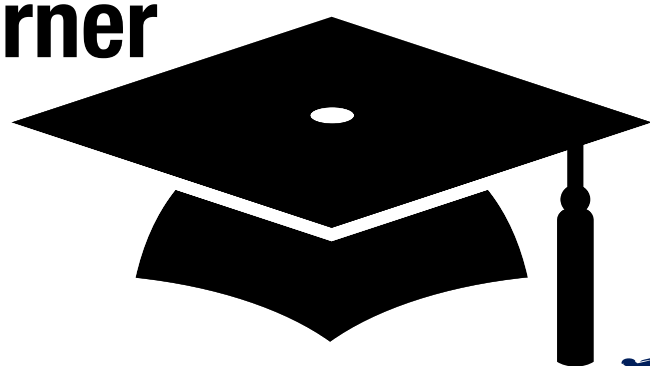
Active DFA Learning

[Angluin, Inform Comput'87]

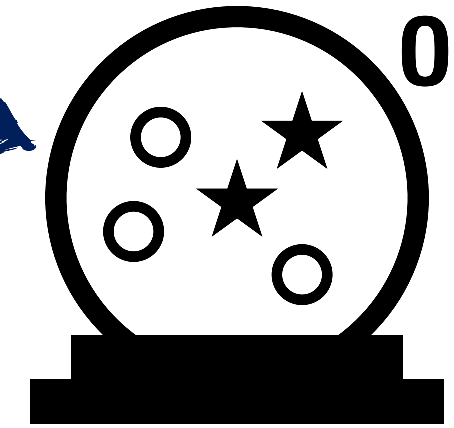
Membership **Equivalence**

Questions: $w \stackrel{?}{\in} L_{\mathrm{target}}$, $\mathscr{L}(\mathscr{A}_{\mathrm{hyp}}) \stackrel{?}{=} L_{\mathrm{target}}$

Learner

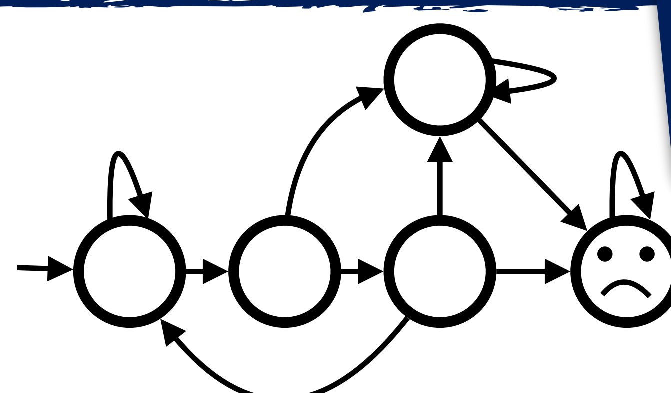


Oracle



Answers: Yes, No, Evidence $w \in \mathscr{L}(\mathscr{A}_{\mathrm{hyp}}) \triangle L_{\mathrm{target}}$

is modeled by



Correctness with finite queries

Active DFA Learning

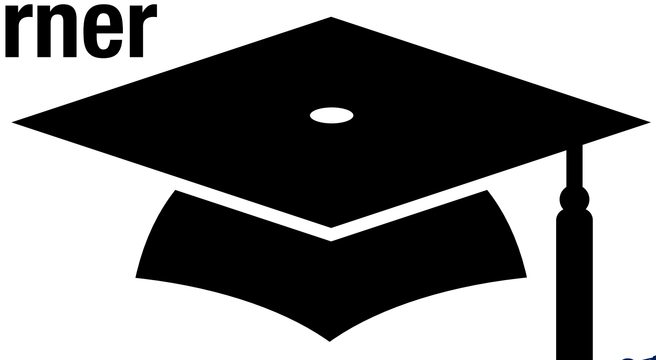
[Angluin, Inform Comput'87]

Membership

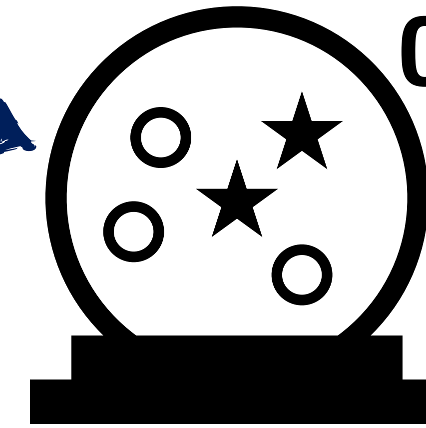
Equivalence

Questions: $w \overset{?}{\in} L_{\mathrm{target}}$, $\mathscr{L}(\mathscr{A}_{\mathrm{hyp}}) \overset{?}{=} L_{\mathrm{target}}$

Learner

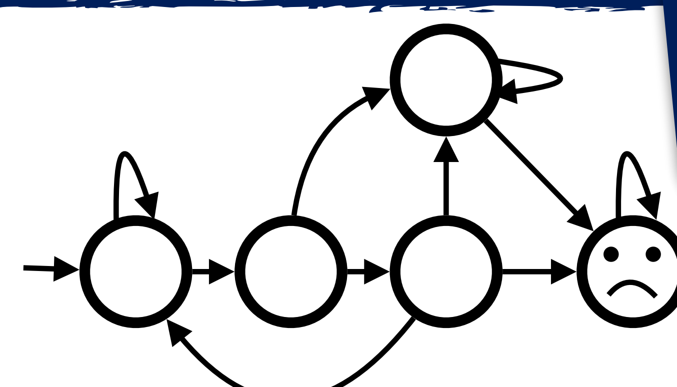


Oracle



Answers: Yes, No, Evidence $w \in \mathscr{L}(\mathscr{A}_{\mathrm{hyp}}) \triangle L_{\mathrm{target}}$

is modeled by



Correctness with finite queries

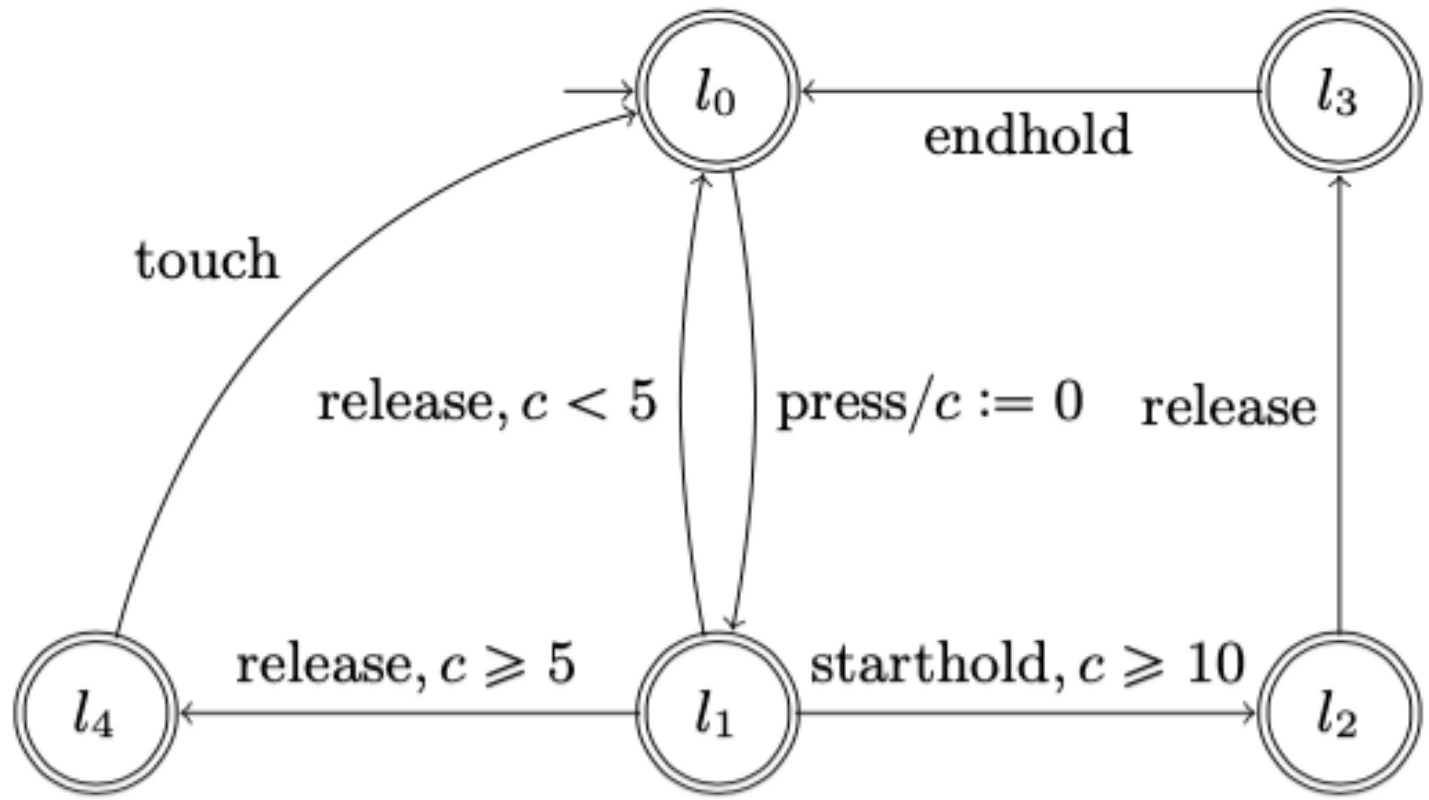
No timing info.

It's Time
to Learn Time!

Timed Automata/Words

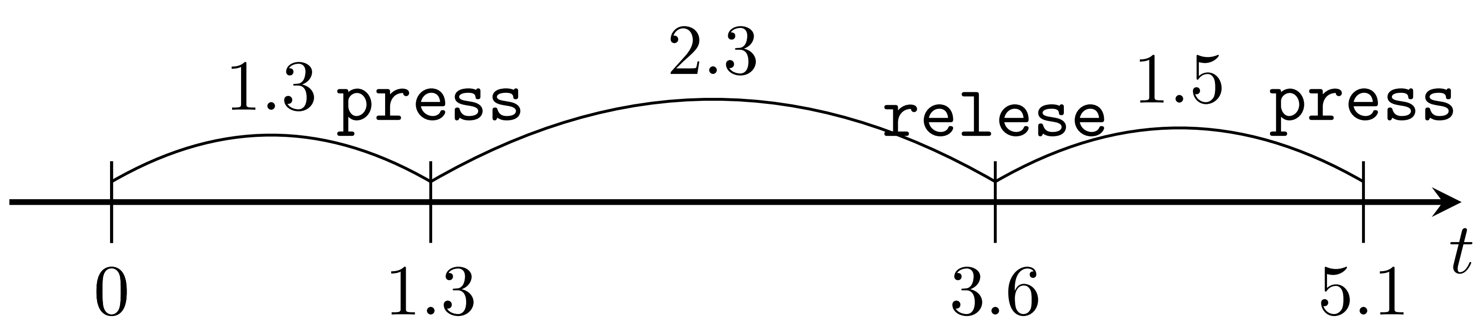
[Alur & Dill, TCS'94]

TA: NFA + timing constraints by clocks, guards, resets



Timed word: Sequence of events + dwell time

- e.g. `1.3 press 2.3 release 1.5 press`



Active Deterministic TA Learning

[Contribution]

Timed word: Sequence of events + dwell time
e.g. `1.3 press 2.3 release 1.5 press`

Deterministic TA

Questions: $w \overset{?}{\in} L_{\mathrm{target}}$, $\mathscr{L}(\mathscr{A}_{\mathrm{hyp}}) \overset{?}{=} L_{\mathrm{target}}$

Answers: Yes, No, Evidence $w \in \mathscr{L}(\mathscr{A}_{\mathrm{hyp}}) \triangle L_{\mathrm{target}}$

Active Deterministic TA Learning

[Contribution]

Timed word: Sequence of events + dwell time
e.g. 1.3 press 2.3 release 1.5 press

Deterministic TA

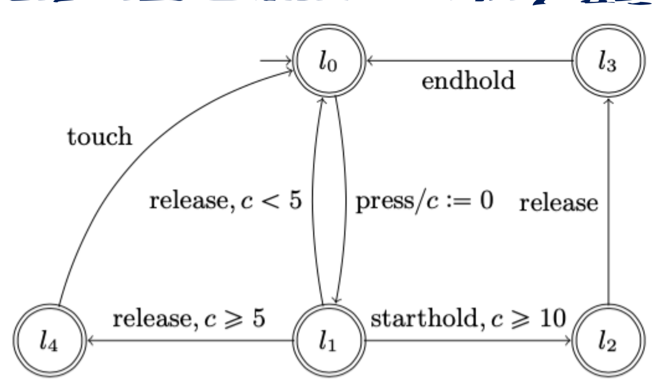
Questions: $w \overset{?}{\in} L_{\mathrm{target}}$, $\mathscr{L}(\mathscr{A}_{\mathrm{hyp}}) \overset{?}{=} L_{\mathrm{target}}$

Learner

Oracle

Answers: Yes, No, Evidence $w \in \mathscr{L}(\mathscr{A}_{\mathrm{hyp}}) \triangle L_{\mathrm{target}}$

is modeled by



Active Deterministic TA Learning

[Contribution]

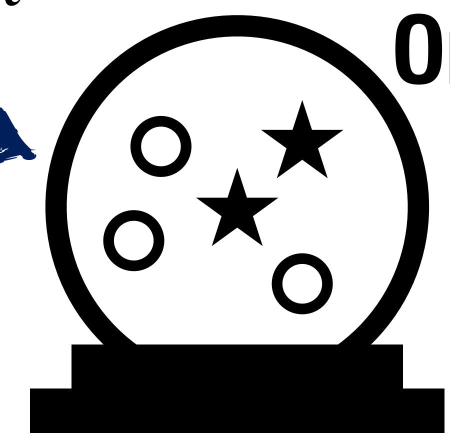
Timed word: Sequence of events + dwell time
e.g. 1.3 press 2.3 release 1.5 press

Deterministic TA

Questions: $w \overset{?}{\in} L_{\mathrm{target}}$, $\mathscr{L}(\mathscr{A}_{\mathrm{hyp}}) \overset{?}{=} L_{\mathrm{target}}$

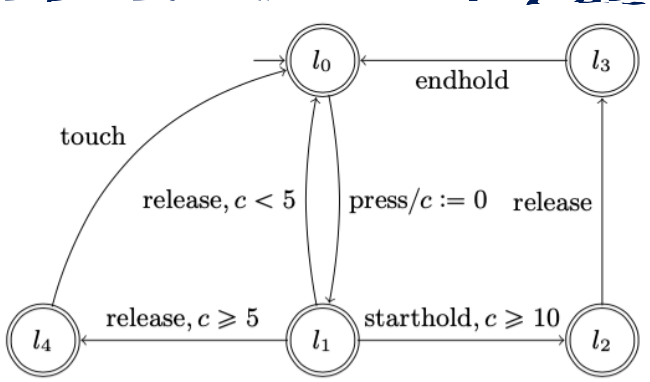
Learner

Oracle



Answers: Yes, No, Evidence $w \in \mathscr{L}(\mathscr{A}_{\mathrm{hyp}}) \triangle L_{\mathrm{target}}$

is modeled by



Correctness with finite queries

Active Deterministic TA Learning

[Contribution]

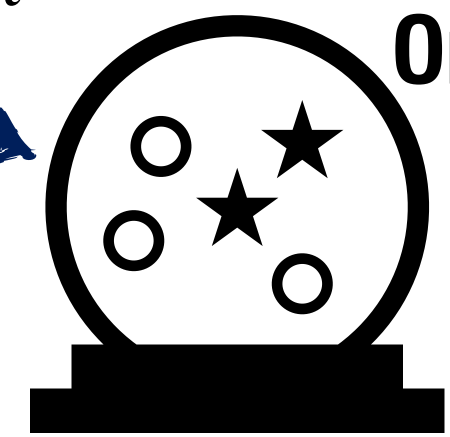
Timed word: Sequence of events + dwell time
e.g. 1.3 press 2.3 release 1.5 press

Deterministic TA

Questions: $w \stackrel{?}{\in} L_{\mathrm{target}}$, $\mathscr{L}(\mathscr{A}_{\mathrm{hyp}}) \stackrel{?}{=} L_{\mathrm{target}}$

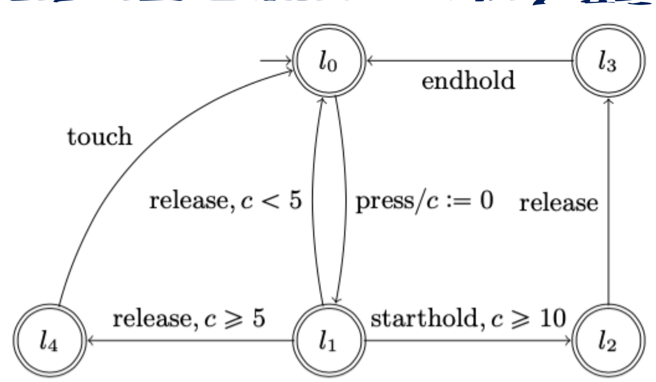
Learner

Oracle



Answers: Yes, No, Evidence $w \in \mathscr{L}(\mathscr{A}_{\mathrm{hyp}}) \triangle L_{\mathrm{target}}$

is modeled by



Correctness with finite queries

With timing info.

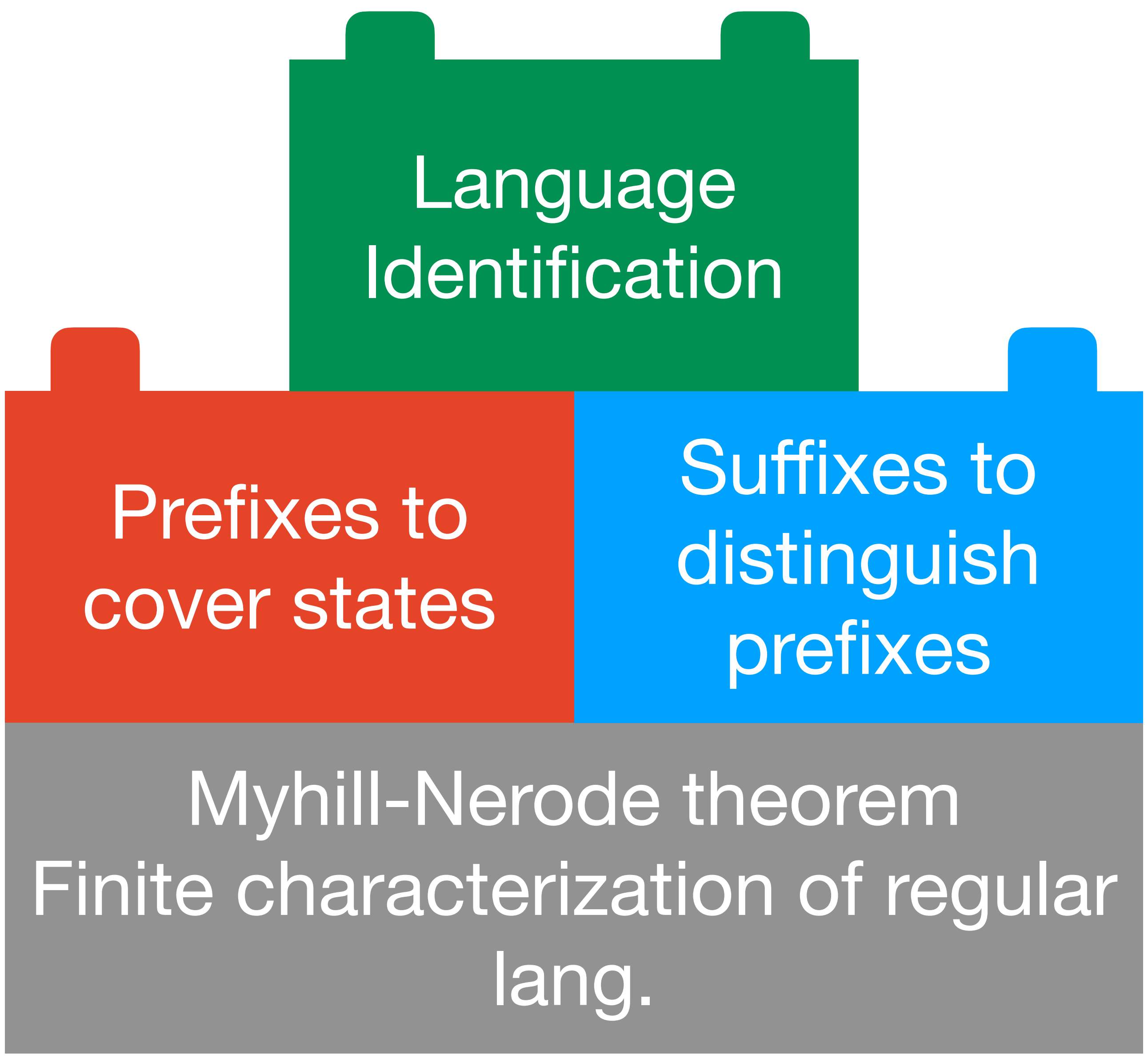
Slogan: Learn Autom. via Finiteness

L* algorithm for DFAs

[Angluin, Inform Comput'87]

L*$_{\text{Timed}}$ algorithm for DTAs

[Contribution]



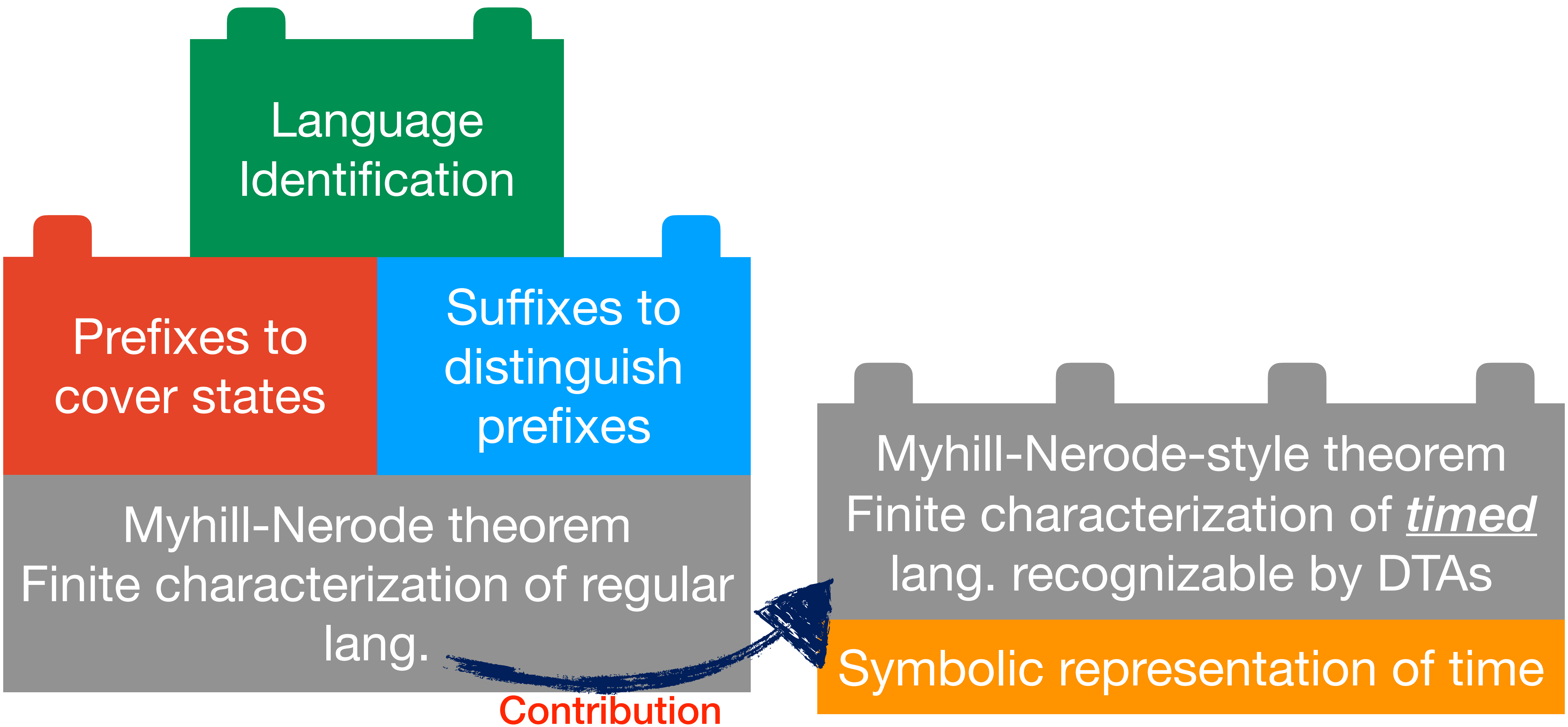
Slogan: Learn Autom. via Finiteness

L* algorithm for DFAs

[Angluin, Inform Comput'87]

L*$_{\text{Timed}}$ algorithm for DTAs

[Contribution]



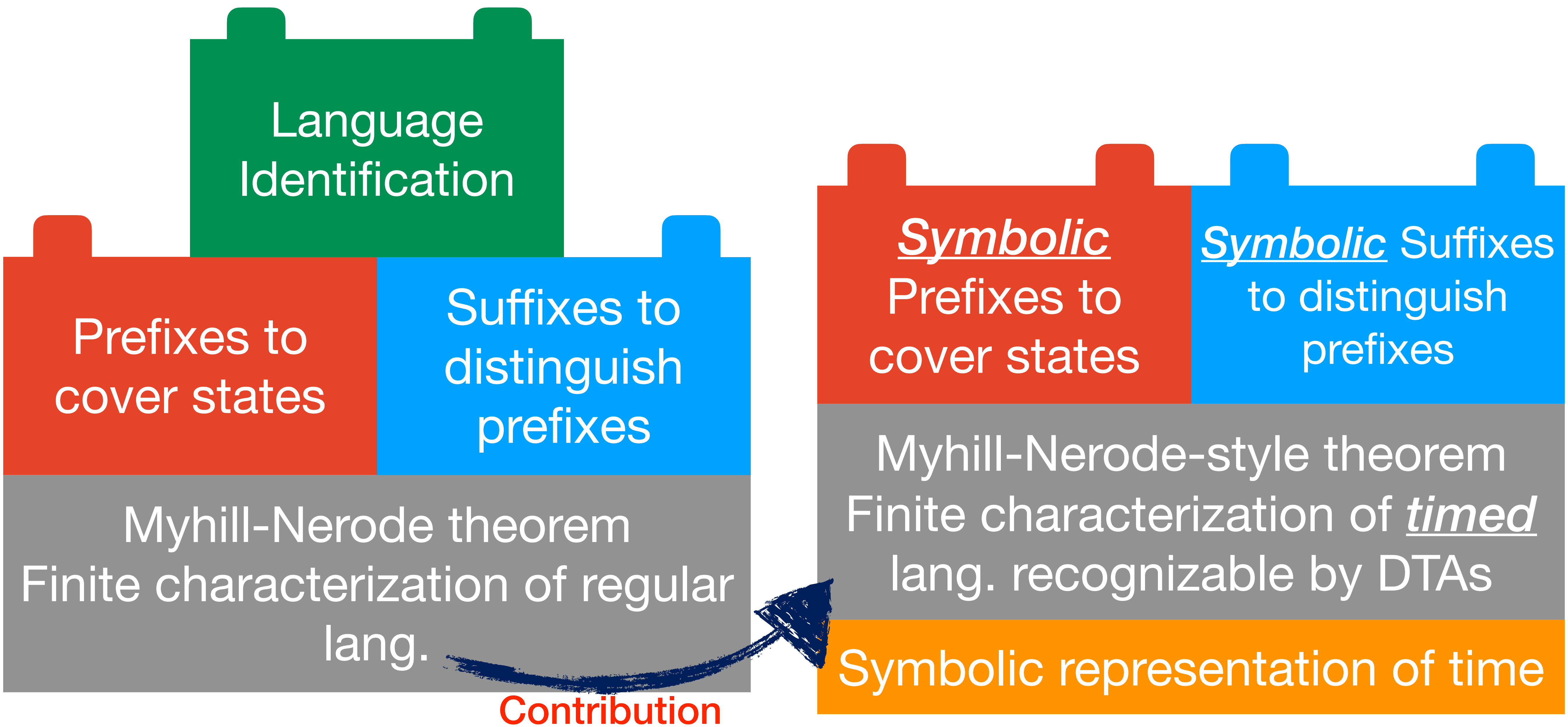
Slogan: Learn Autom. via Finiteness

L* algorithm for DFAs

[Angluin, Inform Comput'87]

L*$_{\text{Timed}}$ algorithm for DTAs

[Contribution]



Slogan: Learn Autom. via Finiteness

L* algorithm for DFAs

[Angluin, Inform Comput'87]

Language Identification

Prefixes to cover states

Suffixes to distinguish prefixes

Myhill-Nerode theorem
Finite characterization of regular lang.

Contribution

L*$_{\text{Timed}}$ algorithm for DTAs

[Contribution]

Timed Language Identification

Symbolic Prefixes to cover states

Symbolic Suffixes to distinguish prefixes

Myhill-Nerode-style theorem
Finite characterization of *timed* lang. recognizable by DTAs

Symbolic representation of time

NOT the First Time to Learn Time!

Another Myhill-Nerode-style characterization also exists (w/ nominal sets)
[Bojańczyk& Lasota, ICALP'12]

Non-exact methods also exist
e.g. [Aichernig et al., NFM'20] with GA

Exact Learning of TAs (w/ correctness guarantee)

For (***multi-clock***) DTAs
[This work]

Driven by Language Characterization (inspired by [Maler & Pnueli, FoSSaCS'04])

more general

Clock reset depends on the transition label

For *one-clock* DTAs
e.g. [Xu et al., ATVA '22]

For Event-recording automata
e.g. [Grinchtein et al., TCS '10]
[Henry et al., FORMATS '20]

For real-time automata
e.g. [An et al., Sci. China Inf. Sci. '20]

One clock + reset at each transition

Contributions

- Myhill-Nerode-style characterization to the timed languages recognizable by DTAs
  - Idea: symbolic handling of timing constraints
- L*-style learning algorithm for DTAs
- Implementation + experiments
  → Works for some practical benchmarks, e.g., FDDI

Outline

- Quick Review: L* algorithm for DFA learning
  - Focusing on Myhill-Nerode theorem & Nerode's congruence
- Active learning of timed lang. recognizable by DTAs
  - Myhill-Nerode-style characterization for timed lang.
  - How the algorithm is extended
- Experiments

Nerode's congruence

Nerode's congruence ($\equiv_L$): For

- language: $L \subseteq \Sigma^*$
- prefixes: $p, p' \in \Sigma^*$

$p \equiv_L p'$ iff. $\forall s \in \Sigma^* . p \cdot s \in L \iff p' \cdot s \in L$

e.g. for $L = (\mathrm{ab})^*$, $\varepsilon \equiv_L \mathrm{ab}$, $\mathrm{a} \equiv_L \mathrm{aba}$, …

Accepted iff $s \in (\mathrm{ab})^*$

Accepted iff $s \in \mathrm{b}(\mathrm{ab})^*$

✔: Easy to construct $\mathscr{A}$ s.t. $\mathscr{L}(\mathscr{A}) = L$ from $\equiv_L$

- **Idea**: Use $\Sigma^* / \equiv_L$ as the state space

✘: Requires *infinite* comparison to check if $p \equiv_L p'$

Learn. Lang. via approx. congruence

Idea: approx. $\equiv_L$ by $\approx_L^S$ with finite suffixes $S \subseteq \Sigma^*$

e.g. $\varepsilon \approx_L^{\{\varepsilon,\mathrm{b}\}} \mathrm{ab}$ iff $\forall s \in \{\varepsilon, \mathrm{b}\}\ \varepsilon \cdot s \in L \iff \mathrm{ab} \cdot s \in L$

✔: Requires *finite* comparison to check if $p \approx_L^S p'$

✔: Still, easy to construct $\mathscr{A}$ from $\approx_L^S$

- **Idea**: Use $P/\approx_L^S$ as the state space for prefixes $P \subseteq \Sigma^*$.

✔: Finite S is enough for regular lang.

- By Myhill-Nerode theorem ($\Sigma^*/\equiv_L$ is finite if L is regular)

Learn. Lang. via approx. congruence

Idea: approx. $\equiv_L$ by $\approx_L^S$ with finite suffixes $S \subseteq \Sigma^*$

e.g. $\varepsilon \approx_L^{\{\varepsilon,\mathrm{b}\}} \mathrm{ab}$ iff $\forall s \in \{\varepsilon, \mathrm{b}\}\ \varepsilon \cdot s \in L \iff \mathrm{ab} \cdot s \in L$

✔: Requires *finite* comparison to check if $p \approx_L^S p'$

✔: Still, easy to construct $\mathscr{A}$ from $\approx_L^S$

- **Idea**: Use $P/\approx_L^S$ as the state space for prefixes $P \subseteq \Sigma^*$.

✔: Finite S is enough for regular lang.

- By Myhill-Nerode theorem ($\Sigma^*/\equiv_L$ is finite if L is regular)

Prefixes to cover states

Suffixes to distinguish prefixes

L* algorithm for Active DFA Learning

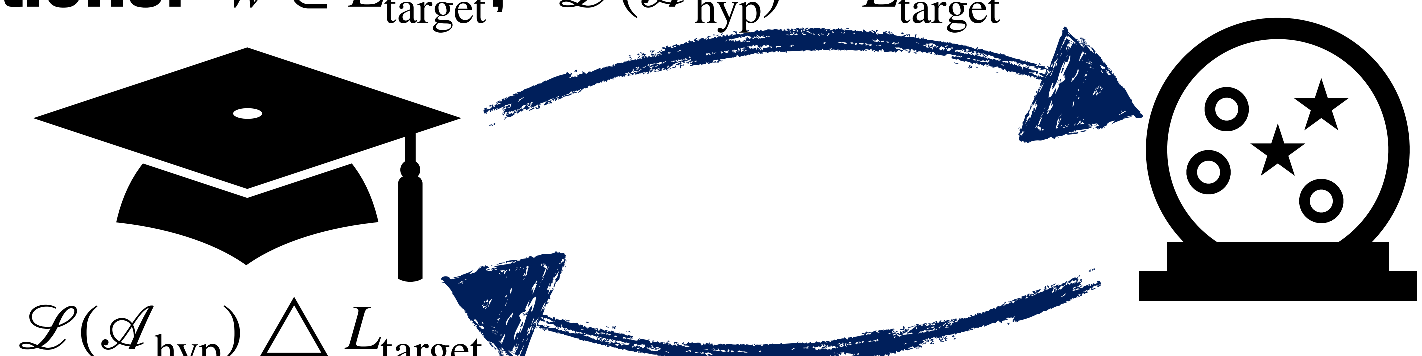
[Angluin, Inform Comput'87]

1. Systematically refine prefixes P and suffixes S
   - e.g. Refine if contradiction is found
2. Test equiv. of $p, p' \in P$ wrt S by mem. questions
3. Make $\mathscr{A}_{\text{hyp}}$ if $P/ \approx^{S}_{L_{\text{target}}}$ is constructed w/o contradiction
   → Use equiv. question to test the correctness
4. Evidence of $\mathscr{L}(\mathscr{A}_{\text{hyp}}) \neq L_{\text{target}}$ is used to refine P
   → Back to 1.

Membership

Equivalence

Questions: $w \overset{?}{\in} L_{\text{target}}$, $\mathscr{L}(\mathscr{A}_{\text{hyp}}) \overset{?}{=} L_{\text{target}}$



Answers: Yes, No, Evidence $w \in \mathscr{L}(\mathscr{A}_{\text{hyp}}) \triangle L_{\text{target}}$

Outline

- Quick Review: L* algorithm for DFA learning
  - Focusing on Myhill-Nerode theorem & Nerode's congruence
- Active learning of timed lang. recognizable by DTAs
  - Myhill-Nerode-style characterization for timed lang.
  - How the algorithm is extended
- Experiments

Focusing on this characterization

Challenge 1: Time is continuous!

Observation: Nerode's congruence on timed words *does not work* for learning

Example: For $L = \{\mathrm{a}\ 1\ \mathrm{b}\}$, the following must be distinguished

- a 0.1
- a 0.11
- a 0.111
- …

Accepted iff s = 0.9 b

Accepted iff s = 0.89 b

Accepted iff s = 0.889 b

S must be *Infinite*!

Gadget 1: Elementary Languages

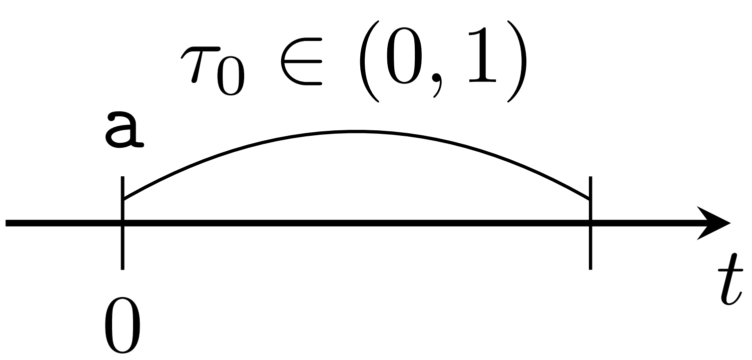
[Maler & Pnueri, FoSSaCS'04]

Use symbolic representation of time

Elementary language: Timed language defined by

- word
- Intervals on the durations between events

Example:



includes

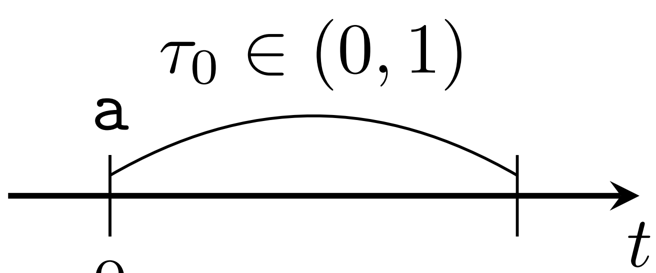
- a 0.1
- a 0.11
- a 0.111
- …

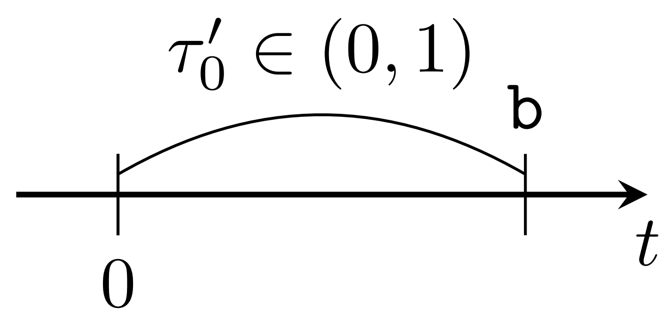
Challenge 2: Membership is inconsistent

The notion of "membership" must be updated!!

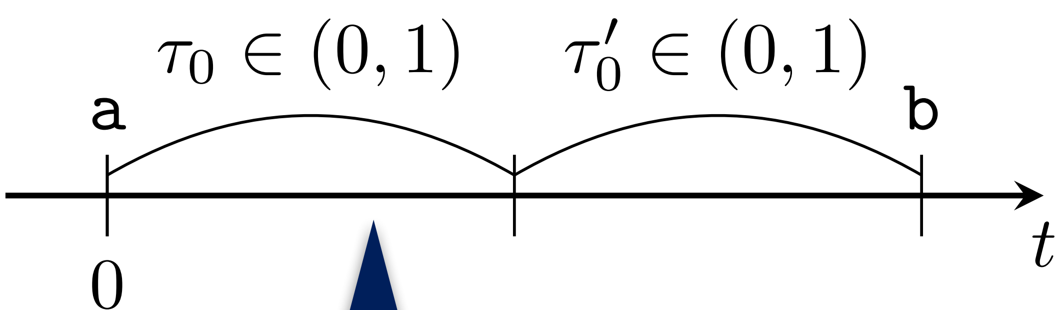
Observation:

We may not have $p \cdot s \subseteq L$ nor $p \cdot s \cap L = \emptyset$

Example: $p =$ 

, $s =$ 

,

$p \cdot s =$ 

, and $L = \{\mathtt{a}\ 1\ \mathtt{b}\}$,

In L if $\tau_0 + \tau'_0 = 1$

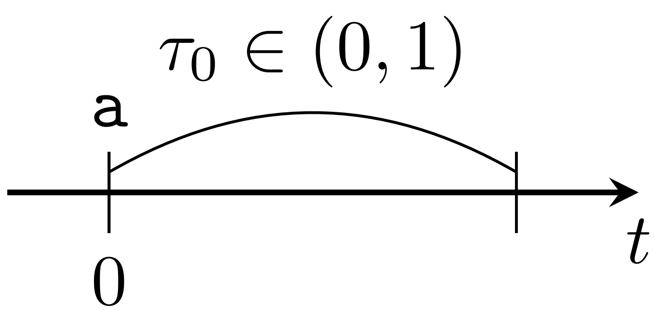
Gadget 2: Symbolic Membership

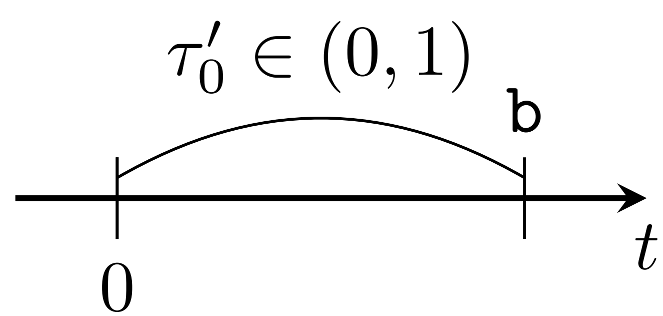
[Contribution]

Use the condition to be included

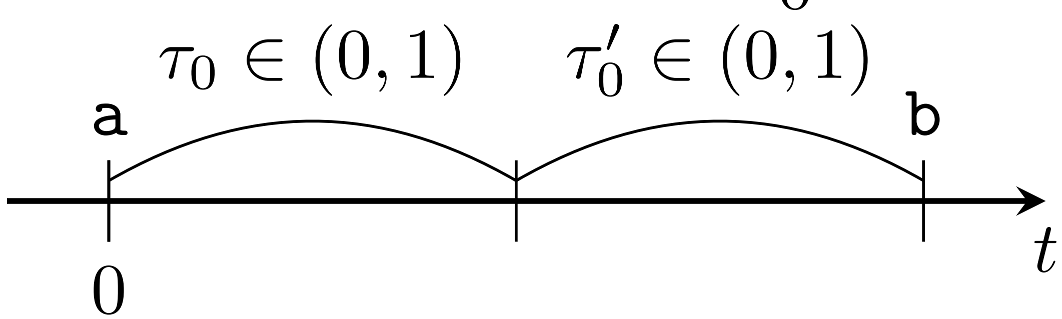
Def. For timed lang. L, L', $\mathtt{mem}_L^{\mathtt{sym}}(L')$ is the strongest constraint s.t. $\forall w \in L'.\, w \in L \iff w \models \mathtt{mem}_L^{\mathtt{sym}}(L')$

Example: For $L = \{\mathtt{a}\, 1\, \mathtt{b}\}$,

$p =$ 

, $s =$ 

,

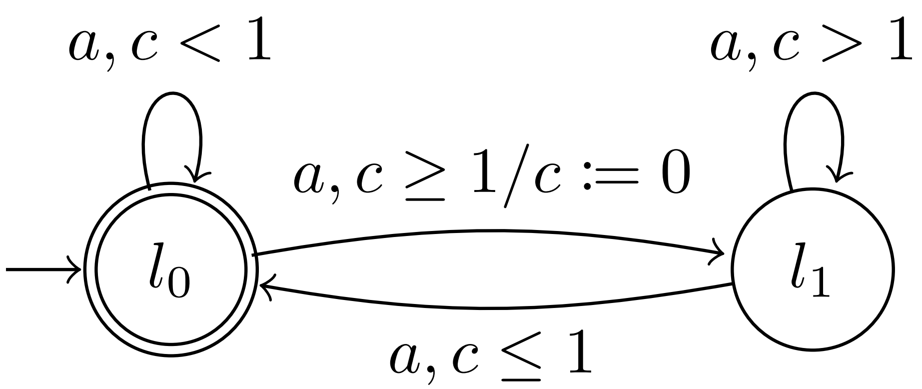
$p \cdot s =$ 

$\mathtt{mem}_L^{\mathtt{sym}}(p \cdot s)$ is $\tau_0 + \tau'_0 = 1$

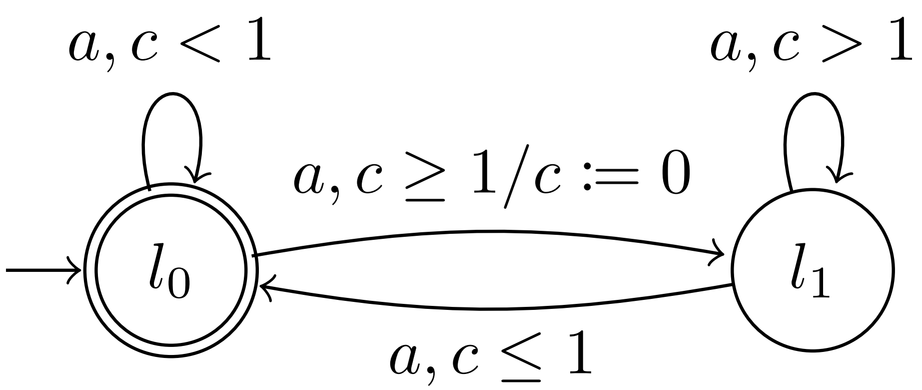
Challenge 3: Direct comparison is too fine

Clock valuations:

w = 0.5 a



w' = 0.3 a 0.2 a

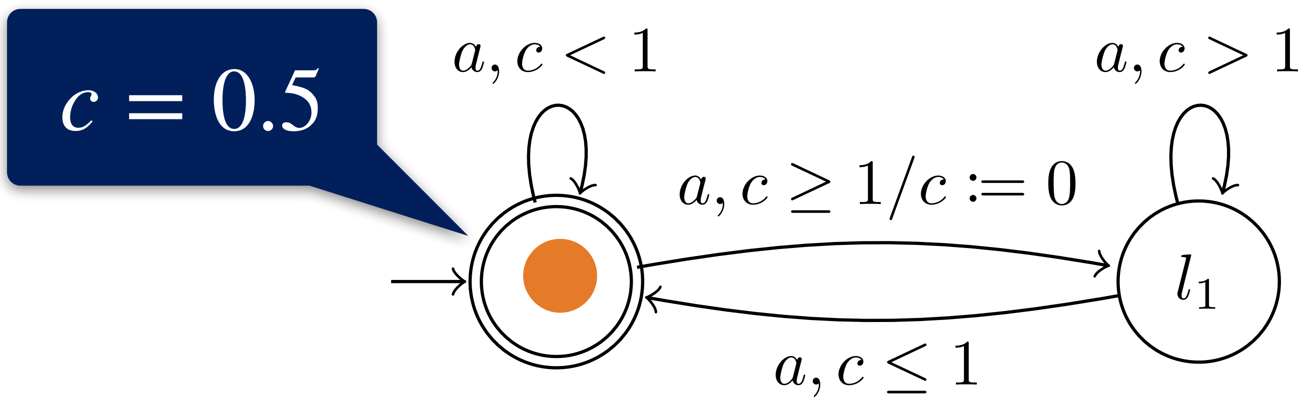


Symbolic membership:

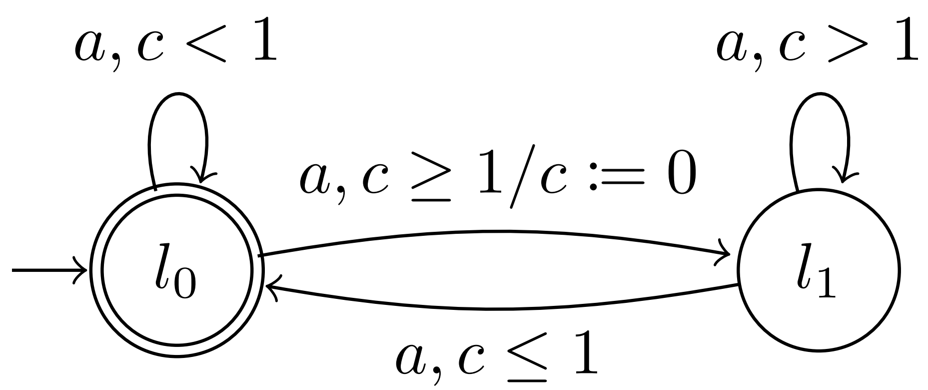
Challenge 3: Direct comparison is too fine

Clock valuations:

w = 0.5 a



w' = 0.3 a 0.2 a

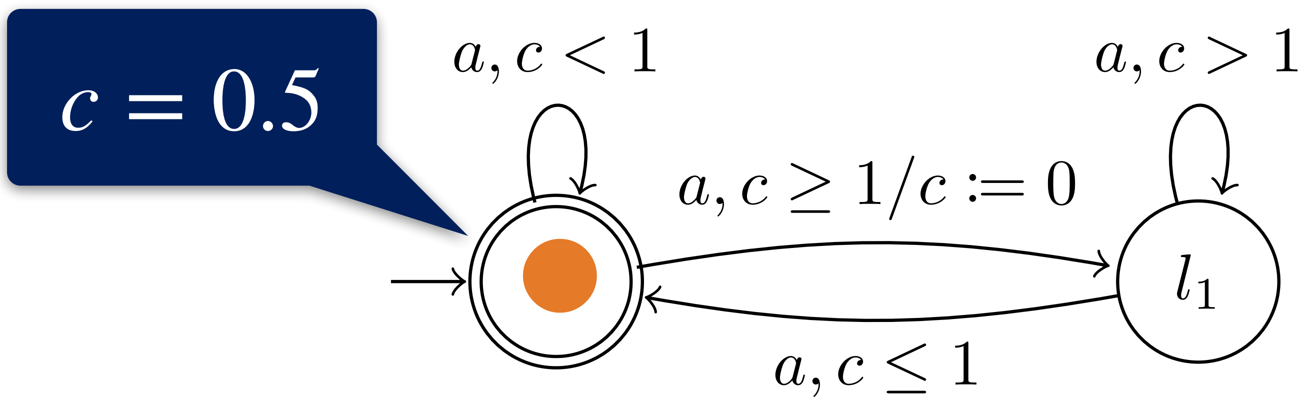


Symbolic membership:

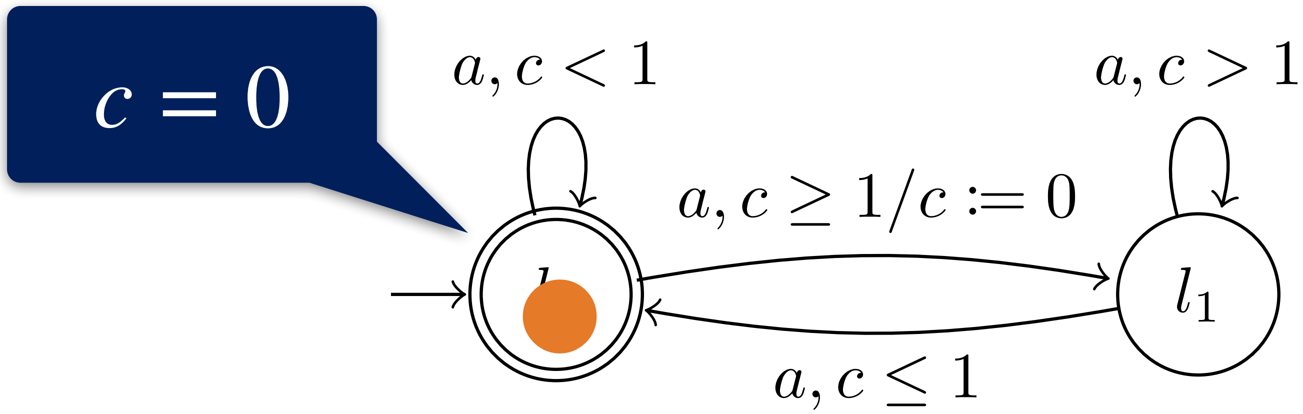
Challenge 3: Direct comparison is too fine

Clock valuations:

w = 0.5 a



w' = 0.3 a 0.2 a

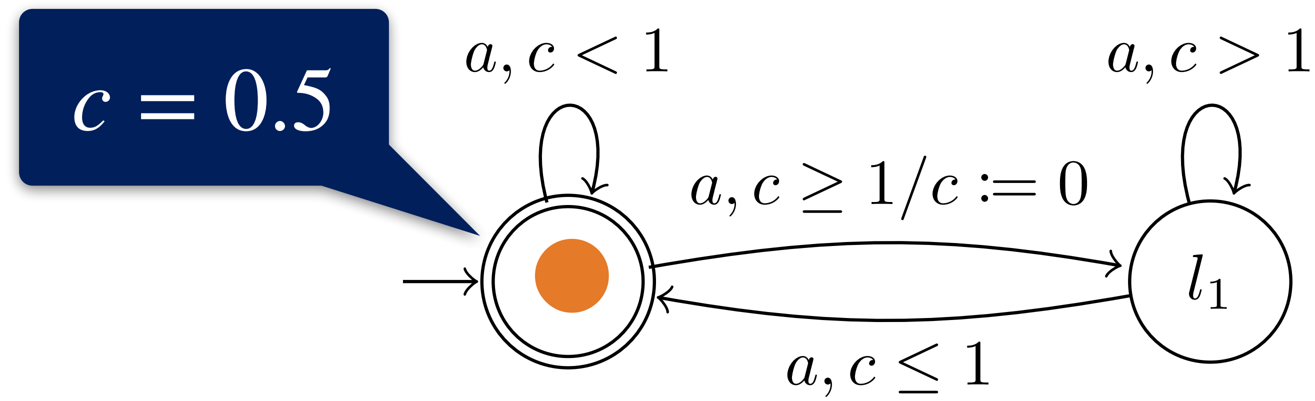


Symbolic membership:

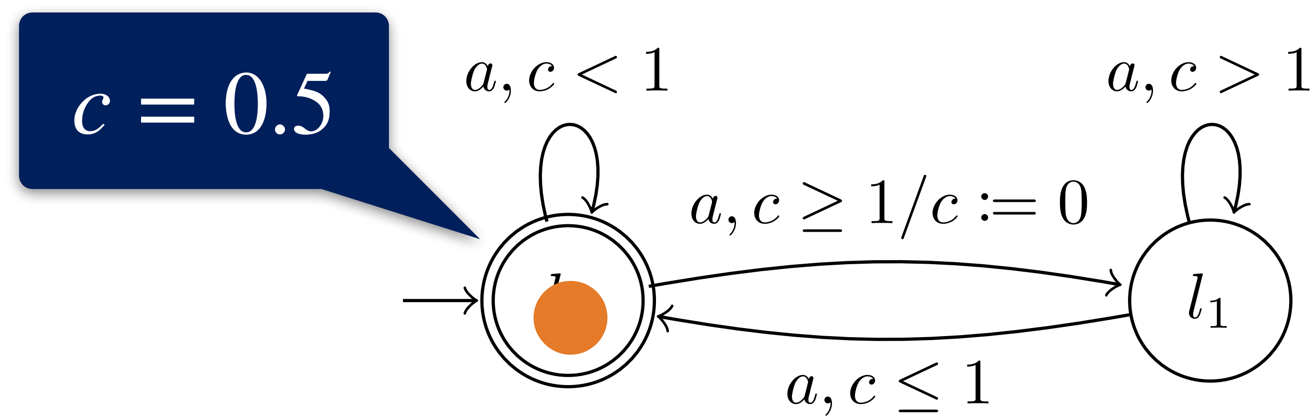
Challenge 3: Direct comparison is too fine

Clock valuations:

w = 0.5 a



w' = 0.3 a 0.2 a



Symbolic membership:

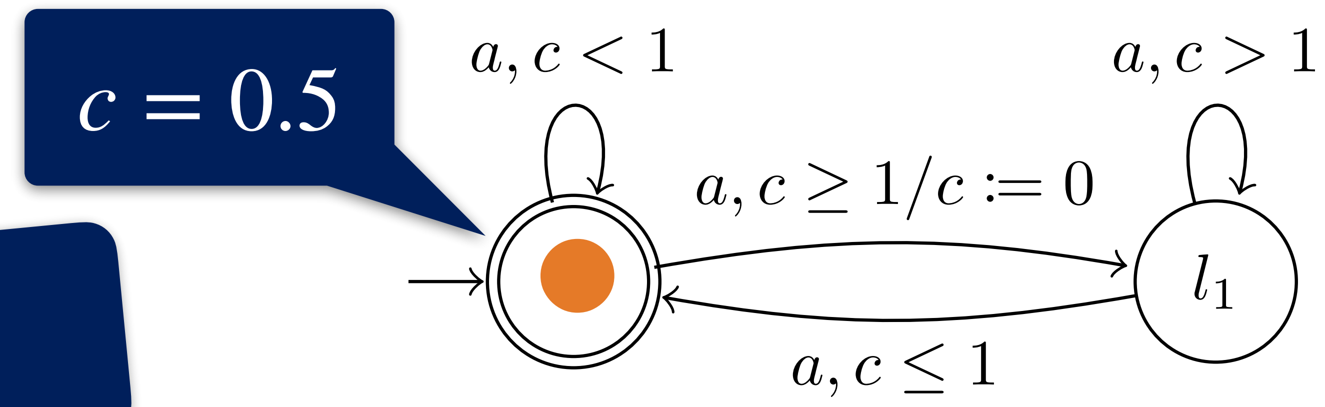
Challenge 3: Direct comparison is too fine

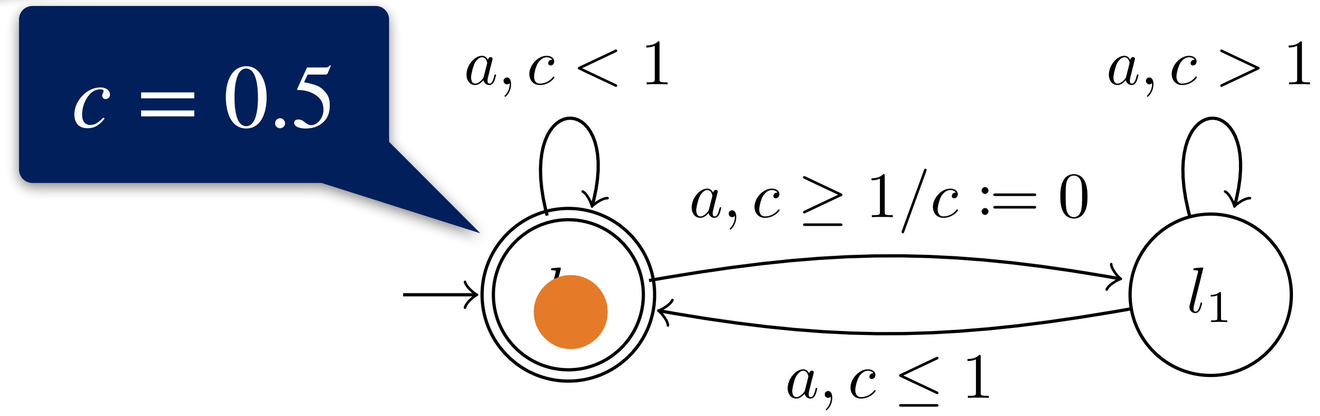
Clock valuations:

w = 0.5 a

Equivalent!

w' = 0.3 a 0.2 a





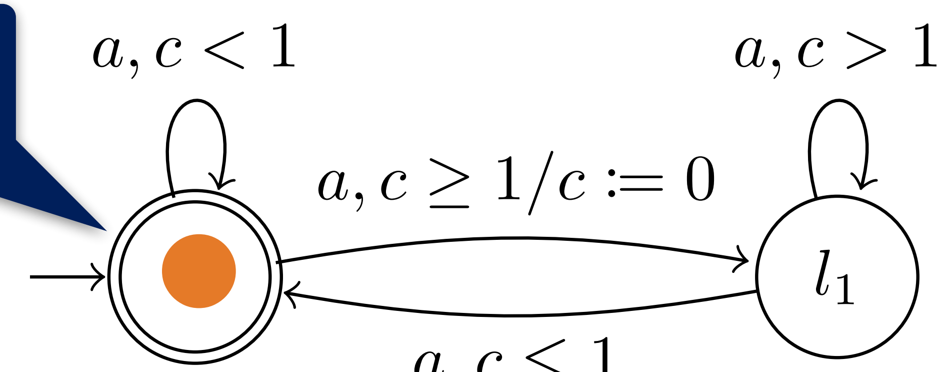
Symbolic membership:

Challenge 3: Direct comparison is too fine

Clock valuations:

w = 0.5 a

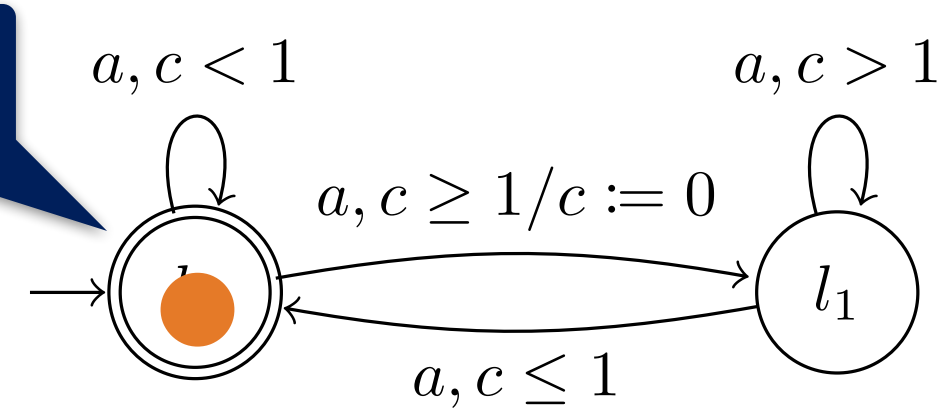
$c = 0.5$



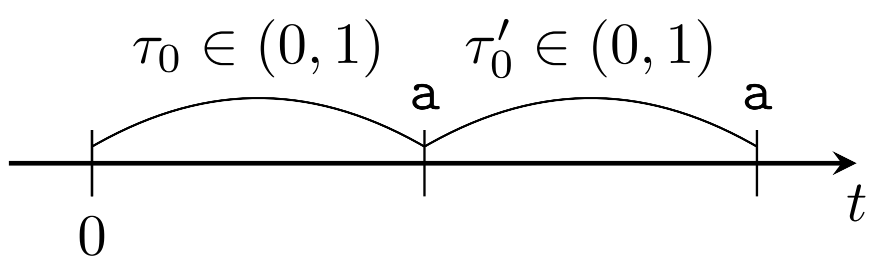
Equivalent!

w' = 0.3 a 0.2 a

$c = 0.5$



Symbolic membership:

For $p \cdot s =$ 

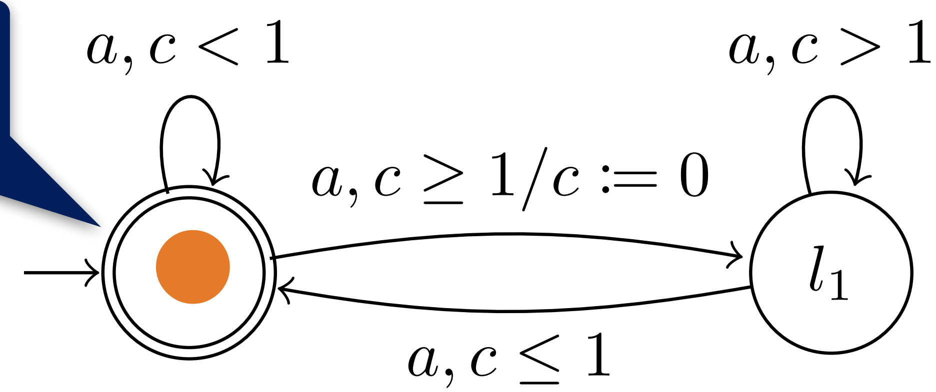
$\mathtt{mem}_L^{\mathtt{sym}}(p \cdot s)$ is $\tau_0 + \tau'_0 < 1$

Challenge 3: Direct comparison is too fine

Clock valuations:

w = 0.5 a

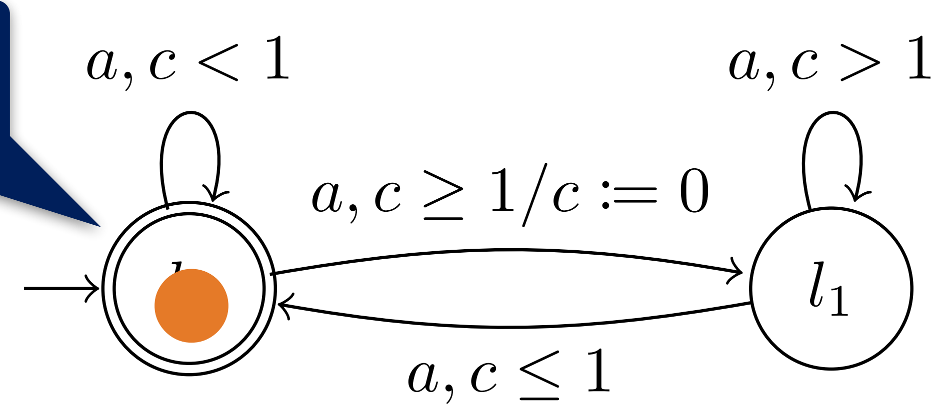
$c = 0.5$



Equivalent!

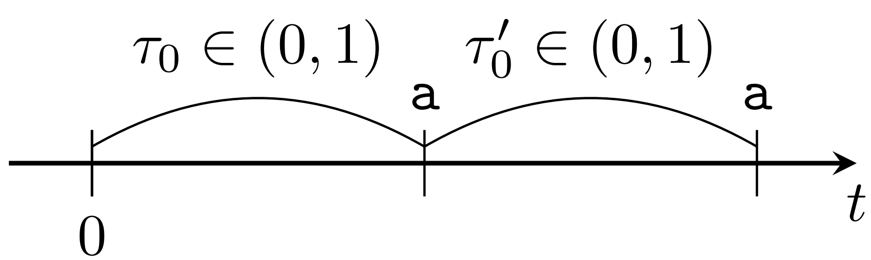
w' = 0.3 a 0.2 a

$c = 0.5$



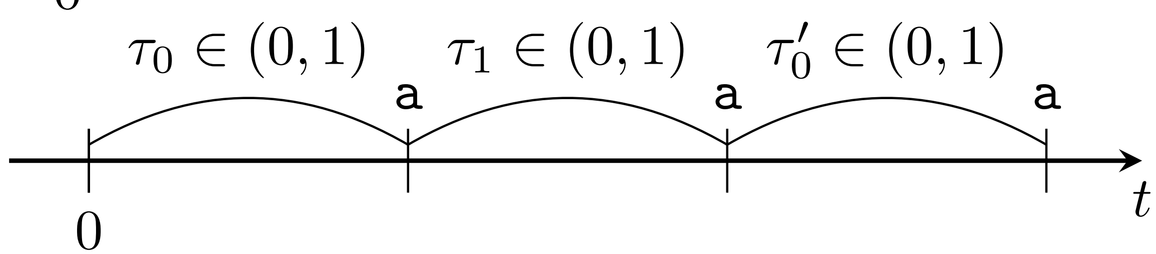
Symbolic membership:

For $p \cdot s =$



$\mathtt{mem}_L^{\mathtt{sym}}(p \cdot s)$ is $\tau_0 + \tau'_0 < 1$

For $p' \cdot s =$



$\mathtt{mem}_L^{\mathtt{sym}}(p \cdot s)$ is $\tau_0 + \tau_1 + \tau'_0 < 1$

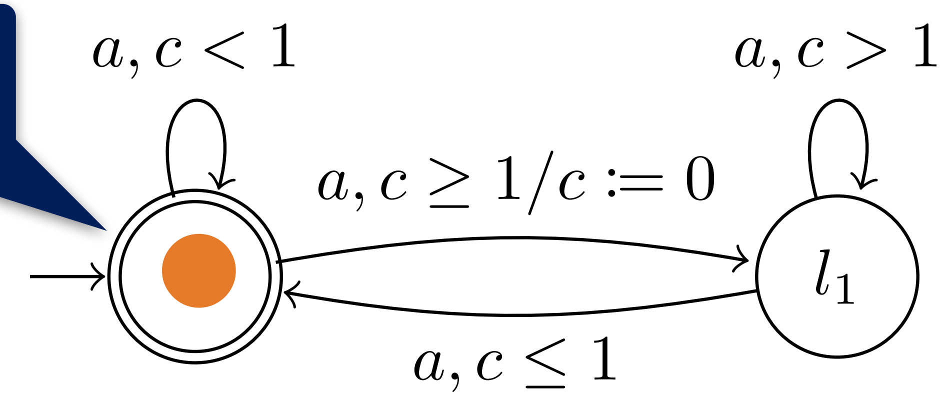
Challenge 3: Direct comparison is too fine

Clock valuations:

$w = 0.5\ \mathtt{a}$

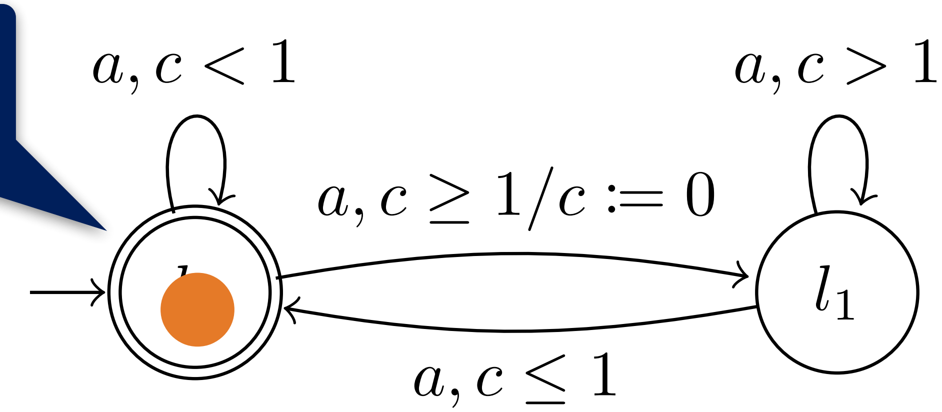
Equivalent!

$c = 0.5$



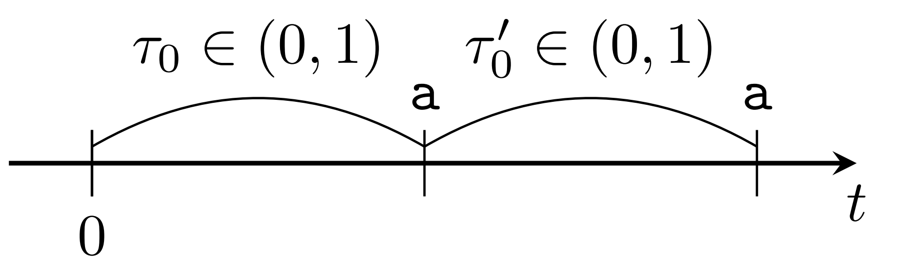
$w' = 0.3\ \mathtt{a}\ 0.2\ \mathtt{a}$

$c = 0.5$



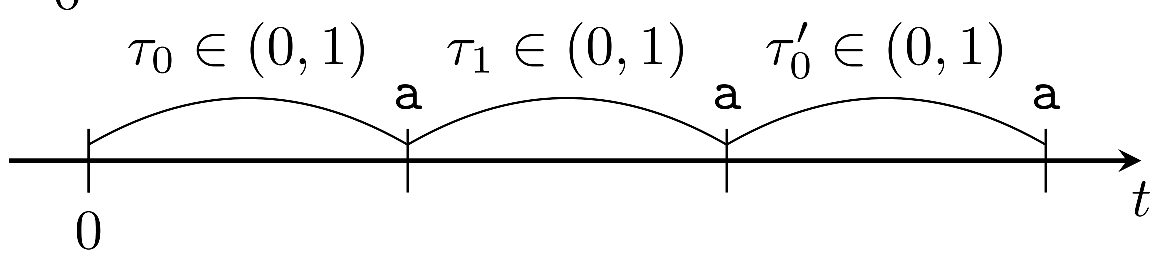
Symbolic membership:

For $p \cdot s =$



$\mathtt{mem}_L^{\mathtt{sym}}(p \cdot s)$ is $\tau_0 + \tau_0' < 1$

For $p' \cdot s =$



Different!

$\mathtt{mem}_L^{\mathtt{sym}}(p \cdot s)$ is $\tau_0 + \tau_1 + \tau_0' < 1$

Gadget 3: Equivalence up to Renaming

[Contribution]

Abstract the reset "position"

Def. For

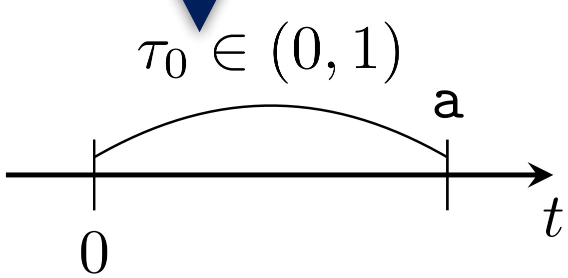
- timed language: L
- (prefix) elementary lang.: p, p'

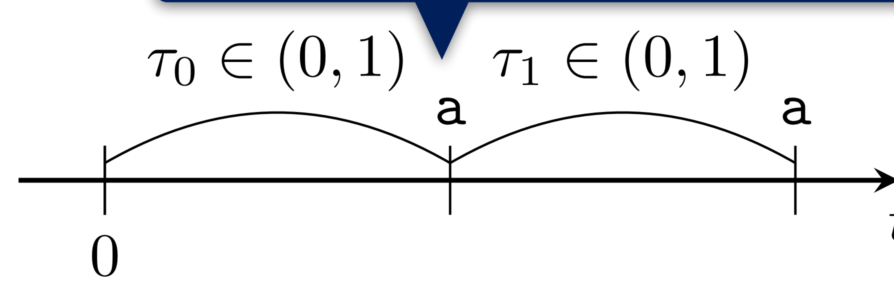
$p \equiv_L^{\mathrm{timed}} p'$ iff. for any (suffix) elementary lang. s, $\mathtt{mem}_L^{\mathtt{sym}}(p \cdot s)$ and $\mathtt{mem}_L^{\mathtt{sym}}(p' \cdot s)$ are equivalent up to some renaming

Example:

$\mathtt{mem}_L^{\mathtt{sym}}(p \cdot s)$ is $\tau_0 + \tau'_0 < 1$

$\mathtt{mem}_L^{\mathtt{sym}}(p \cdot s)$ is $\tau_0 + \tau_1 + \tau'_0 < 1$

For $p =$ 

and $p' =$ 

,

we have $p \equiv_L^{\mathrm{timed}} p'$ with renaming $\tau_0 \leftrightarrows \tau_0 + \tau_1$

Myhill-Nerode-Style Characterization

[Contribution]

Theorem

$\equiv_L^{\mathrm{timed}}$ makes ***finite*** classes iff. L is recognizable by a DTA

Corollary The above classes are distinguishable by a ***finite*** set S of elementary lang.

→ We can learn a DTA via incremental construction of S

M. Waga (Kyoto U.)

L^*_{timed} algorithm for Active DTA Learning

Outline is same as the L* for DFAs

[Contribution]

Diff. 1: Use symbolic membership questions

- Achieved by finitely many membership questions

Diff. 2: Search renaming to test equivalence

- Exhaustive trial w/o additional questions (by memoization)

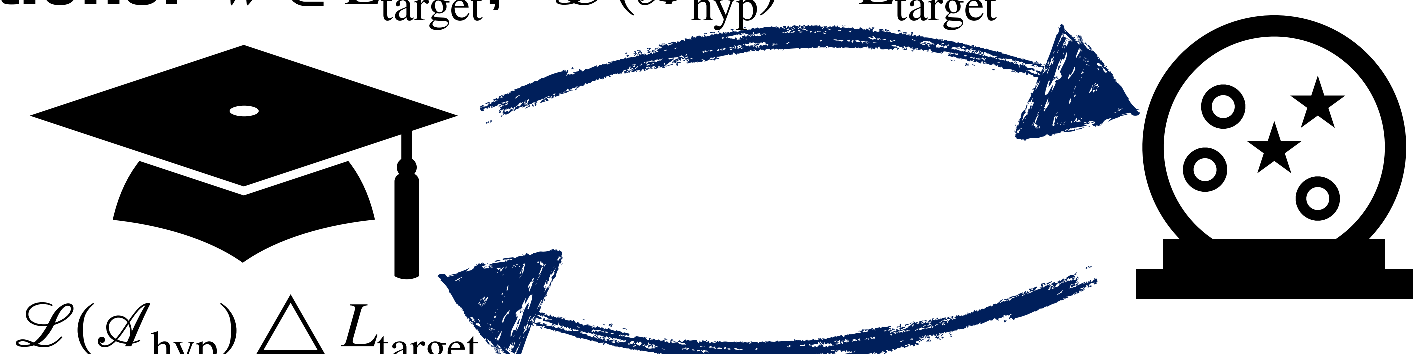
Diff. 3: Definition of "contradiction" to refine P and S is changed

- Due to continuity of time

Membership

Equivalence

Questions: $w \overset{?}{\in} L_{\text{target}}$, $\mathscr{L}(\mathscr{A}_{\text{hyp}}) \overset{?}{=} L_{\text{target}}$



Answers: Yes, No, Evidence $w \in \mathscr{L}(\mathscr{A}_{\text{hyp}}) \triangle L_{\text{target}}$

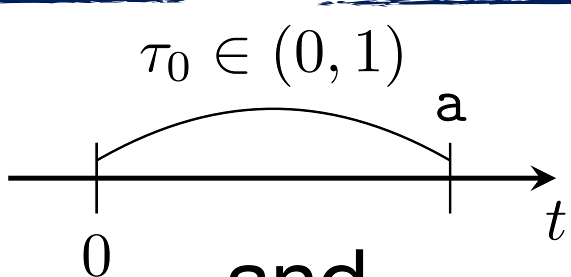
DTA from Nerode-style Congruence

Conceptually similar
e.g. elem. lang.,
renaming

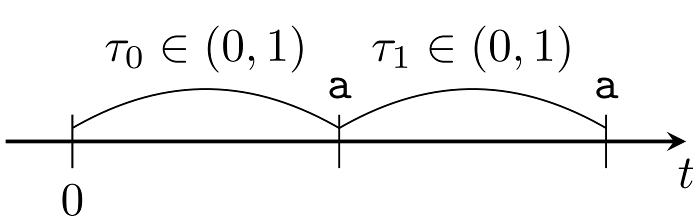
Straightforward

Construction in
[Maler & Pnueli, FoSSaCS'04]

Nerode-style Congruence



and

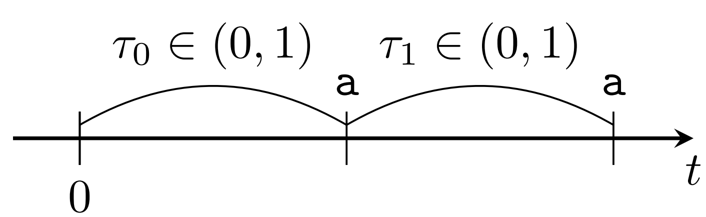


are equivalent

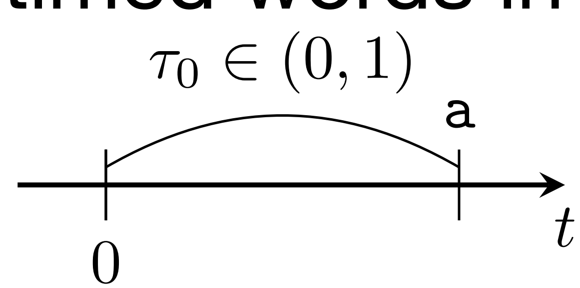
Monoid-based Characterization

[Maler & Pnueli, FoSSaCS'04])

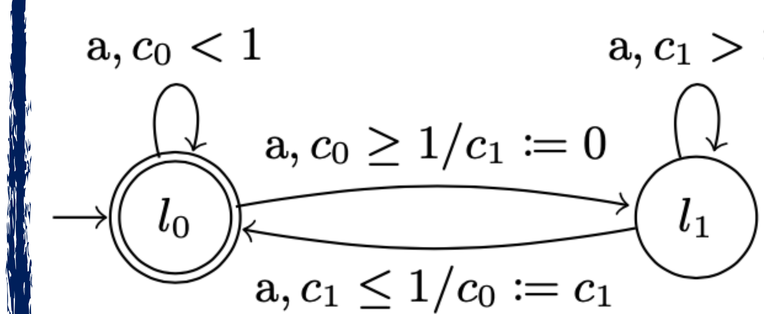
Timed words in



are mapped to
timed words in

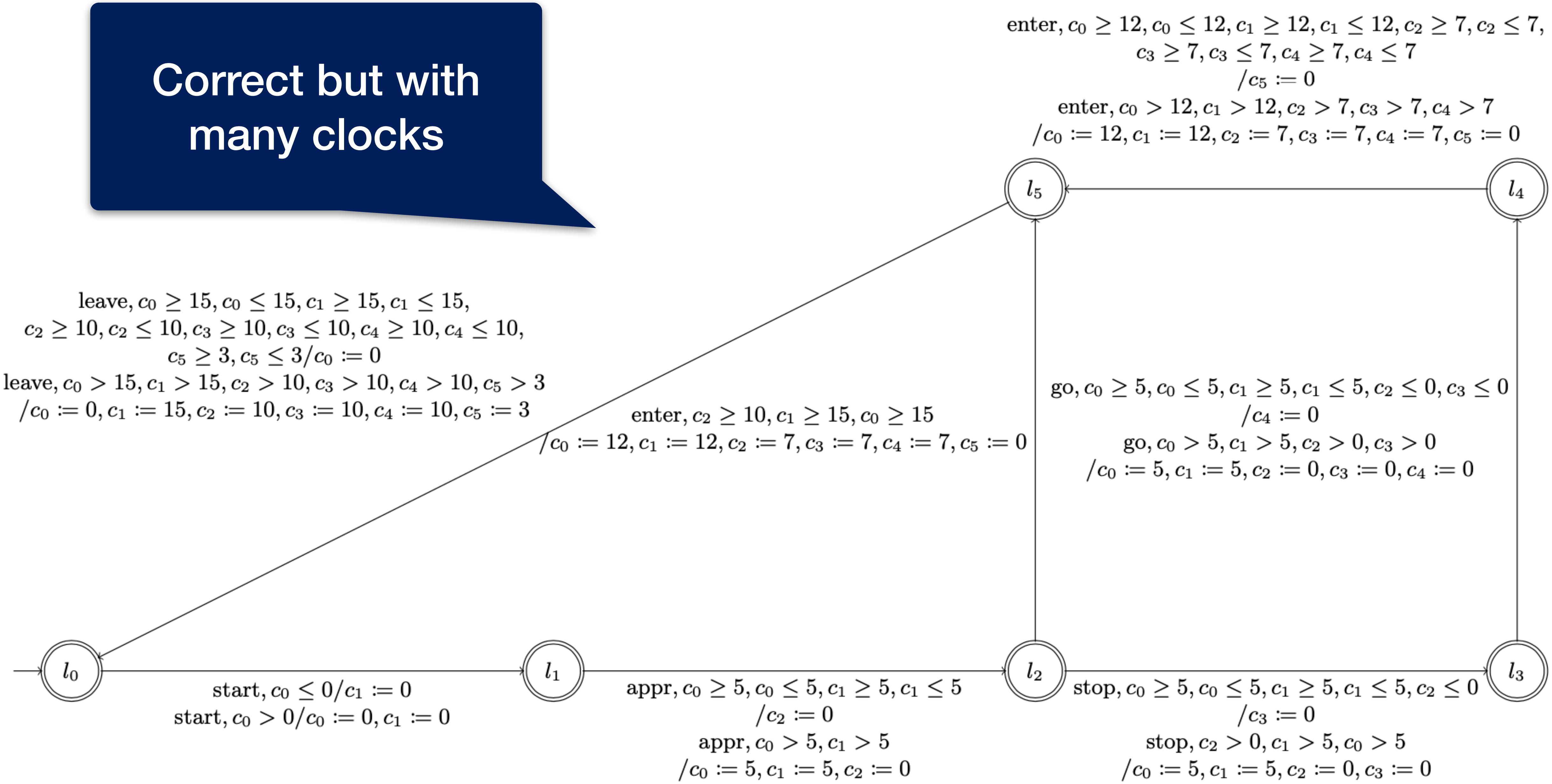


DTA



But no optimization yet…

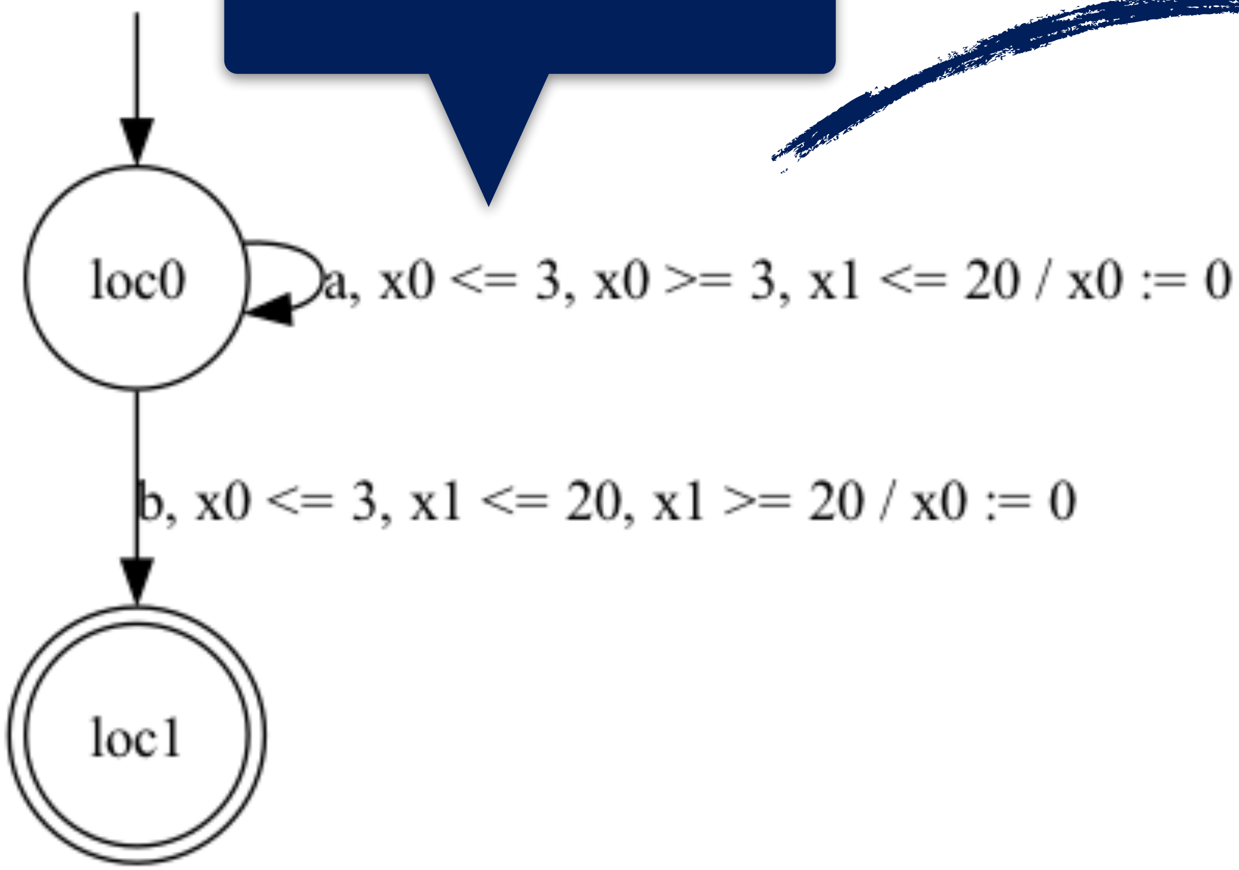
Correct but with many clocks



No minimality in DTA construction

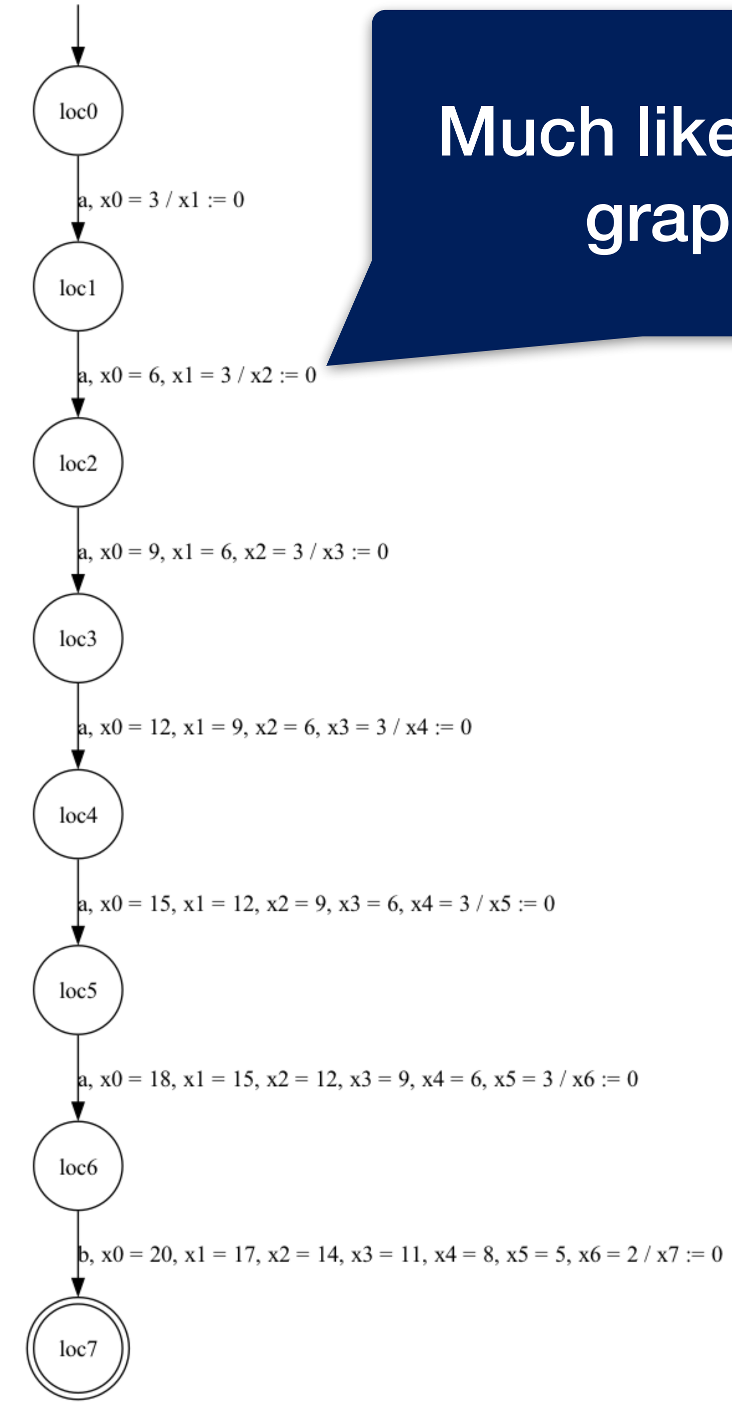
Target DTA

of loops (6) is encoded by clock



Learned DTA

Much like zone graph



This example is based on an input from Frits Vaandrager via private communication

Correctness and Complexity

Theorem

For any DTA $\mathscr{A}_{\mathrm{tgt}}$, $\mathrm{L}^*_{\mathrm{timed}}$ returns a DTA $\mathscr{A}_{\mathrm{hyp}}$ satisfying $\mathscr{L}(\mathscr{A}_{\mathrm{tgt}}) = \mathscr{L}(\mathscr{A}_{\mathrm{hyp}})$ with finite queries.

Correctness with finite queries

Theorem

The number of membership queries is singly exponential to $|Q_{\mathrm{tgt}}|$ and doubly exponential to $|C_{\mathrm{tgt}}|$.

- First exp. of $|C_{\mathrm{tgt}}|$ is same as zone/region
- Another exp. to make symbolic membership
- The other part is polynomial (same as L*)

Outline

- Quick Review: L* algorithm for DFA learning
  - Focusing on Myhill-Nerode theorem & Nerode's congruence
- Active learning of timed lang. recognizable by DTAs
  - Myhill-Nerode-style characterization for timed lang.
  - How the algorithm is extended
- Experiments

Setting of Experiments

- Implemented L^*_{timed} in C++
- Show the results for 7 practical benchmarks taken from literature
  - The other results are in the paper
- Baseline: OneSMT (Xu et al., ATVA'22)
  - Outline is similar but dedicated to one-clock DTAs
  - Python implementation
- Intel Core i9-10980XE 125 GiB RAM with Ubuntu 20.04.5 LTS

of Membership Questions

L^*_{timed} requires more questions due to general setting

Only for one-clock DTAs

	OneSMT	L^*_{timed} (Ours)
Auth. Key Man.	3,453	12,263
Car Alarm Sys.	4,769	66,067
Light	210	3,057
Particle Cont.	10,390	245,134
TCP	4,713	11,300
Train	838	13,487
FDDI	N/A (7 clocks)	9,986,271

of Equivalence Questions

Characterization by $\approx_L^S$ seems helpful to find contradictions

Only for one-clock DTAs

	OneSMT	L^*_{timed} (Ours)
Auth. Key Man.	49	11
Car Alarm Sys.	18	17
Light	7	7
Particle Cont.	29	23
TCP	32	15
Train	13	8
FDDI	N/A (7 clocks)	43

Execution Time

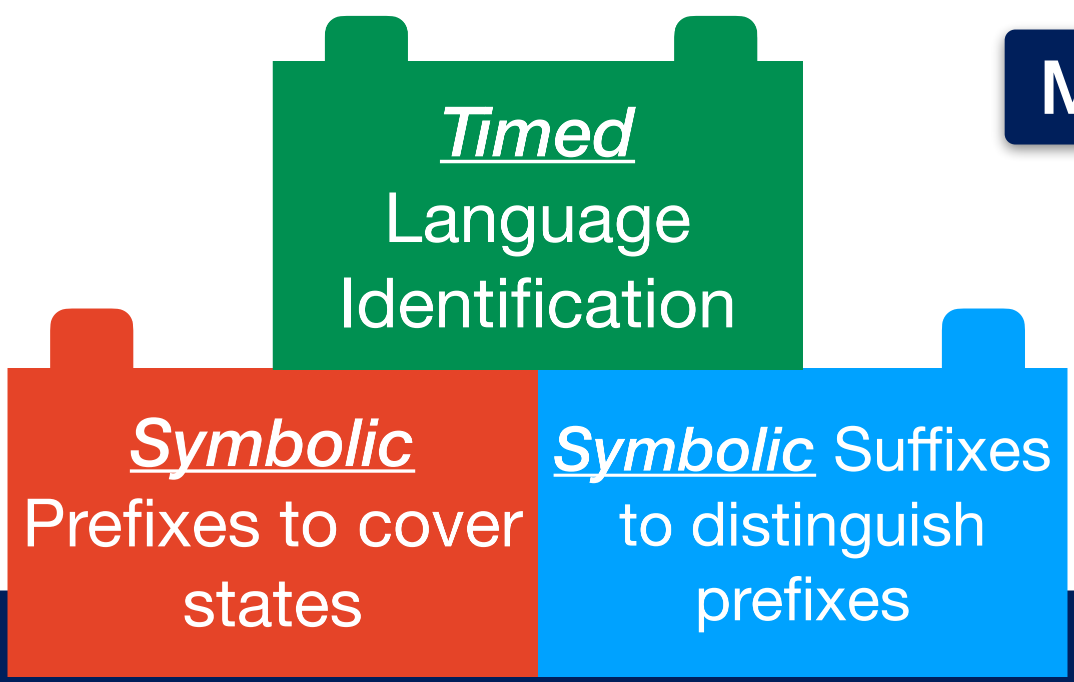
Perhaps C++ vs. Python but can be by less eq. queries

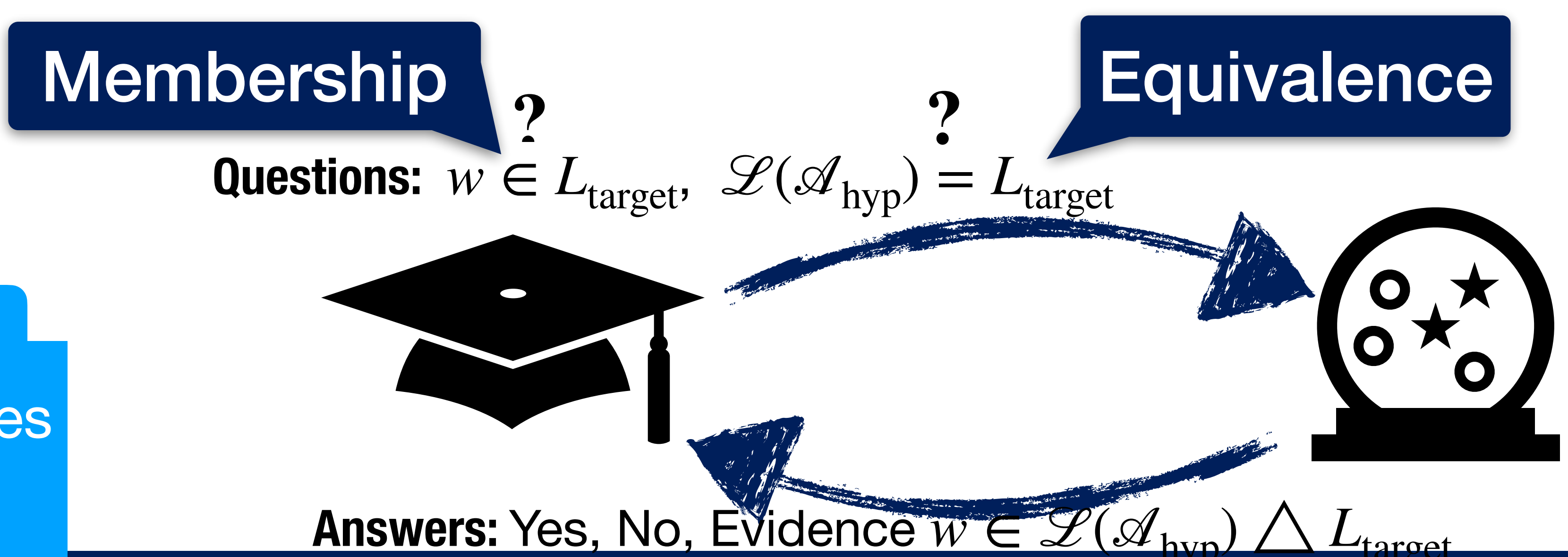
Only for one-clock DTAs

	OneSMT	L^*_{timed} (Ours)
Auth. Key Man.	7.97 sec	0.585 sec
Car Alarm Sys.	95.8 sec	4.65 sec
Light	0.932 sec	0.0330 sec
Particle Cont.	124 sec	64.9 sec
TCP	22.0 sec	0.382 sec
Train	1.13 sec	0.172 sec
FDDI	N/A (7 clocks)	3000 sec

Conclusion

- Myhill-Nerode-style characterization to the timed languages recognizable by DTAs
  - Idea: symbolic handling of timing constraints
- L*-style learning algorithm for DTAs
- Implementation + experiments
  → Works for some practical benchmarks, e.g., FDDI





Future Directions

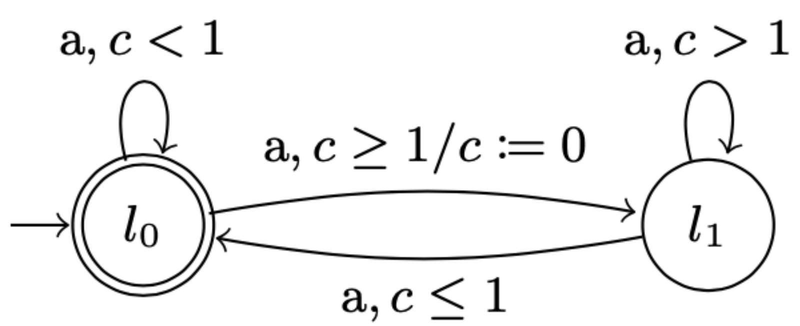
- Comparison with the characterization with nominal sets
- Optimization of DTA construction
    - e.g. reduction of clocks
- Handling of I/O actions
    - action? and action!
- Combination with model checking for testing

M. Waga (Kyoto U.)

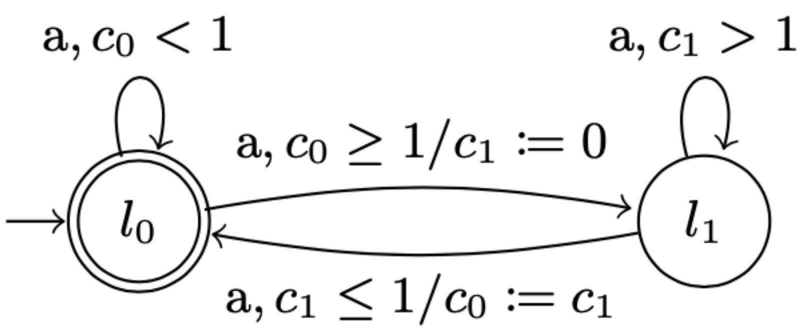
Appendix

Example

Target DTA



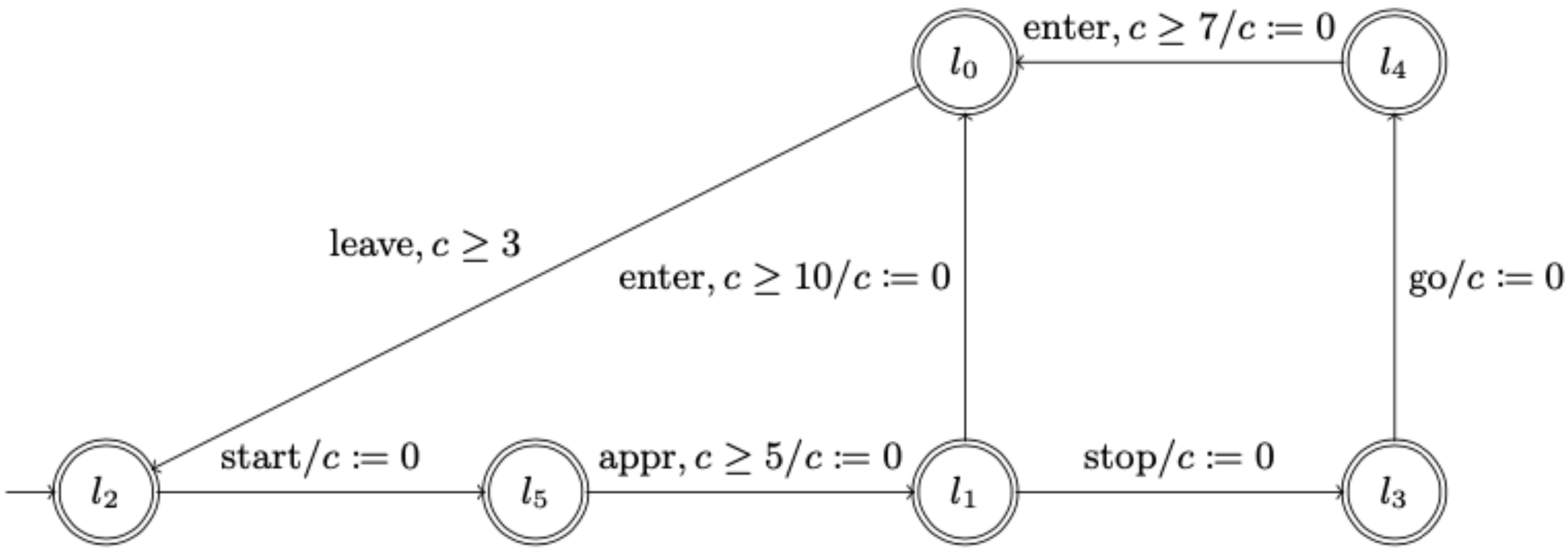
Learned DTA



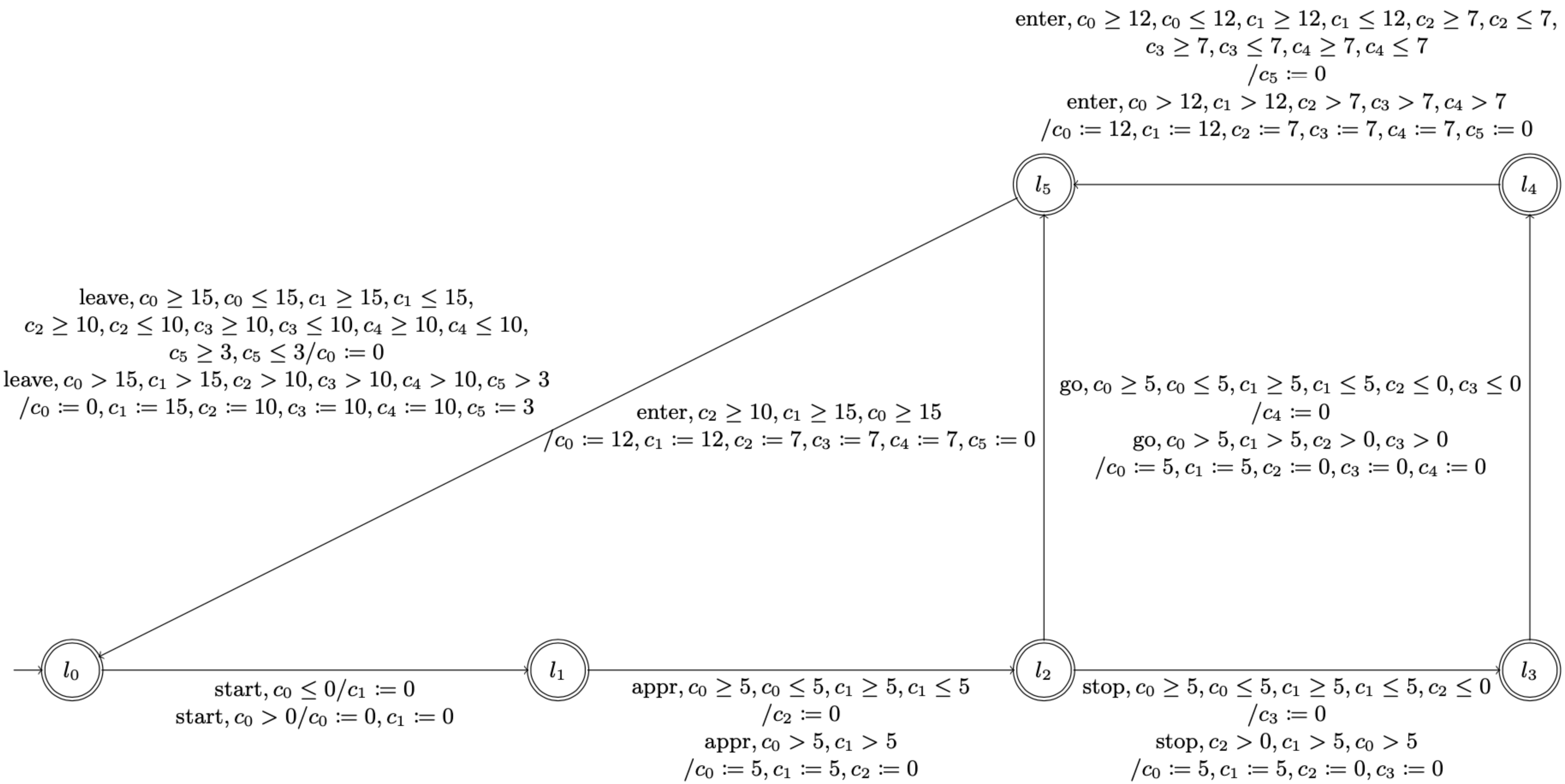
Observation Table

	$(\varepsilon, \tau'_0 = 0)$	$(\mathrm{a}, \tau'_1 = 0 < \tau'_0 < 1)$
$(\varepsilon, \tau_0 = 0)$	$\top$	$\top$
$(\varepsilon, \tau_0 \in (0,1))$	$\top$	$\tau_0 + \tau'_0 \in (0,1)$
$(\varepsilon, \tau_0 = 1)$	$\top$	$\bot$
$(\mathrm{a}, \tau_0 = \tau_0 + \tau_1 = 1 \wedge \tau_1 = 0)$	$\bot$	$\top$
$(\mathrm{a}, \tau_0 = 1 \wedge \tau_1 \in (0,1))$	$\bot$	$\tau_1 + \tau'_0 \in (0,1]$
$(\mathrm{a}, \tau_0 = \tau_1 = 1 \wedge \tau_0 + \tau_1 = 2)$	$\bot$	$\bot$
$(\mathrm{a}, \tau_0 = \tau_0 + \tau_1 = \tau_1 = 0)$	$\top$	$\top$
$(\mathrm{a}, \tau_0 = \tau_0 + \tau_1 \in (0,1) \wedge \tau_1 = 0)$	$\top$	$\tau_0 + \tau_1 + \tau'_0 \in (0,1)$
$(\varepsilon, \tau_0 \in (1,2))$	$\top$	$\bot$
$(\mathrm{aa}, \tau_0 = \tau_0 + \tau_1 = \tau_0 + \tau_1 + \tau_2 = 1 \wedge \tau_1 = \tau_2 = \tau_1 + \tau_2 = 0)$	$\top$	$\top$
$(\mathrm{aa}, \tau_0 = 1 \wedge \tau_1 = \tau_1 + \tau_2 \in (0,1) \wedge \tau_0 + \tau_1 = \tau_0 + \tau_1 + \tau_2 \in (1,2) \wedge \tau_2 = 0)$	$\top$	$\tau_1 + \tau_2 + \tau'_0 \in (0,1)$
$(\mathrm{a}, \tau_0 = 1 < \tau_1 < 2 < \tau_0 + \tau_1 < 3)$	$\bot$	$\bot$
$(\mathrm{aa}, \tau_0 = \tau_1 = \tau_1 + \tau_2 = 1 \wedge \tau_0 + \tau_1 = \tau_0 + \tau_1 + \tau_2 = 2 \wedge \tau_2 = 0)$	$\top$	$\bot$

Example: Train (Target)



Example: Train (Learned)



Summary of Experiment

		$\lvert L\rvert$	$\lvert\Sigma\rvert$	$\lvert C\rvert$	K_C	# of Mem. queries	# of Eq. queries	Exec. time [sec.]
AKM	LearnTA	17	12	1	5	12,263	**11**	**5.85e-01**
	OneSMT	17	12	1	5	**3,453**	49	7.97e+00
CAS	LearnTA	14	10	1	27	66,067	**17**	**4.65e+00**
	OneSMT	14	10	1	27	**4,769**	18	9.58e+01
Light	LearnTA	5	5	1	10	3,057	**7**	**3.30e-02**
	OneSMT	5	5	1	10	**210**	7	9.32e-01
PC	LearnTA	26	17	1	10	245,134	**23**	**6.49e+01**
	OneSMT	26	17	1	10	**10,390**	29	1.24e+02
TCP	LearnTA	22	13	1	2	11,300	**15**	**3.82e-01**
	OneSMT	22	13	1	2	**4,713**	32	2.20e+01
Train	LearnTA	6	6	1	10	13,487	**8**	**1.72e-01**
	OneSMT	6	6	1	10	**838**	13	1.13e+00
FDDI	LearnTA	16	5	7	6	**9,986,271**	**43**	**3.00e+03**