ICECCS

June 15, 2023
Toulouse, France

Expiring opacity problems in parametric timed automata

Étienne André[1], Engel Lefaucheux[2], Dylan Marinho[2]

[1] Université Sorbonne Paris Nord, LIPN, CNRS UMR 7030, F-93430 Villetaneuse, France
[2] Université de Lorraine, CNRS, Inria, LORIA, F-54000, Nancy, France

These works are partially supported by the ANR-NRF research program ProMiS (ANR-19-CE25-0015) and the ANR research program BisoUS (ANR-22-CE48-0012).

A simple example of timing attack

```
1 # input pwd     : Real password
2 # input attempt: Tentative password
3 for i = 0 to min(len(pwd), len(attempt)) - 1 do
4 if pwd[i] =/= attempt[i] then
5 return false
6 done
7 return true
```

A simple example of timing attack

```
1 # input pwd     : Real password
2 # input attempt: Tentative password
3 for i = 0 to min(len(pwd), len(attempt)) - 1 do
4 if pwd[i] =/= attempt[i] then
5 return false
6 done
7 return true
```

pwd	c	h	o	c	o	l	a	t	i	n	e
attempt	c	h	a	s	s	o	u	l	e	t	

Execution time:

A simple example of timing attack

```
1 # input pwd     : Real password
2 # input attempt: Tentative password
3 for i = 0 to min(len(pwd), len(attempt)) - 1 do
4 if pwd[i] =/= attempt[i] then
5 return false
6 done
7 return true
```

pwd	c	h	o	c	o	l	a	t	i	n	e
attempt	c	h	a	s	s	o	u	l	e	t	

Execution time: ϵ

A simple example of timing attack

```
1 # input pwd     : Real password
2 # input attempt: Tentative password
3 for i = 0 to min(len(pwd), len(attempt)) - 1 do
4 if pwd[i] =/= attempt[i] then
5 return false
6 done
7 return true
```

pwd	c	h	o	c	o	l	a	t	i	n	e
attempt	c	h	a	s	s	o	u	l	e	t	

Execution time: $\epsilon + \epsilon$

A simple example of timing attack

```
1 # input pwd     : Real password
2 # input attempt: Tentative password
3 for i = 0 to min(len(pwd), len(attempt)) - 1 do
4 if pwd[i] =/= attempt[i] then
5 return false
6 done
7 return true
```

pwd	c	h	o	c	o	l	a	t	i	n	e
attempt	c	h	a	s	s	o	u	l	e	t	

Execution time: $\epsilon + \epsilon + \epsilon$

A simple example of timing attack

```
1 # input pwd     : Real password
2 # input attempt: Tentative password
3 for i = 0 to min(len(pwd), len(attempt)) - 1 do
4 if pwd[i] =/= attempt[i] then
5 return false
6 done
7 return true
```

pwd	c	h	o	c	o	l	a	t	i	n	e
attempt	c	h	a	s	s	o	u	l	e	t	

Execution time: $\epsilon + \epsilon + \epsilon$

- Problem: The execution time is proportional to the number of consecutive correct characters from the beginning of `attempt`

Context: timing attacks

- Principle: deduce private information from timing data (execution time)

Issues:

- May depend on the implementation (or, even worse, be introduced by the compiler)
- A relatively trivial solution: make the program last always its maximum execution time
  Drawback: loss of efficiency

$\rightsquigarrow$ Non-trivial problem

Informal problems

Question: can we make sure all execution times are secure?

Decision problem: Full execution-time opacity

Can we decide whether it is impossible to infer information on the internal behavior, whatever (for all) execution times?

Informal problems

Question: can we make sure all execution times are secure?

Decision problem: Full execution-time opacity

Can we decide whether it is impossible to infer information on the internal behavior, whatever (for all) execution times?

Further question: can we also tune internal timing constants to make the system resisting to timing attacks?

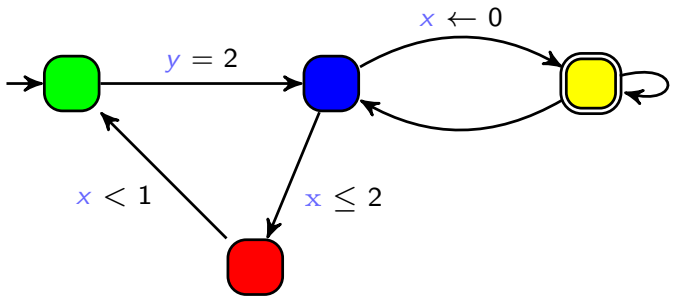
Synthesis problem: Execution-time opacity synthesis

Exhibit execution times and internal timing constants for which it is not possible to infer information on the internal behavior

Outline

Outline

Timed model checking

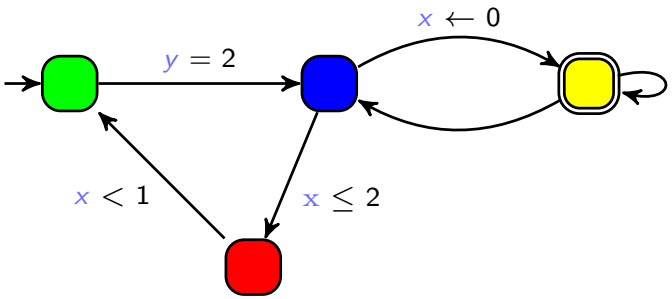


A model of the system

is unreachable

A property to be satisfied

Timed model checking



A model of the system

?

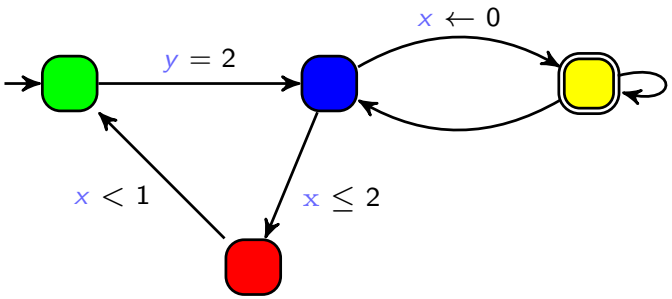
$\models$

is unreachable

A property to be satisfied

- Question: does the model of the system satisfy the property?

Timed model checking



A model of the system

?

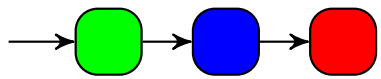
is unreachable

A property to be satisfied

- Question: does the model of the system satisfy the property?

Yes

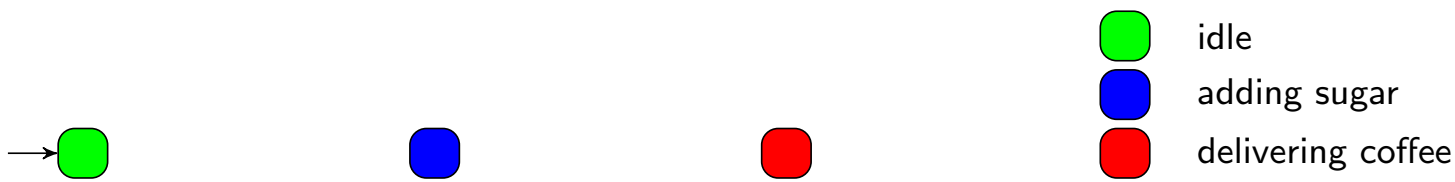
No



Counterexample

Timed automaton (TA)

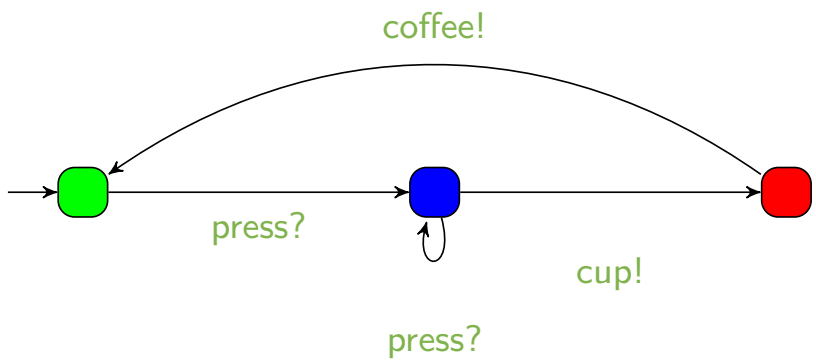
- Finite state automaton (sets of locations)

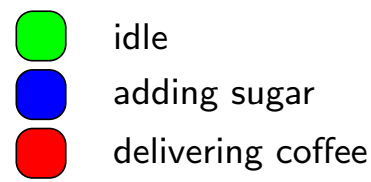


[AD94] Rajeev Alur and David L. Dill. "A theory of timed automata". In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235. DOI: 10.1016/0304-3975(94)90010-8

Timed automaton (TA)

- Finite state automaton (sets of locations and actions)

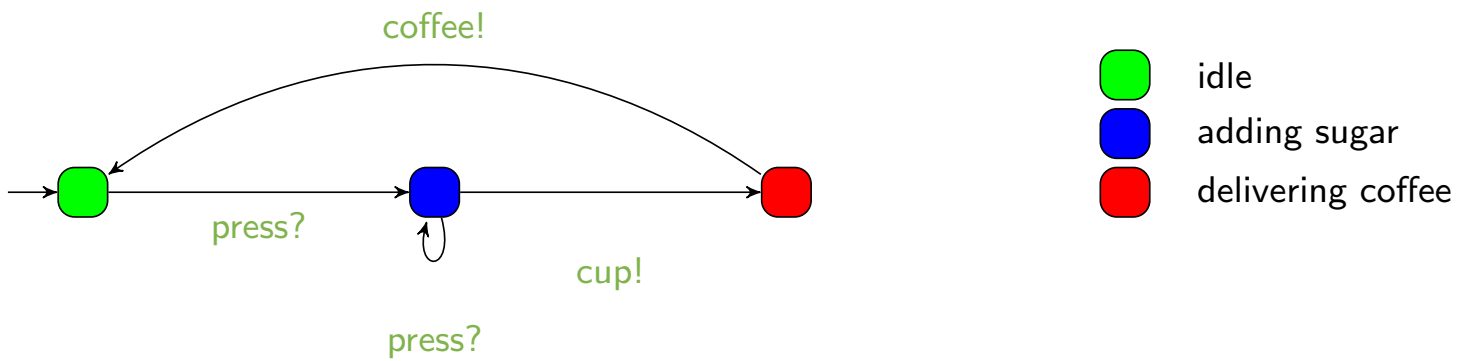




[AD94] Rajeev Alur and David L. Dill. "A theory of timed automata". In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235. DOI: 10.1016/0304-3975(94)90010-8

Timed automaton (TA)

- Finite state automaton (sets of locations and actions) augmented with a set X of clocks [AD94]
  - Real-valued variables evolving linearly at the same rate

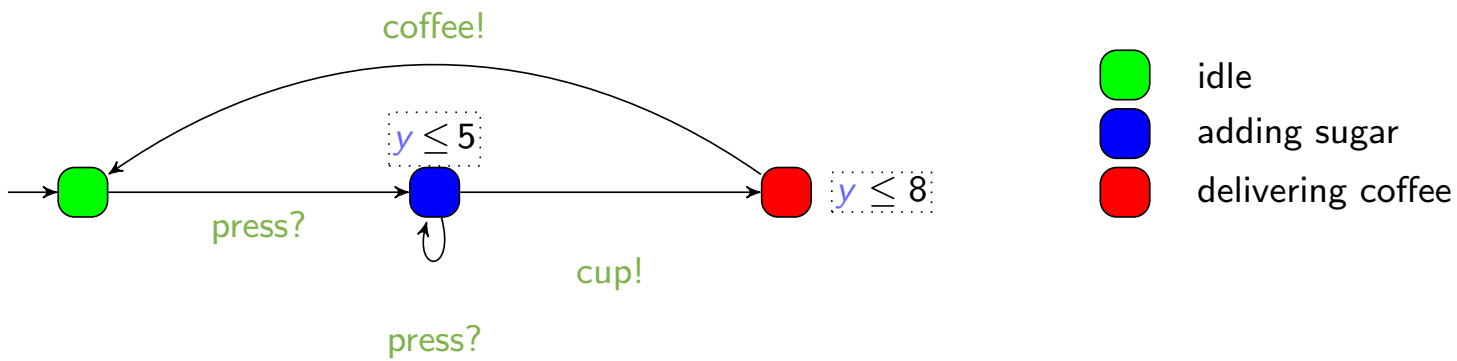


[AD94] Rajeev Alur and David L. Dill. "A theory of timed automata". In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235. DOI: 10.1016/0304-3975(94)90010-8

Timed automaton (TA)

- Finite state automaton (sets of locations and actions) augmented with a set X of clocks [AD94]
  - Real-valued variables evolving linearly at the same rate
  - Can be compared to integer constants in invariants

- Features
  - Location invariant: property to be verified to stay at a location

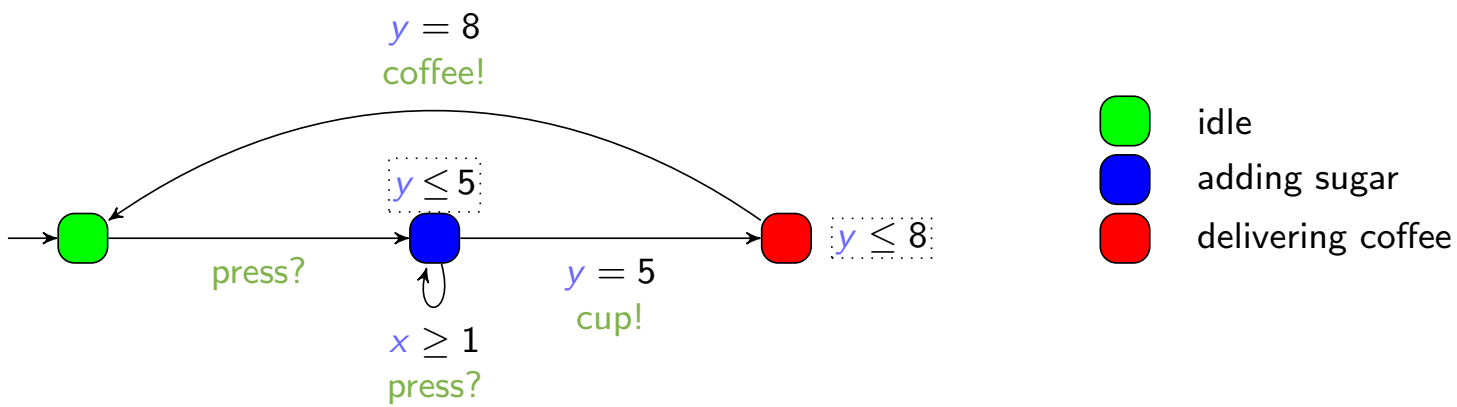


[AD94] Rajeev Alur and David L. Dill. "A theory of timed automata". In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235. DOI: 10.1016/0304-3975(94)90010-8

Timed automaton (TA)

- Finite state automaton (sets of locations and actions) augmented with a set X of clocks [AD94]
  - Real-valued variables evolving linearly at the same rate
  - Can be compared to integer constants in invariants and guards

- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property to be verified to enable a transition

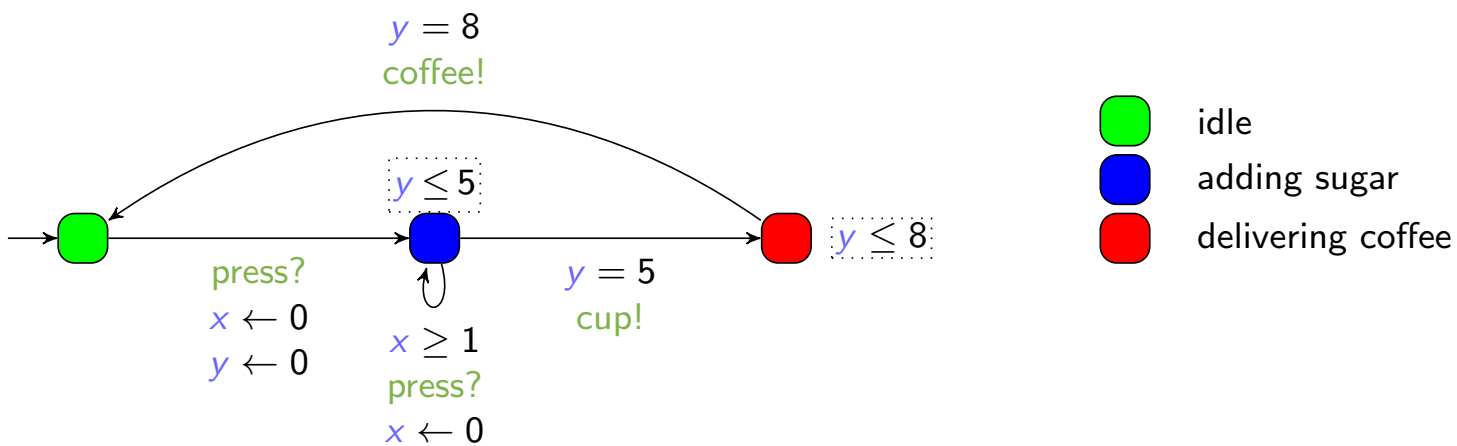


[AD94] Rajeev Alur and David L. Dill. "A theory of timed automata". In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235. DOI: 10.1016/0304-3975(94)90010-8

Timed automaton (TA)

- Finite state automaton (sets of locations and actions) augmented with a set X of clocks [AD94]
  - Real-valued variables evolving linearly at the same rate
  - Can be compared to integer constants in invariants and guards

- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property to be verified to enable a transition
  - Clock reset: some of the clocks can be set to 0 along transitions



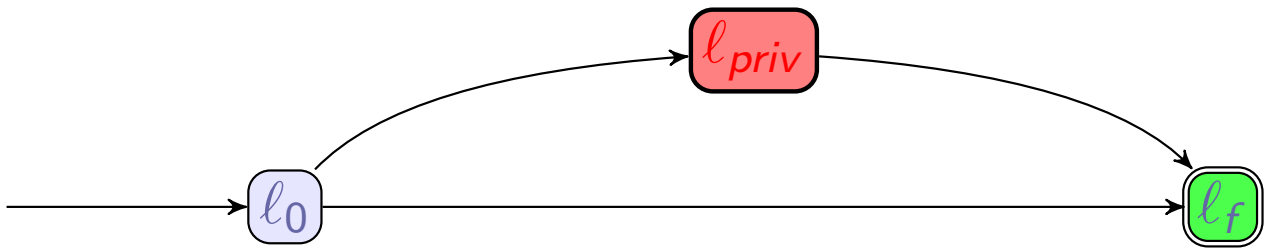
[AD94] Rajeev Alur and David L. Dill. "A theory of timed automata". In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235. DOI: 10.1016/0304-3975(94)90010-8

Outline

Formalization

Hypotheses: [AS19]

- A start location ℓ_0 and an end location ℓ_f
- A special private location ℓ_{priv}



Definition (execution-time opacity)

The system is ET-opaque for a duration d if there exist two runs to ℓ_f of duration d

1. one visiting ℓ_{priv}
2. one *not* visiting ℓ_{priv}

[AS19] Étienne André and Jun Sun. "Parametric Timed Model Checking for Guaranteeing Timed Opacity". In: *ATVA* (Oct. 28–31, 2019). Ed. by Yu-Fang Chen, Chih-Hong Cheng, and Javier Esparza. Vol. 11781. Lecture Notes in Computer Science. Taipei, Taiwan: Springer, 2019, pp. 115–130. DOI: 10.1007/978-3-030-31784-3_7

Three levels of ET-opacity

Existential – $\exists$

There exist two runs of duration d,
one visiting ℓ_{priv},
one not visiting ℓ_{priv}

Weak

For all duration d,
There exists a run of duration d visiting ℓ_{priv}
$\Rightarrow$
There exists a run of duration d not visiting ℓ_{priv}

Full

For all duration d,
There exists a run of duration d visiting ℓ_{priv}
$\Leftrightarrow$
There exists a run of duration d not visiting ℓ_{priv}

Three levels of ET-opacity

Existential – $\exists$

$$\text{private durations} \cap \text{public durations} \neq \emptyset$$

Weak

For all duration d,
There exists a run of duration d visiting ℓ_{priv}
$\Rightarrow$
There exists a run of duration d not visiting ℓ_{priv}

Full

For all duration d,
There exists a run of duration d visiting ℓ_{priv}
$\Leftrightarrow$
There exists a run of duration d not visiting ℓ_{priv}

Three levels of ET-opacity

Existential – $\exists$

private durations $\cap$ public durations $\neq \emptyset$

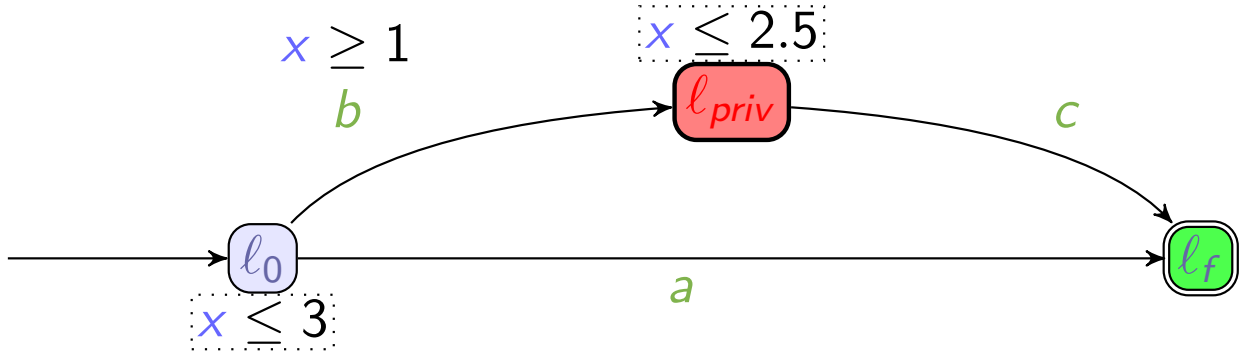
Weak

private durations $\subseteq$ public durations

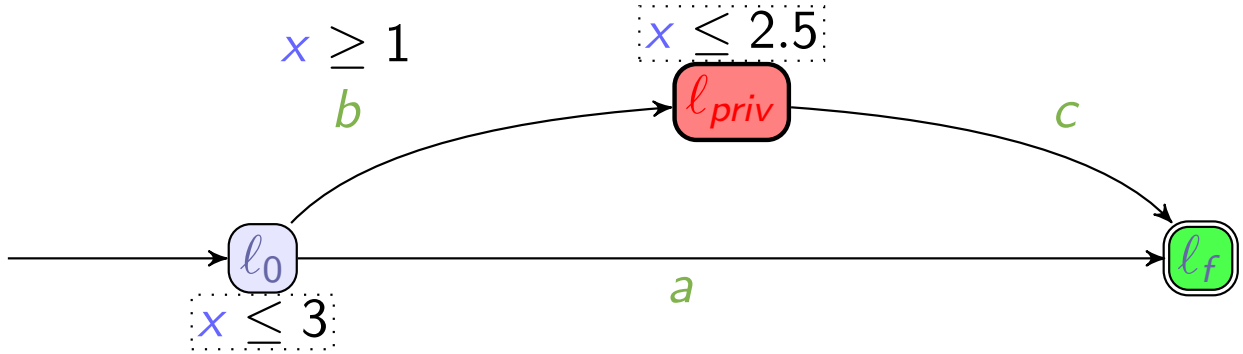
Full

private durations $=$ public durations

Example

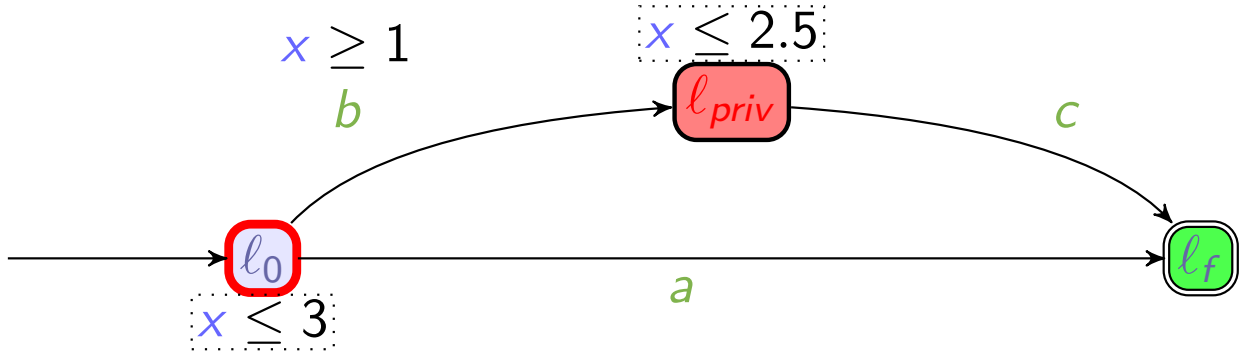


Example



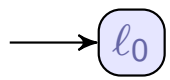
- There exist *(at least)* two runs of duration $d = 2$:

Example

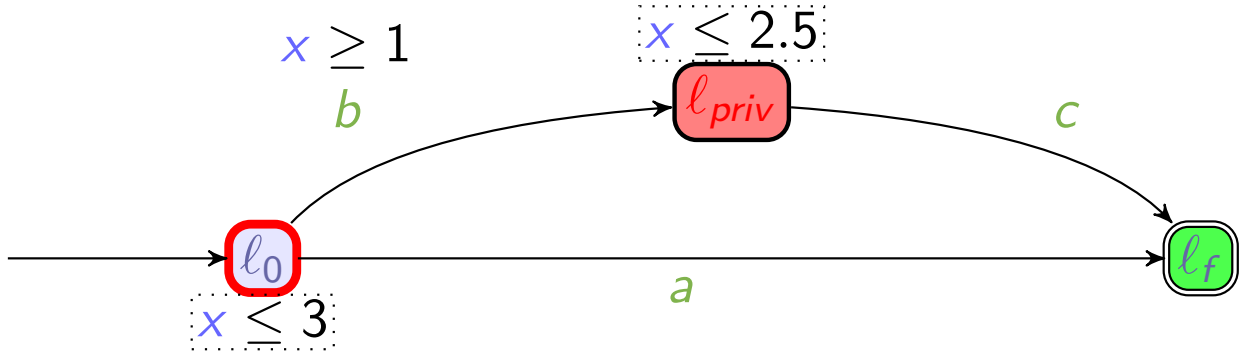


- There exist *(at least)* two runs of duration $d = 2$:

visiting ℓ_{priv}

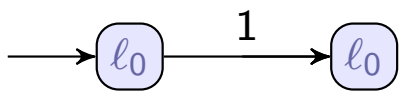


Example

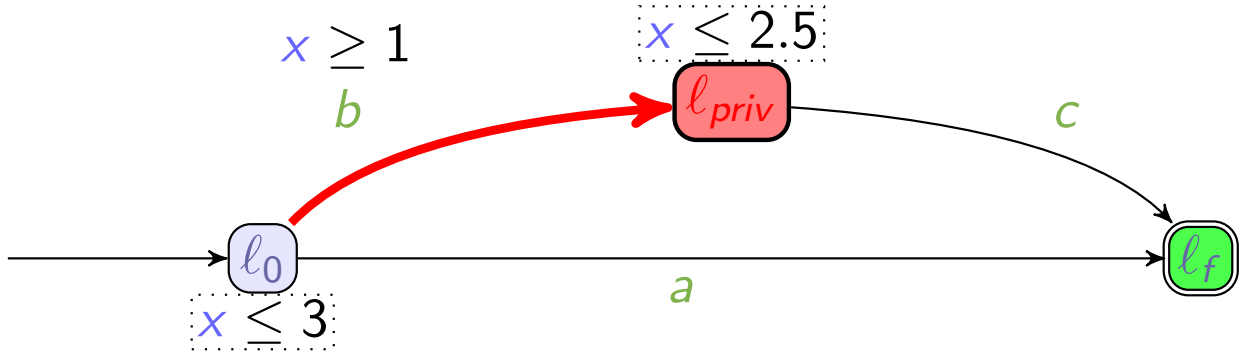


- There exist *(at least)* two runs of duration $d = 2$:

visiting ℓ_{priv}

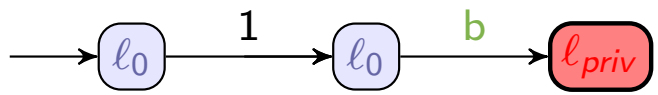


Example

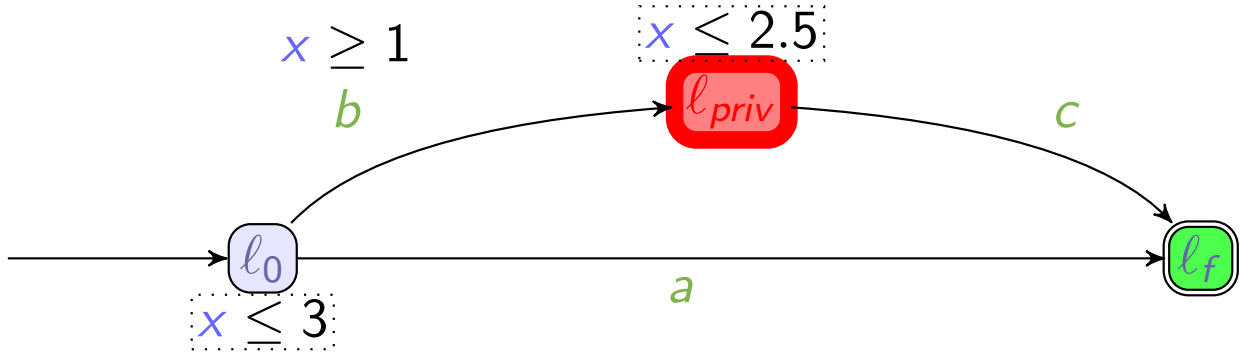


- There exist *(at least)* two runs of duration $d = 2$:

visiting ℓ_{priv}

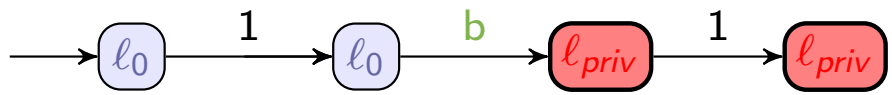


Example

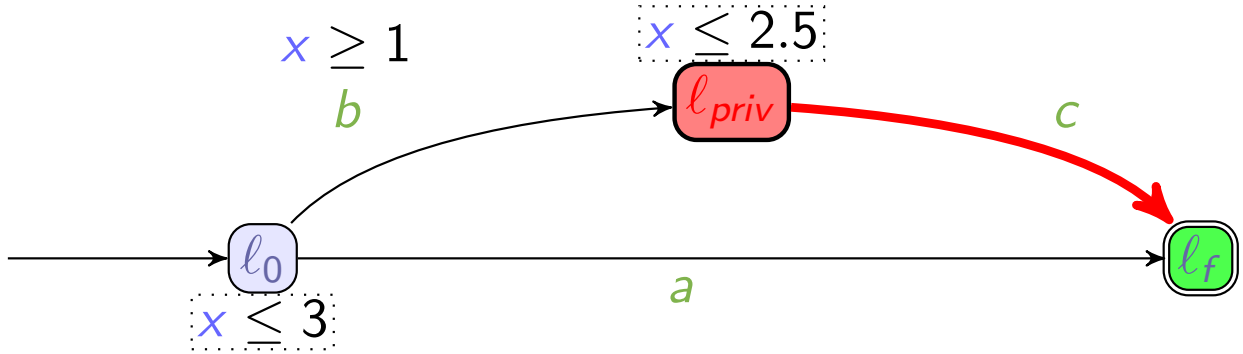


- There exist *(at least)* two runs of duration $d = 2$:

visiting ℓ_{priv}

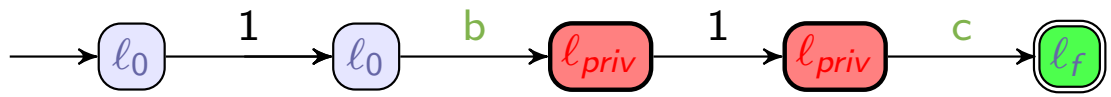


Example

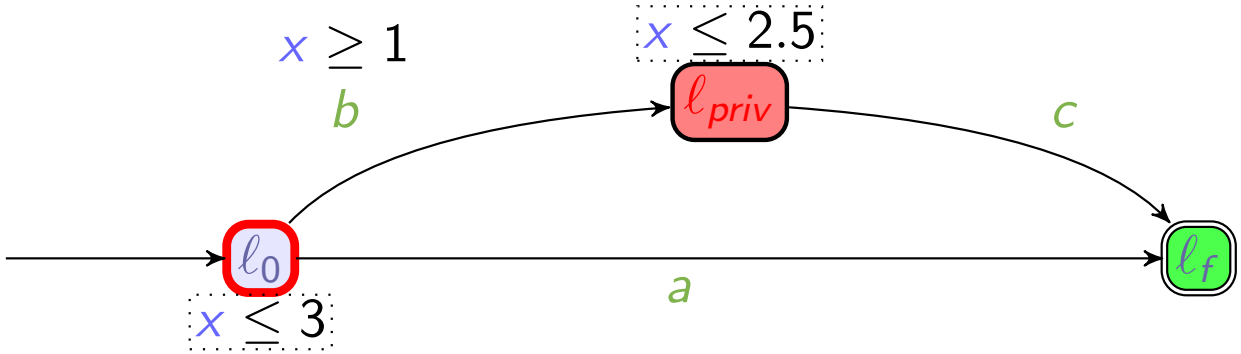


- There exist *(at least)* two runs of duration $d = 2$:

visiting ℓ_{priv}

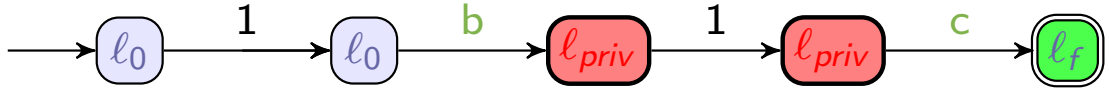


Example

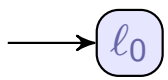


- There exist *(at least)* two runs of duration $d = 2$:

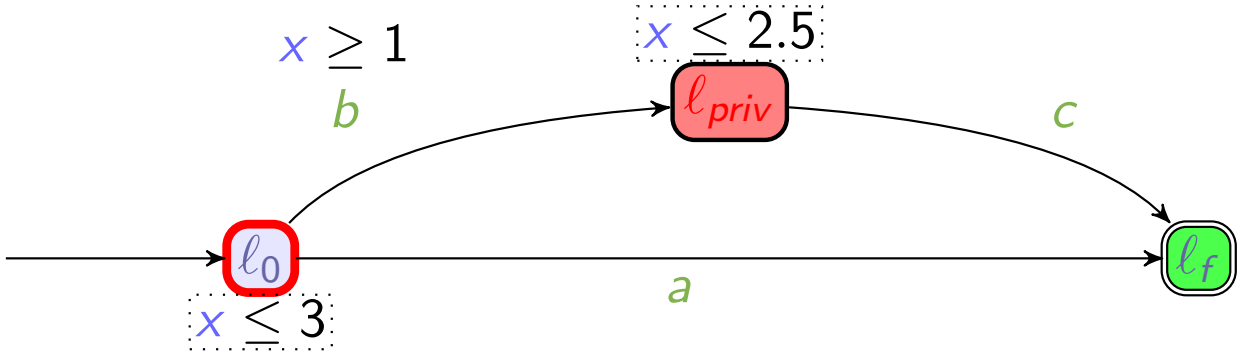
visiting ℓ_{priv}



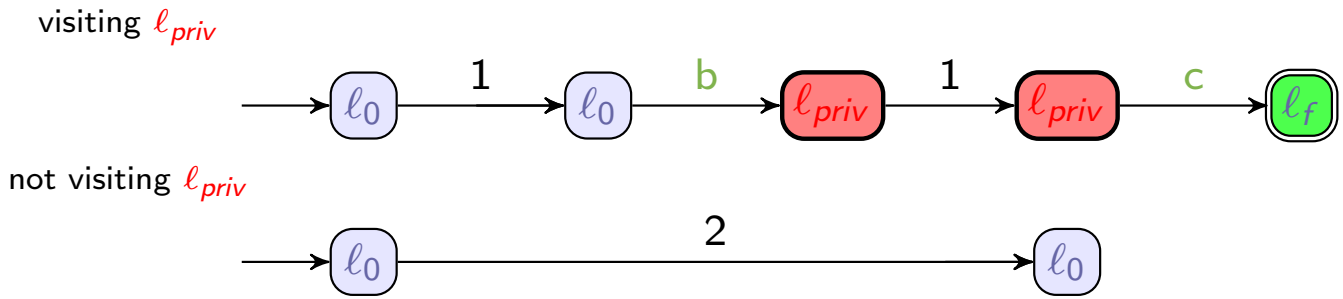
not visiting ℓ_{priv}



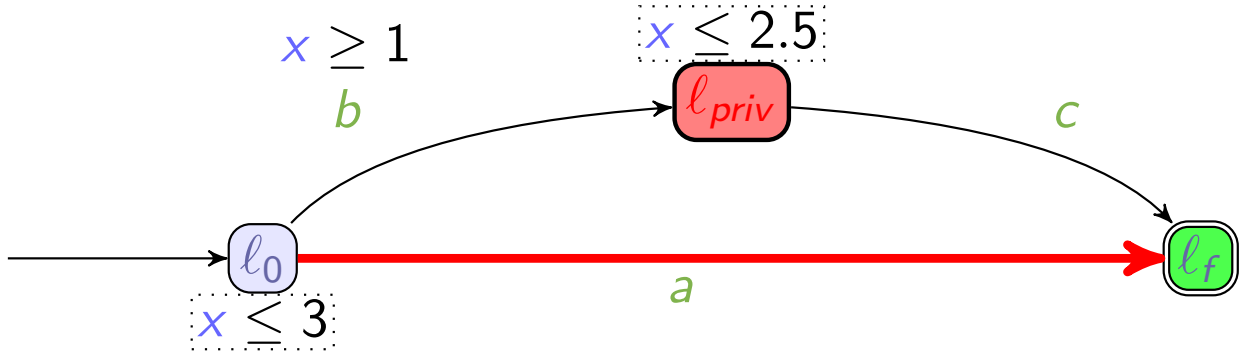
Example



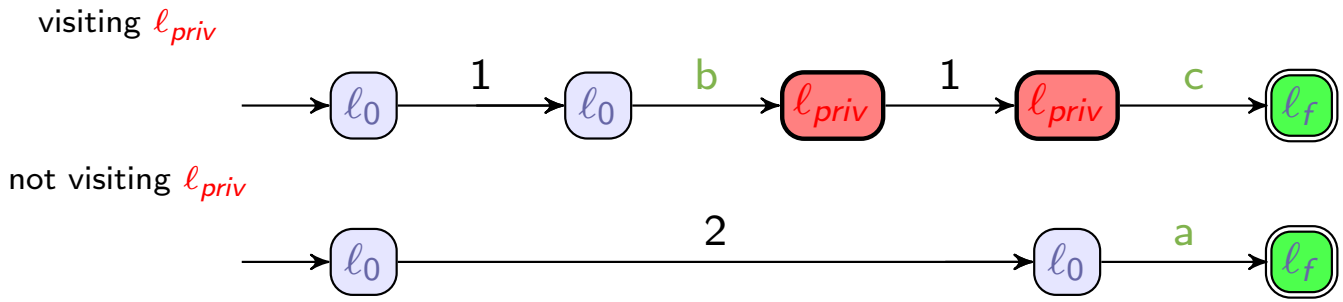
- There exist *(at least)* two runs of duration $d = 2$:



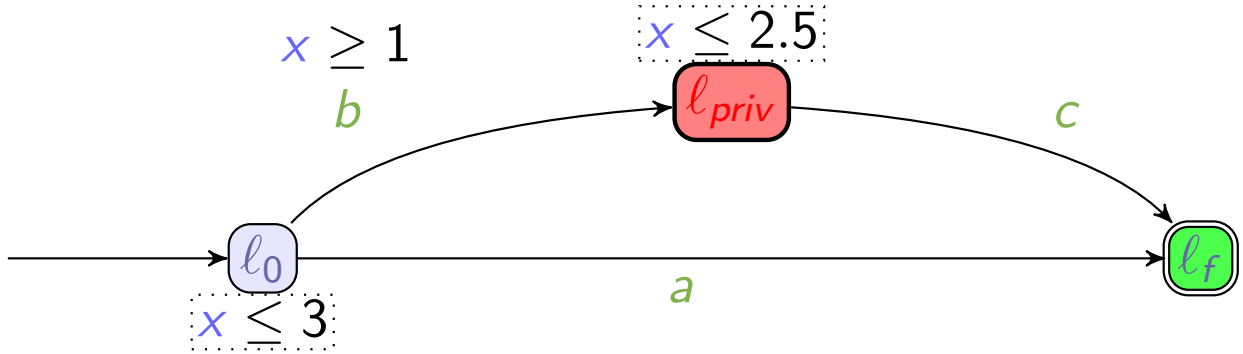
Example



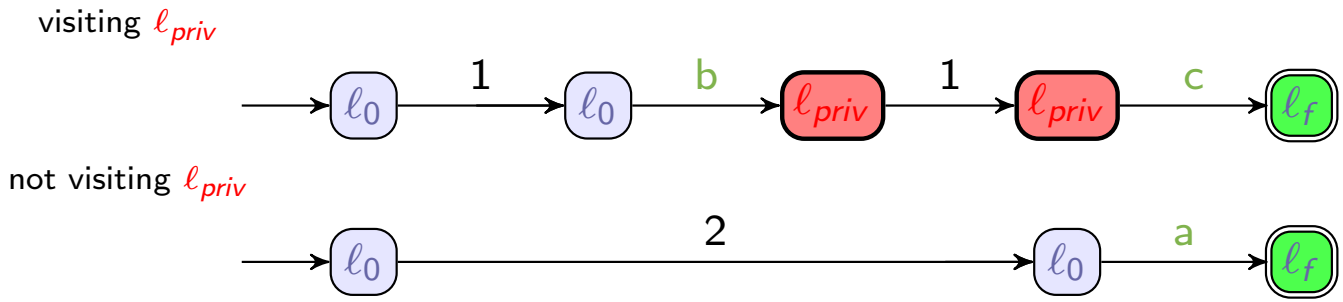
- There exist *(at least)* two runs of duration $d = 2$:



Example



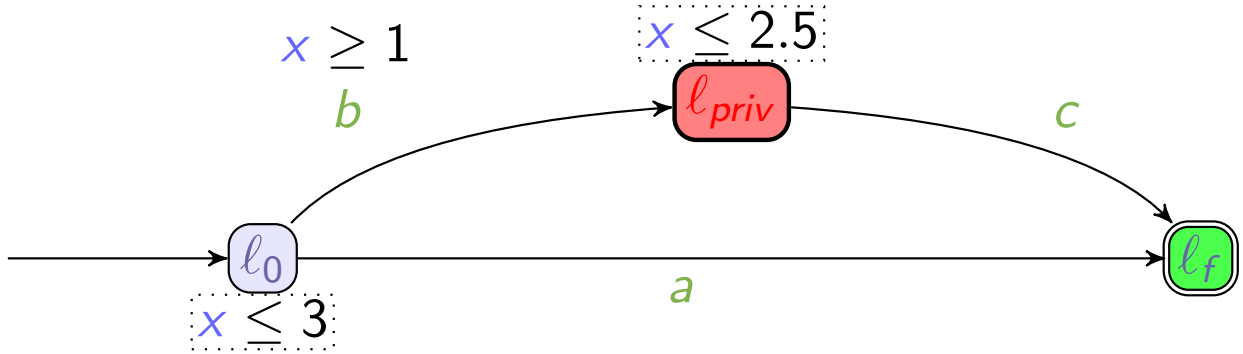
- There exist *(at least)* two runs of duration $d = 2$:



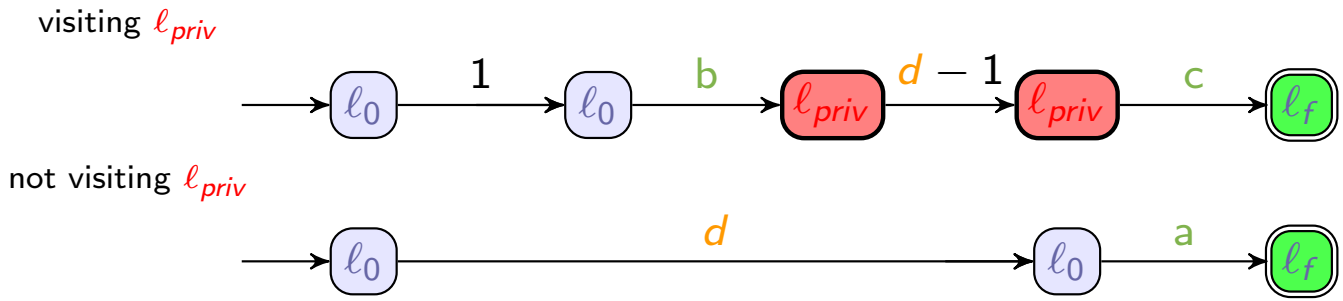
The system is ET-opaque for a duration $d = 2$

The system is $\exists$-ET-opaque

Example



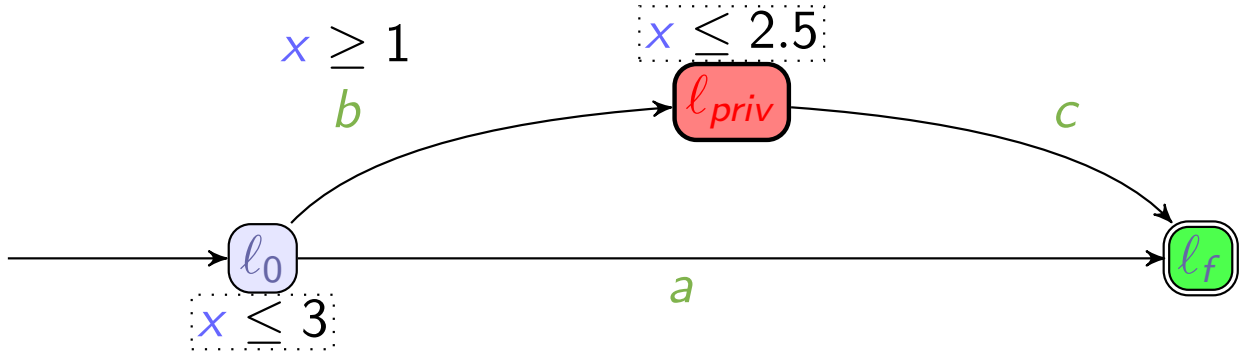
- There exist *(at least)* two runs of duration d for all durations $d \in [1, 2.5]$:



The system is ET-opaque for all durations in $[1, 2.5]$

The system is $\exists$-ET-opaque

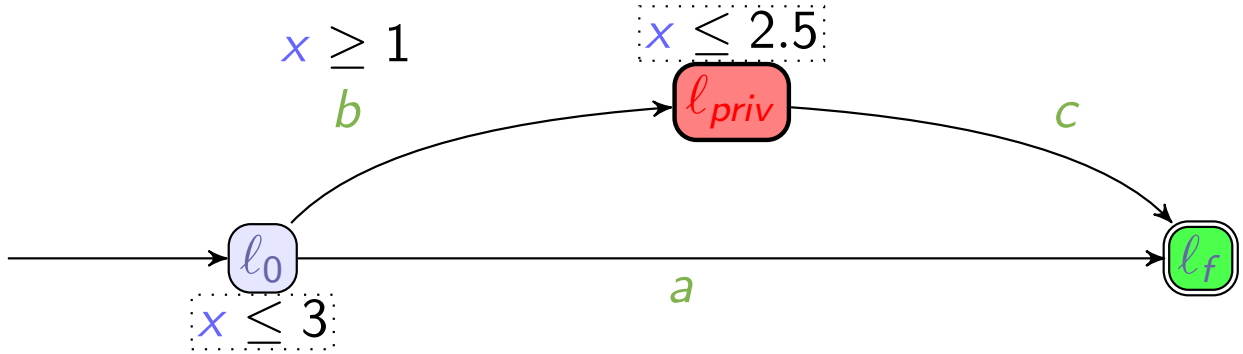
Example



- There exist *(at least)* two runs of duration d for all durations $d \in [1, 2.5]$

The system is $\exists$-ET-opaque

Example

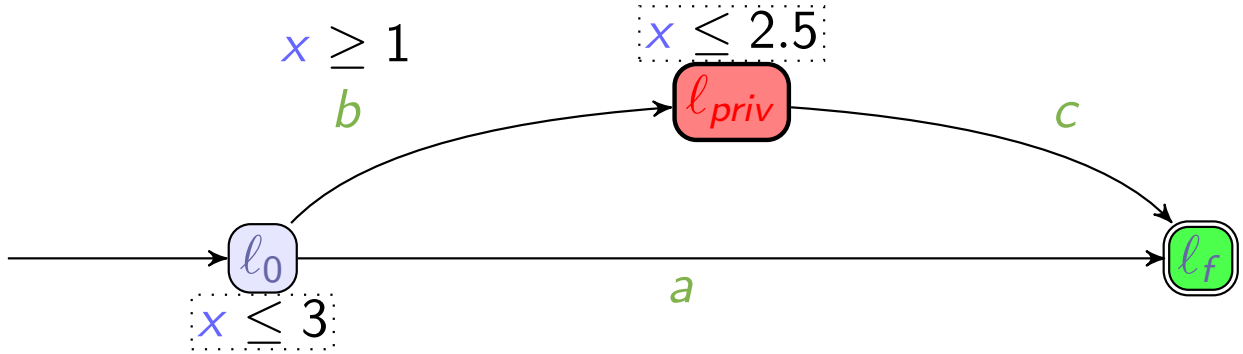


- There exist *(at least)* two runs of duration d for all durations $d \in [1, 2.5]$

The system is $\exists$-ET-opaque

- But,
  - private execution times are $[1, 2.5]$
    public execution times are $[0, 3]$

Example

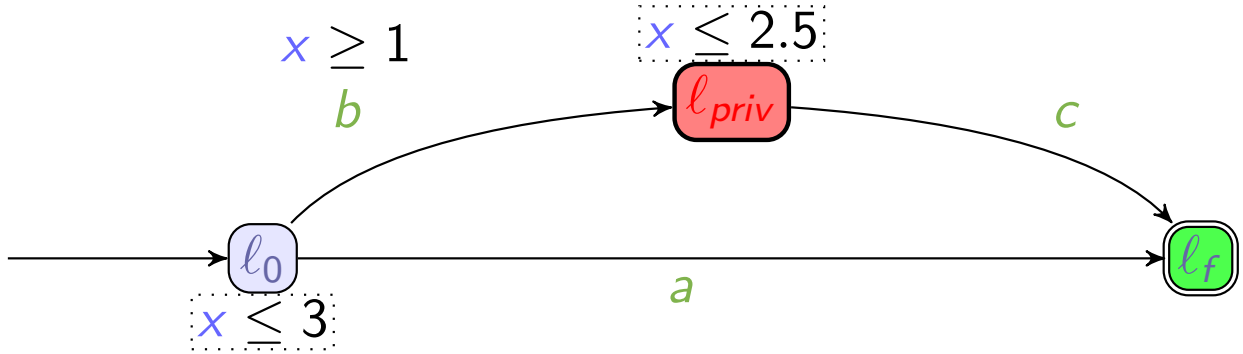


- There exist *(at least)* two runs of duration d for all durations $d \in [1, 2.5]$

The system is $\exists$-ET-opaque

- But,
  - private execution times are $[1, 2.5]$
    public execution times are $[0, 3]$
  - private durations $\subseteq$ public durations

Example



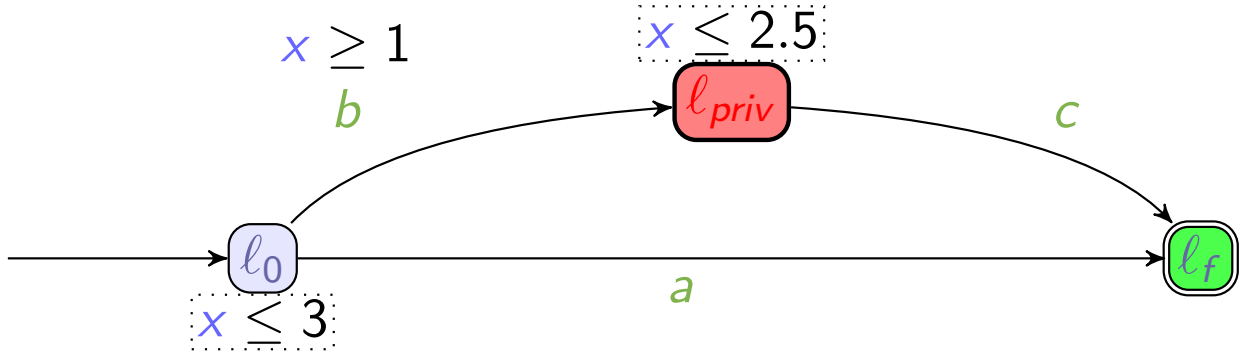
- There exist *(at least)* two runs of duration d for all durations $d \in [1, 2.5]$

The system is $\exists$-ET-opaque

- But,
  - private execution times are $[1, 2.5]$
    public execution times are $[0, 3]$
  - private durations $\subseteq$ public durations

The system is weakly ET-opaque

Example



- There exist *(at least)* two runs of duration d for all durations $d \in [1, 2.5]$

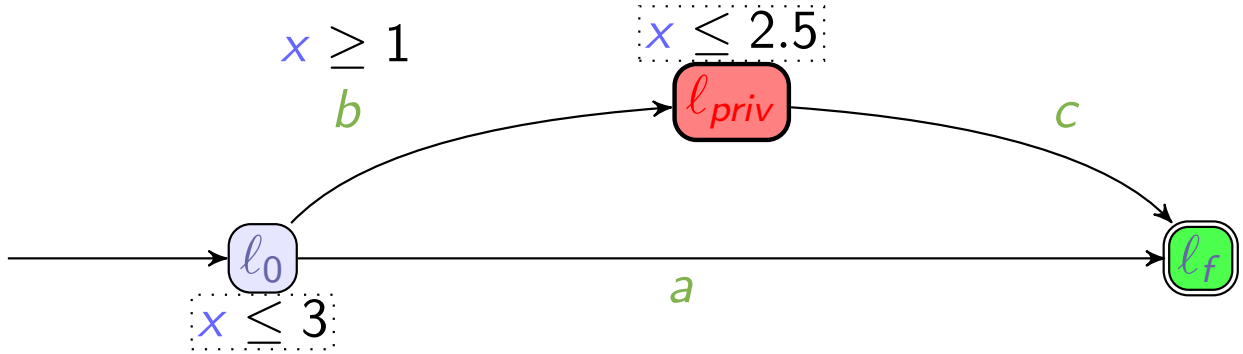
The system is $\exists$-ET-opaque

- But,
  - private execution times are $[1, 2.5]$
    public execution times are $[0, 3]$
  - private durations $\subseteq$ public durations

The system is weakly ET-opaque

  - private durations $\neq$ public durations

Example



- There exist *(at least)* two runs of duration d for all durations $d \in [1, 2.5]$

The system is $\exists$-ET-opaque

- But,
  - private execution times are $[1, 2.5]$
    public execution times are $[0, 3]$
  - private durations $\subseteq$ public durations

The system is weakly ET-opaque

  - private durations $\neq$ public durations

The system is *not* fully ET-opaque

Outline

Outline

Expiring ET-opacity

- How to deal with outdated secrets?
  e. g., cache values, status of the memory, ...

Idea

The secret can expire: beyond a certain duration, knowing the secret is useless to the attacker (e. g., a cache value) [Amm+21][a]

[a] [Amm+21] Ikhlass Ammar, Yamen El Touati, Moez Yeddes, and John Mullins. "Bounded opacity for timed systems". In: *Journal of Information Security and Applications* 61 (Sept. 2021), pp. 1–13. DOI: 10.1016/j.jisa.2021.102926

Expiring ET-opacity

Knowing an expired secret is equivalent to not knowing a secret

	Secret runs	**Non-secret runs**
ET-opacity	Runs visiting the private location (= private runs)	Runs not visiting the private location (= public runs)
expiring-ET-opacity	Private runs with ℓ_{priv} visit $\leq \Delta$ before the system completion	(i) Public runs and (ii) Private runs with ℓ_{priv} visit $> \Delta$ before the system completion

Two levels of ET-opacity

Weak

$$\text{private durations} \subseteq \text{public durations}$$

Full

$$\text{private durations} = \text{public durations}$$

Existential–$\exists$ expiring version is left as future work.

Two levels of expiring ET-opacity

Weak expiring

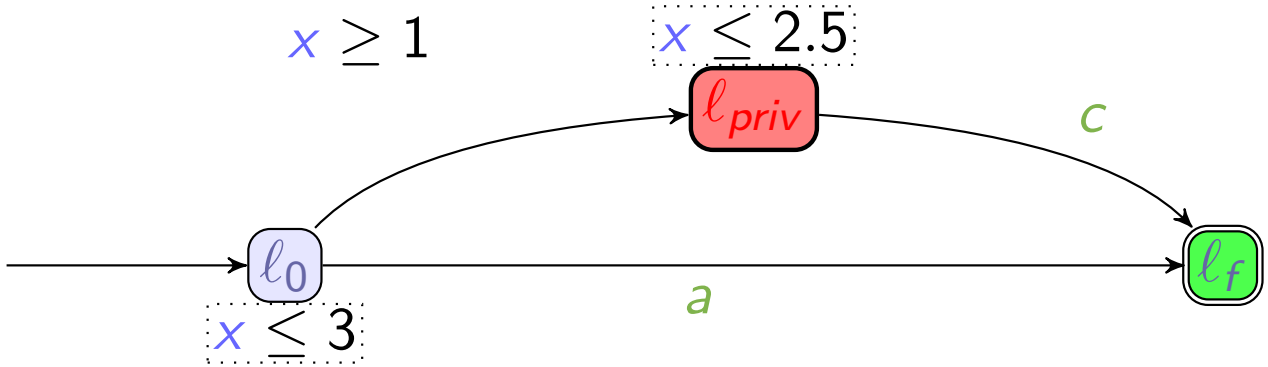
secret durations $\subseteq$ non-secret durations

Full expiring

secret durations $=$ non-secret durations

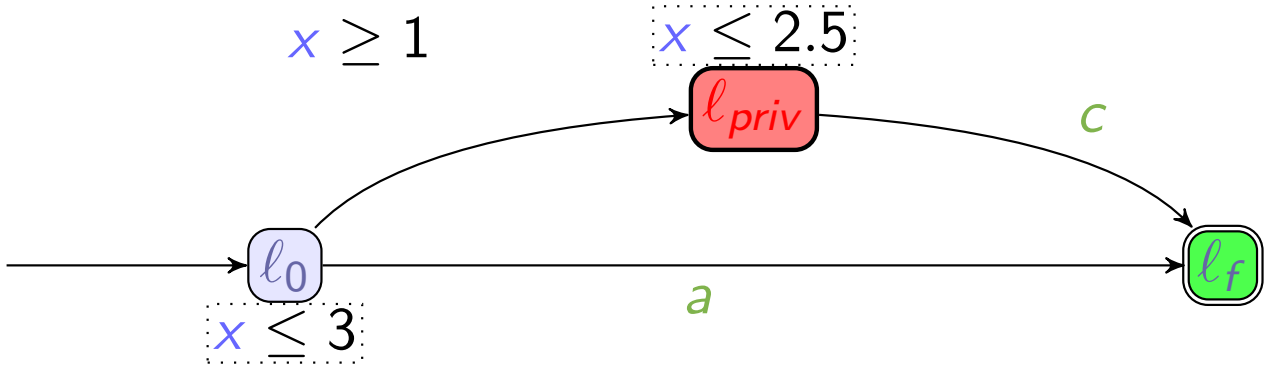
Existential–$\exists$ expiring version is left as future work.

Example



ET-opacity notion		Secret	Non secret	Answer
	$\exists$			$\surd$
	weak	$[1, 2.5]$	$[0, 3]$	$\surd$
	full			$\times$
	$\exists$-exp.			$\surd$
$\Delta = 1$	weak-exp.	$[1, 2.5]$	$(2, 2.5] \cup [0, 3]$	$\surd$
	full-exp.			$\times$

Example

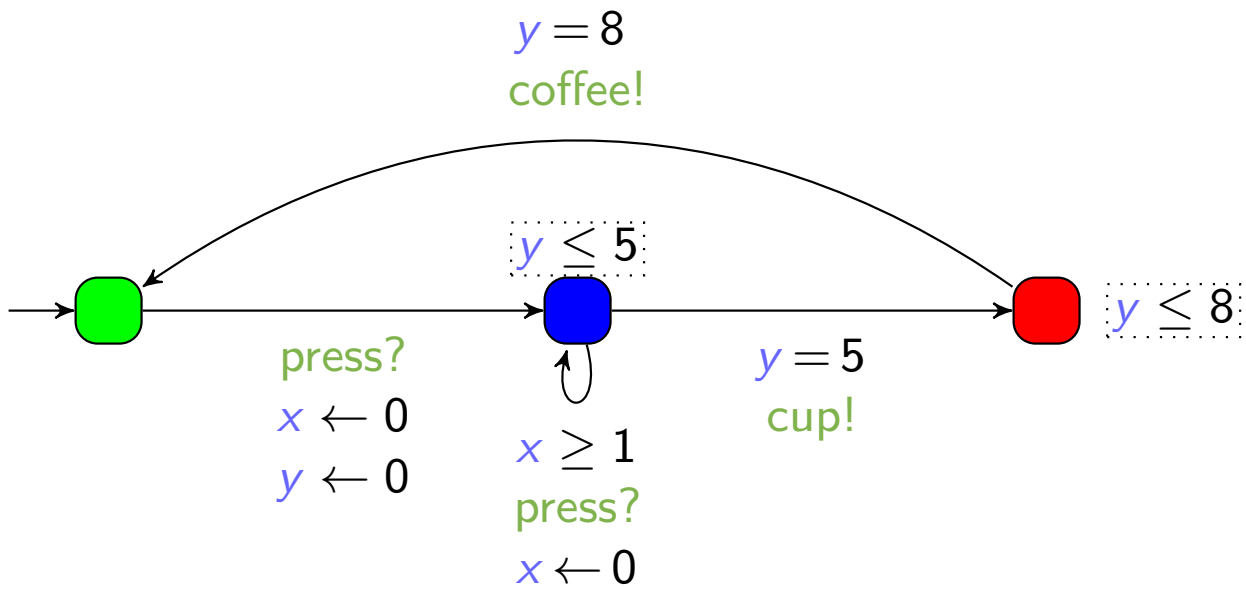


ET-opacity notion		Secret	Non secret	Answer
	$\exists$			$\surd$
	weak	$[1, 2.5]$	$[0, 3]$	$\surd$
	full			$\times$
	$\exists$-exp.			$\surd$
$\Delta = 1$	weak-exp.	$[1, 2.5]$	$(2, 2.5] \cup [0, 3]$	$\surd$
	full-exp.			$\times$
	$\exists$-exp.			$\surd$
$\Delta = 1.25$	weak-exp.	$[1, 2.5]$	$(2.25, 2.5] \cup [0, 3]$	$\surd$
	full-exp.			$\times$

Outline

Timed Automaton (PTA)

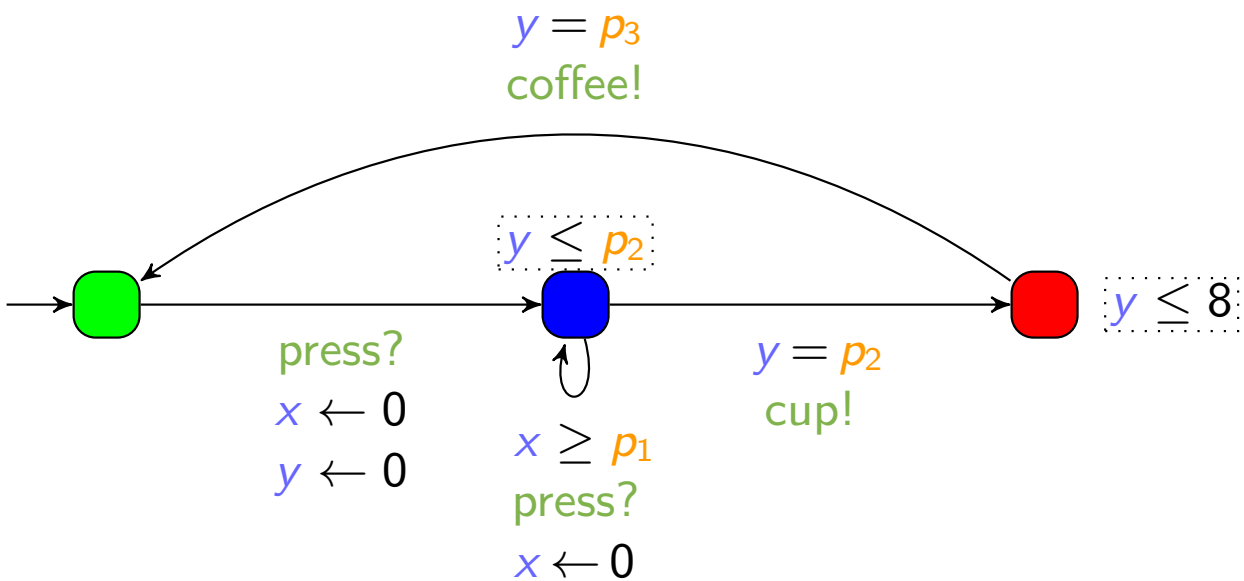
- Timed automaton (sets of locations, actions and clocks)



[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. "Parametric real-time reasoning". In: *STOC* (May 16–18, 1993). Ed. by S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal. San Diego, California, United States: ACM, 1993, pp. 592–601. DOI: 10.1145/167088.167242

Parametric Timed Automaton (PTA)

- Timed automaton (sets of locations, actions and clocks) augmented with a set P of parameters [AHV93]
  - Unknown constants compared to a clock in guards and invariants



[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. "Parametric real-time reasoning". In: *STOC* (May 16–18, 1993). Ed. by S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal. San Diego, California, United States: ACM, 1993, pp. 592–601. DOI: 10.1145/167088.167242

Two classes of parametric problems

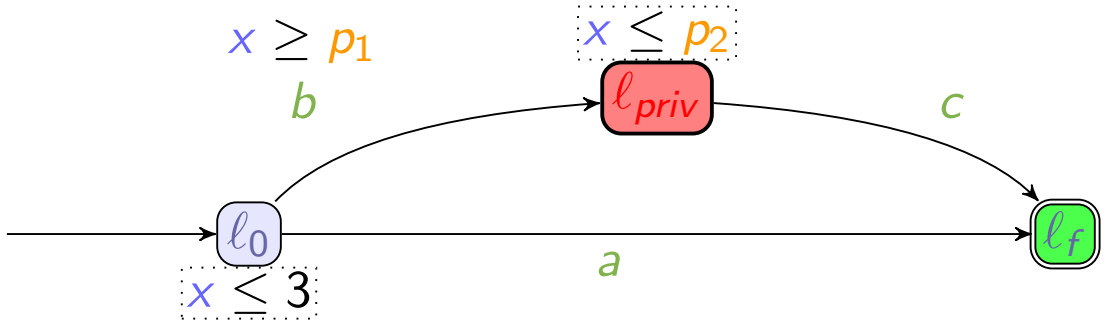
(p+Δ)-Emptiness problem

Decide whether the set of parameter valuations v and Δ s. t. $v(\mathcal{A})$ is expiring-ET-opaque is empty

(p+Δ)-Synthesis problem

Synthesize the set of parameter valuations v and Δ s. t. $v(\mathcal{A})$ is expiring-ET-opaque

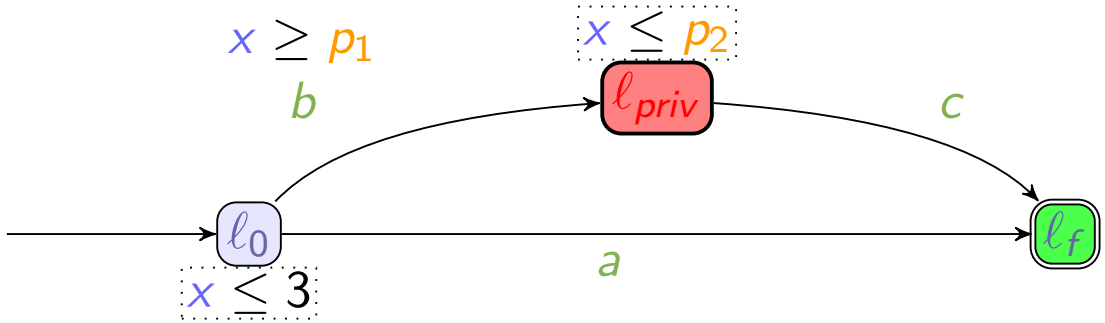
Example



	if $p_1 \leq 3$	otherwise
Secret	$[p_1, \min(\Delta + 3, p_2)]$	$\emptyset$
Non-secret	$(p_1 + \Delta, p_2] \cup [0, 3]$	$\emptyset \cup [0, 3]$

ET-opacity notion	**(p+Δ)-Emptiness**	**(p+Δ)-Synthesis**
weak		
full		

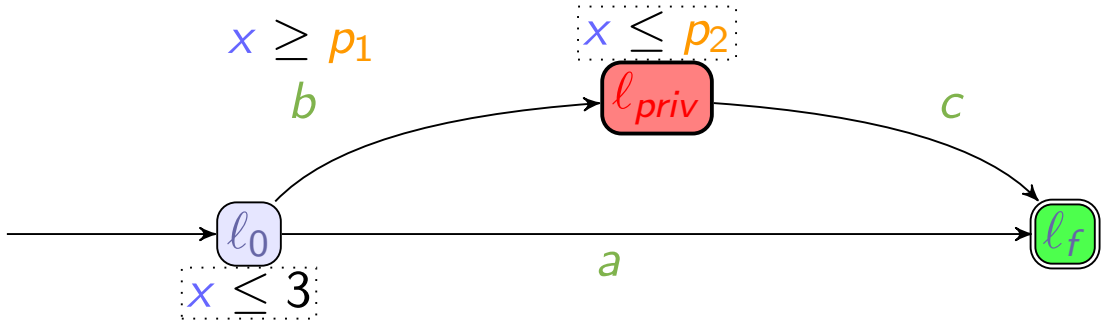
Example



	if $p_1 \leq 3$	otherwise
Secret	$[p_1, \min(\Delta + 3, p_2)]$	$\emptyset$
Non-secret	$(p_1 + \Delta, p_2] \cup [0, 3]$	$\emptyset \cup [0, 3]$

ET-opacity notion	**(p+Δ)-Emptiness**	**(p+Δ)-Synthesis**
weak	$\times_{(\exists \nu)}$	
full	$\times_{(\exists \nu)}$	

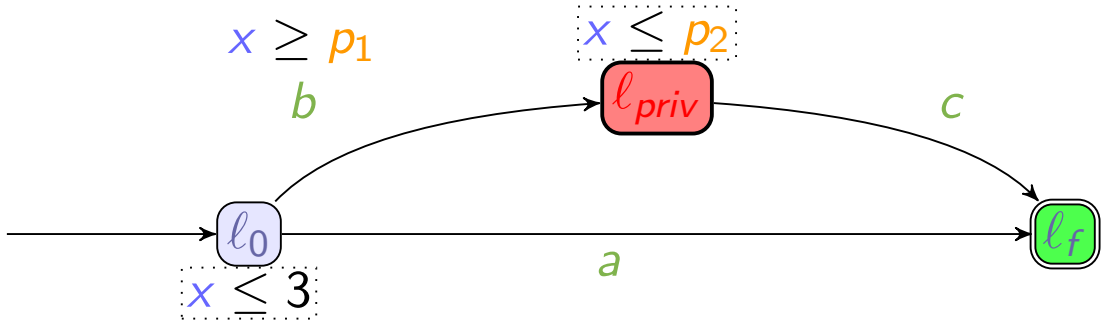
Example



	if $p_1 \leq 3$	otherwise
Secret	$[p_1, \min(\Delta + 3, p_2)]$	$\emptyset$
Non-secret	$(p_1 + \Delta, p_2] \cup [0, 3]$	$\emptyset \cup [0, 3]$

ET-opacity notion	**(p+Δ)-Emptiness**	**(p+Δ)-Synthesis**
weak	$\times_{(\exists \nu)}$	$p_1 > p_2 \quad \vee \quad p_1 > 3 \quad \vee \quad \Delta = 0$ $\vee \quad p_2 \leq 3 \quad \vee \quad p_1 + \Delta <= 3$
full	$\times_{(\exists \nu)}$	

Example



	if $p_1 \leq 3$	otherwise
Secret	$[p_1, \min(\Delta + 3, p_2)]$	$\emptyset$
Non-secret	$(p_1 + \Delta, p_2] \cup [0, 3]$	$\emptyset \cup [0, 3]$

ET-opacity notion	**(p+Δ)-Emptiness**	**(p+Δ)-Synthesis**
weak	$\times_{(\exists v)}$	$p_1 > p_2 \quad \vee \quad p_1 > 3 \quad \vee \quad \Delta = 0$ $\vee \quad p_2 \leq 3 \quad \vee \quad p_1 + \Delta <= 3$
full	$\times_{(\exists v)}$	$p_1 = 0 \quad \wedge \quad (\quad (\Delta \leq 3 \wedge 3 \leq p_2 \leq \Delta + 3)$ $\vee (p_2 = 3) \quad)$

Outline

Summary of the results for expiring-ET-opacity

		weakly expiring-ET-opaque	**fully expiring-ET-opaque**
Δ-emptiness	TA	$\surd$	$\surd$
Δ-synthesis	TA	$\surd$	?
$(p+\Delta)$-emptiness	L/U-PTA	$\times$	$\times$
	PTA	$\times$	$\times$
$(p+\Delta)$-synthesis	L/U-PTA	$\times$	$\times$
	PTA	$\times$	$\times$

L/U-PTA (*Lower/Upper-PTA*): subclass of PTA where the parameters are partitioned into two sets (either compared to clocks as upperbound, or as lower bound) [BL09]

[BL09] Laura Bozzelli and Salvatore La Torre. "Decision problems for lower/upper bound parametric timed automata". In: *Formal Methods in System Design* 35.2 (2009), pp. 121–151. DOI: 10.1007/s10703-009-0074-0

Summary of the results for expiring-ET-opacity

		weakly expiring-ET-opaque	**fully expiring-ET-opaque**
Δ-emptiness	TA	$\surd$	$\surd$
Δ-synthesis	TA	$\surd$	?
$(p+\Delta)$-emptiness	L/U-PTA	$\times$	$\times$
	PTA	$\times$	$\times$
$(p+\Delta)$-synthesis	L/U-PTA	$\times$	$\times$
	PTA	$\times$	$\times$

L/U-PTA (*Lower/Upper-PTA*): subclass of PTA where the parameters are partitioned into two sets (either compared to clocks as upperbound, or as lower bound) [BL09]

Proofs are based on the region automaton (for TAs) and by reduction from EF-emptiness (for PTAs).

(see formal proofs in paper)

[BL09] Laura Bozzelli and Salvatore La Torre. "Decision problems for lower/upper bound parametric timed automata". In: *Formal Methods in System Design* 35.2 (2009), pp. 121–151. DOI: 10.1007/s10703-009-0074-0

Outline

Perspectives

Theory

- $\exists$-expiring-ET-opacity
- Some restricted problems remain open

  e. g., PTA with one clock
- Study more restrictive sub-classes, with the hope to exhibit a decidable one

  Promising subclass: U-PTAs (only upper-bound parameters)

Perspectives

Algorithmic and implementation

- Computation of expiring bounds (and parameters) ensuring expiring-ET-opacity
- Automatic translation of programs to timed automata
- Repairing a non ET-opaque system

References I

[AD94] Rajeev Alur and David L. Dill. "A theory of timed automata". In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235. DOI: 10.1016/0304-3975(94)90010-8.

[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. "Parametric real-time reasoning". In: *STOC* (May 16–18, 1993). Ed. by S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal. San Diego, California, United States: ACM, 1993, pp. 592–601. DOI: 10.1145/167088.167242.

[Amm+21] Ikhlass Ammar, Yamen El Touati, Moez Yeddes, and John Mullins. "Bounded opacity for timed systems". In: *Journal of Information Security and Applications* 61 (Sept. 2021), pp. 1–13. DOI: 10.1016/j.jisa.2021.102926.

References II

[AS19] Étienne André and Jun Sun. "Parametric Timed Model Checking for Guaranteeing Timed Opacity". In: *ATVA* (Oct. 28–31, 2019). Ed. by Yu-Fang Chen, Chih-Hong Cheng, and Javier Esparza. Vol. 11781. Lecture Notes in Computer Science. Taipei, Taiwan: Springer, 2019, pp. 115–130. DOI: 10.1007/978-3-030-31784-3_7.

[BL09] Laura Bozzelli and Salvatore La Torre. "Decision problems for lower/upper bound parametric timed automata". In: *Formal Methods in System Design* 35.2 (2009), pp. 121–151. DOI: 10.1007/s10703-009-0074-0.