TiCSA 2023

23rd April 2023
Paris, France

Configuring timing parameters to ensure opacity

Étienne André

Université Sorbonne Paris Nord, LIPN, CNRS UMR 7030, F-93430 Villetaneuse, France
Based on joint works with Engel Lefaucheux, Didier Lime, Dylan Marinho and Sun Jun

Context: side-channel attacks

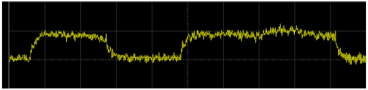
- Threats to a system using non-algorithmic weaknesses

- Example
  - Number of pizzas (and order time) ordered by the white house prior to major war announcements [1]

[1] `http://home.xnet.com/~warinner/pizzacites.html`

Context: side-channel attacks

- Threats to a system using non-algorithmic weaknesses
  - Cache attack
  - Electromagnetic attacks
  - Power attacks



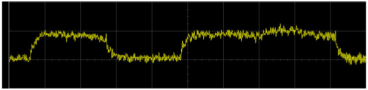
  - Acoustic attacks
  - Timing attacks
  - etc.

- Example
  - Number of pizzas (and order time) ordered by the white house prior to major war announcements [1]

[1] `http://home.xnet.com/~warinner/pizzacites.html`

Context: side-channel attacks

- Threats to a system using non-algorithmic weaknesses
  - Cache attack
  - Electromagnetic attacks
  - Power attacks



  - Acoustic attacks
  - Timing attacks
  - etc.

- Example
  - Number of pizzas (and order time) ordered by the white house prior to major war announcements [1]

[1] http://home.xnet.com/~warinner/pizzacites.html

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 potential solution: make the program last always its maximum execution time
  Drawback: loss of efficiency

$\rightsquigarrow$ Non-trivial problem

A simple example of timing attack

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

A simple example of timing attack

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

pwd	c	h	o	u	d	o	u	f	u
attempt	c	h	e	e	s	e			

Execution time:

A simple example of timing attack

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

pwd	c	h	o	u	d	o	u	f	u
attempt	c	h	e	e	s	e			

Execution time: ϵ

A simple example of timing attack

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

pwd	c	h	o	u	d	o	u	f	u
attempt	c	h	e	e	s	e			

Execution time: $\epsilon + \epsilon$

A simple example of timing attack

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

pwd	c	h	o	u	d	o	u	f	u
attempt	c	h	e	e	s	e			

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

A simple example of timing attack

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

pwd	c	h	o	u	d	o	u	f	u
attempt	c	h	e	e	s	e			

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

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

Outline

Our attacker model

Attacker capabilities

- Has access to the model (white box)
- Can only observe the execution time

Attacker goal

- Wants to deduce some private information based on these observations

Informal problems

Question: can we exhibit secure execution times?

Execution-time opacity computation

Compute execution times for which the attacker cannot deduce private information by observing the execution time

Informal problems

Question: can we exhibit secure execution times?

Execution-time opacity computation

Compute execution times for which the attacker cannot deduce private information by observing the execution time

Question: can we decide whether all execution times are secure?

Full execution-time opacity

Decide whether the attacker cannot deduce private information, for all execution times

Informal problems: configuration

Question: can we also configure internal timing constants to make the system resisting to timing attacks?

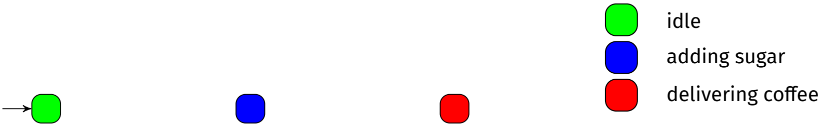
Execution-time opacity synthesis

Exhibit execution times and internal timing constants for which the attacker cannot deduce private information by observing the execution time

Outline

Timed automaton (TA)

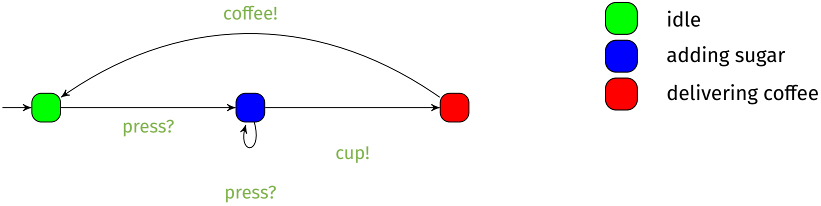
- Finite-state automaton (sets of locations)



[AD94] Rajeev Alur and David L. Dill. “A theory of timed automata”. In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235

Timed automaton (TA)

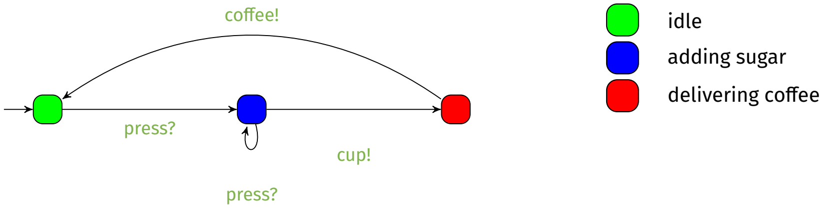
- Finite-state automaton (sets of locations and actions)



[AD94] Rajeev Alur and David L. Dill. "A theory of timed automata". In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235

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

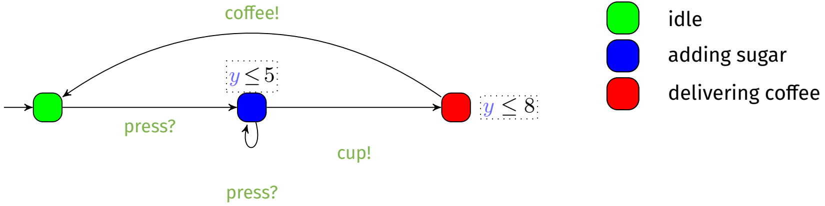


[AD94] Rajeev Alur and David L. Dill. "A theory of timed automata". In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235

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

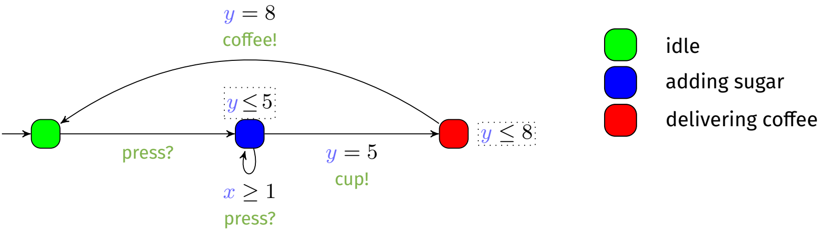


[AD94] Rajeev Alur and David L. Dill. "A theory of timed automata". In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235

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

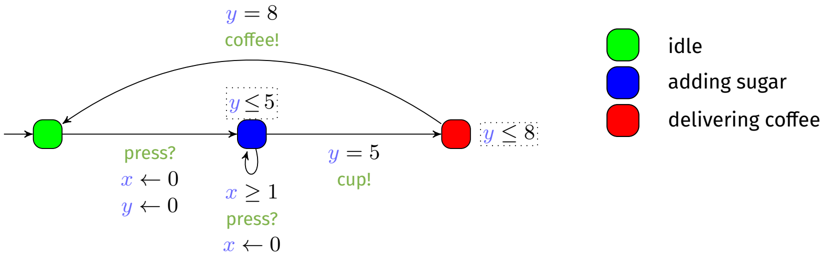


[AD94] Rajeev Alur and David L. Dill. "A theory of timed automata". In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235

Timed automaton (TA)

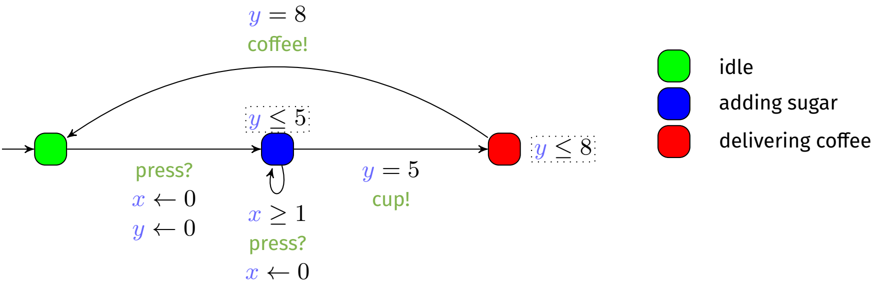
- Finite-state automaton (sets of locations and actions) augmented with a set X of clocks [AD94]
  - Real-valued variables evolving linearly at the same rate
  - Can be compared to integer constants in invariants and guards

- 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

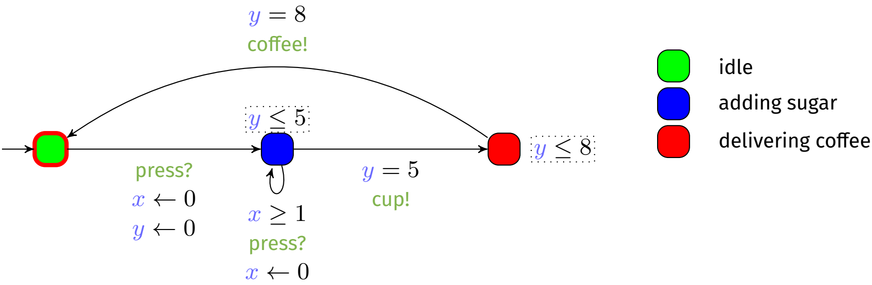


[AD94] Rajeev Alur and David L. Dill. “A theory of timed automata”. In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235

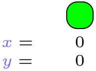
Examples of executions



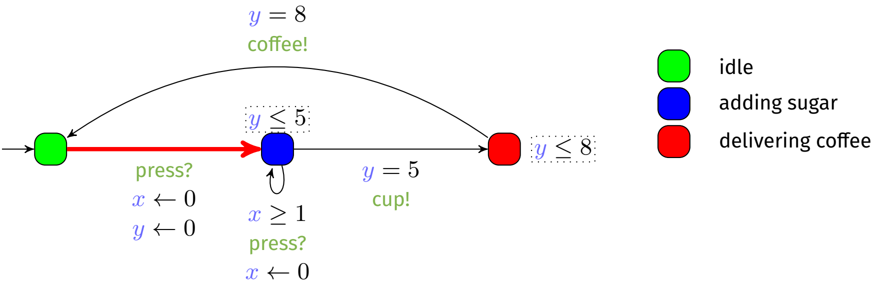
Examples of executions



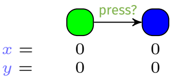
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



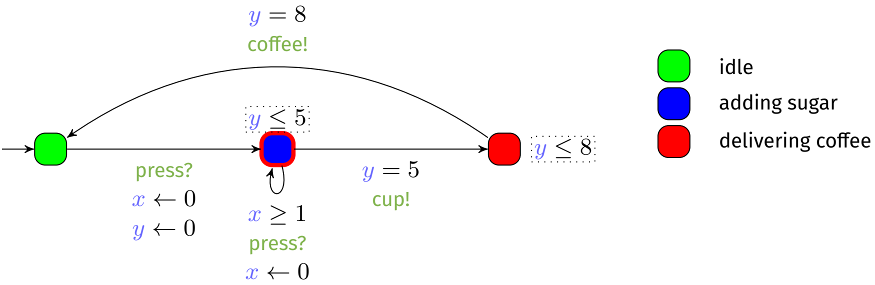
Examples of executions



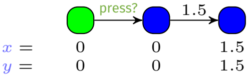
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



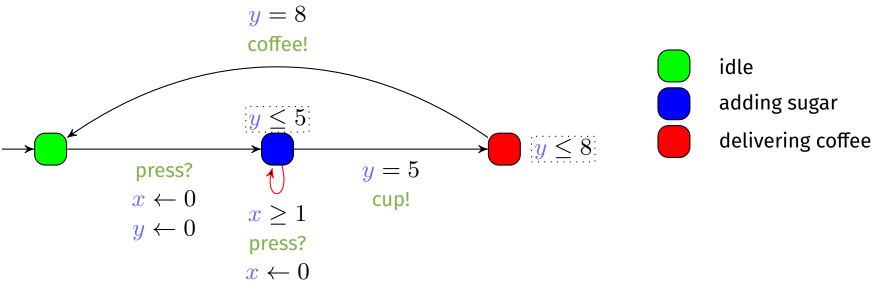
Examples of executions



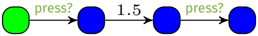
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



Examples of executions

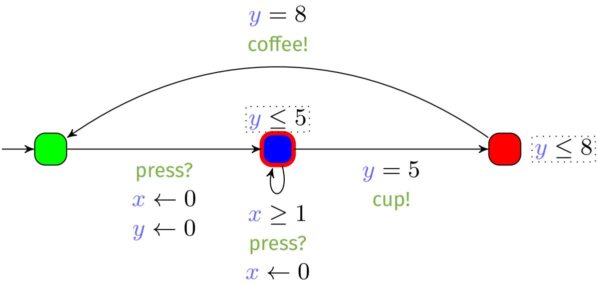


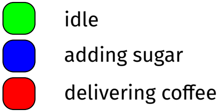
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



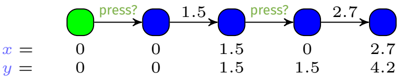
$x =$	0	0	1.5	0
$y =$	0	0	1.5	1.5

Examples of executions

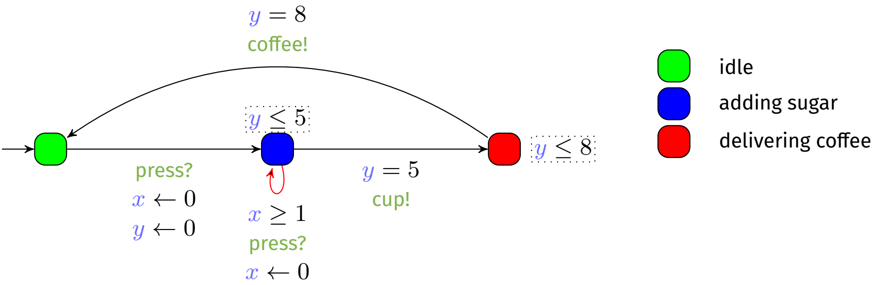




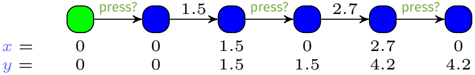
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



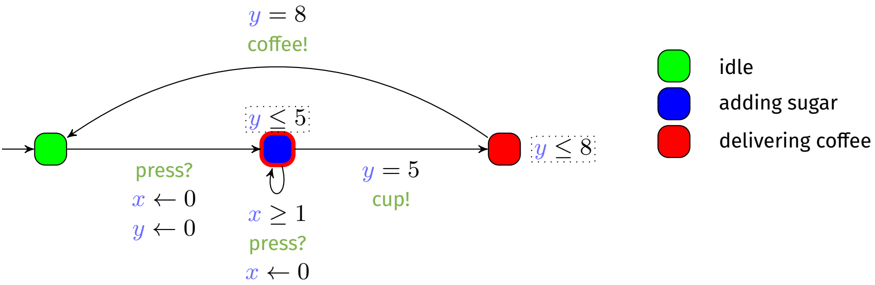
Examples of executions



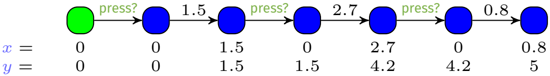
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



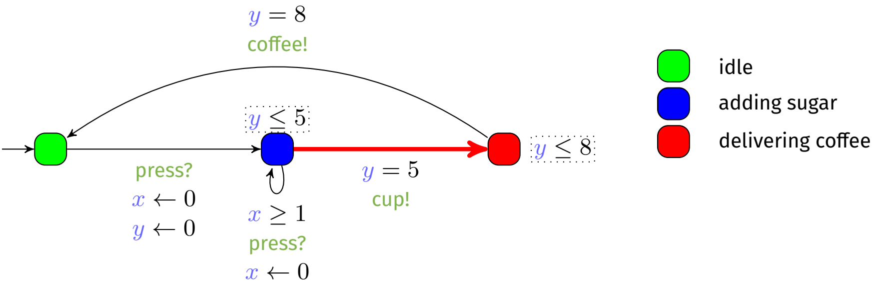
Examples of executions



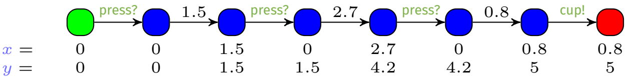
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



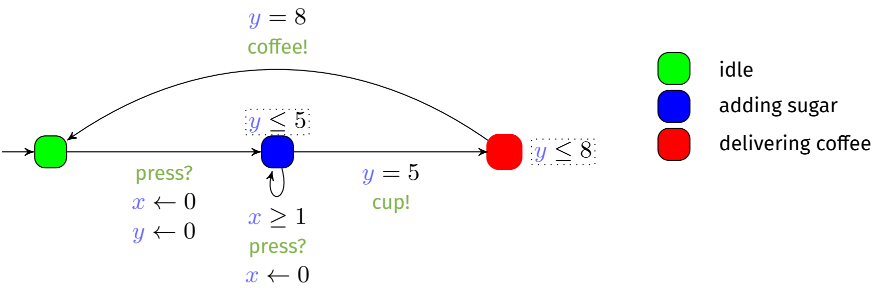
Examples of executions



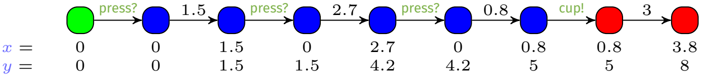
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



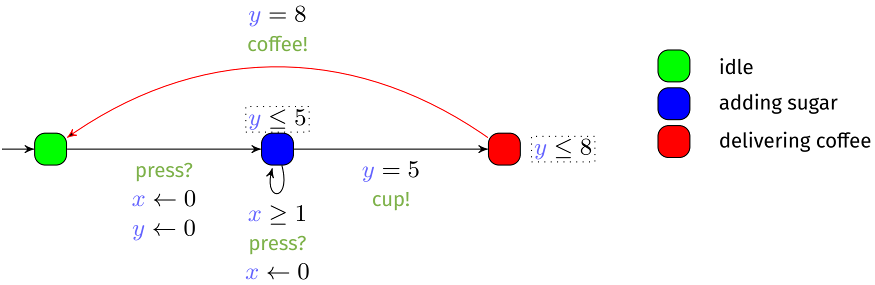
Examples of executions



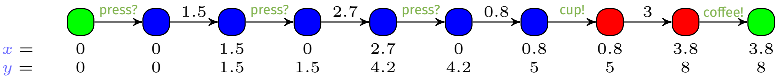
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



Examples of executions



- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar

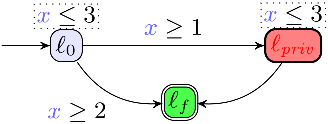


Outline

Formalization

Hypotheses:

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



Definition (execution-time opacity [And+22])

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

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

[And+22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. "Guaranteeing timed opacity using parametric timed model checking". In: *ACM Transactions on Software Engineering and Methodology* 31.4 (Oct. 2022), pp. 1–36

Weak and full ET-opacity

Definition (weak execution-time opacity)

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

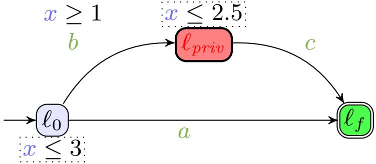
That is: private durations $\subseteq$ public durations

Definition (full execution-time opacity)

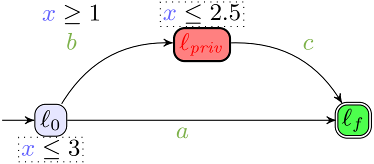
For each duration d,
There exists a run of duration d visiting ℓ_{priv}
$\Leftrightarrow$
There exists a run of duration d not visiting ℓ_{priv}

That is: private durations $=$ public durations

Illustrating weak and full execution-time opacity

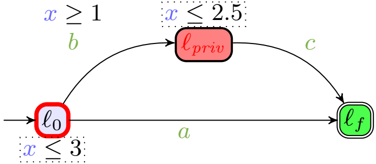


Illustrating weak and full execution-time opacity



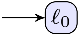
- There exist *(at least)* two runs of duration $d = 2$:

Illustrating weak and full execution-time opacity

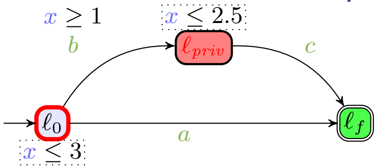


- There exist *(at least)* two runs of duration $d = 2$:

visiting ℓ_{priv}

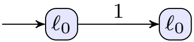


Illustrating weak and full execution-time opacity

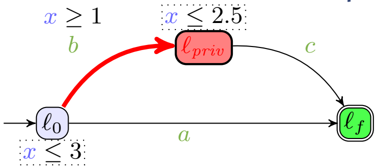


- There exist *(at least)* two runs of duration $d = 2$:

visiting ℓ_{priv}

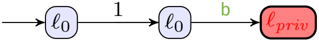


Illustrating weak and full execution-time opacity

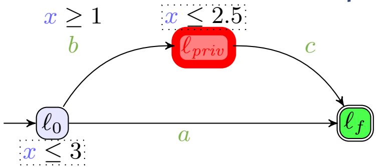


- There exist *(at least)* two runs of duration $d = 2$:

visiting ℓ_{priv}

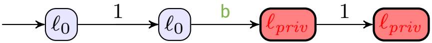


Illustrating weak and full execution-time opacity

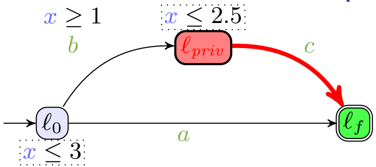


- There exist *(at least)* two runs of duration $d = 2$:

visiting ℓ_{priv}

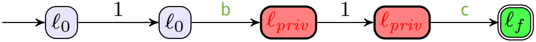


Illustrating weak and full execution-time opacity

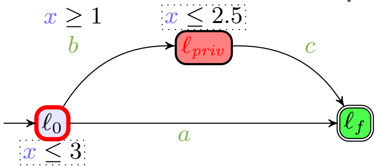


- There exist *(at least)* two runs of duration $d = 2$:

visiting ℓ_{priv}

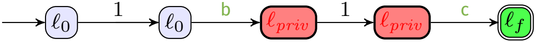


Illustrating weak and full execution-time opacity

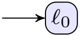


- There exist *(at least)* two runs of duration $d = 2$:

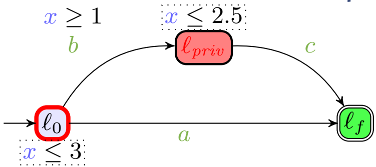
visiting ℓ_{priv}



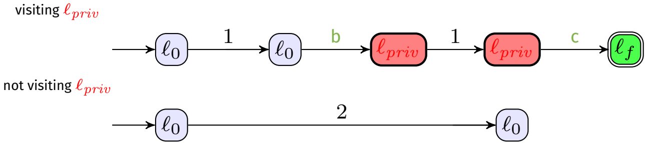
not visiting ℓ_{priv}



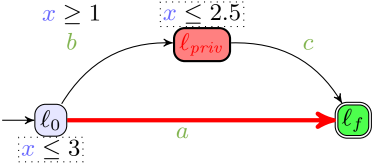
Illustrating weak and full execution-time opacity



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



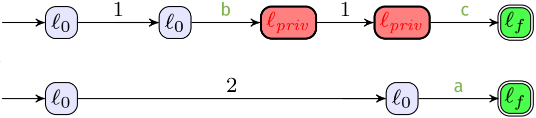
Illustrating weak and full execution-time opacity



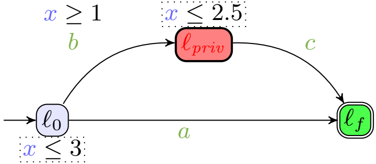
- There exist *(at least)* two runs of duration $d = 2$:

visiting ℓ_{priv}

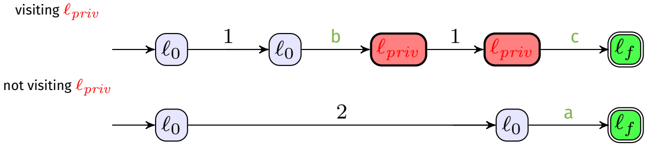
not visiting ℓ_{priv}



Illustrating weak and full execution-time opacity

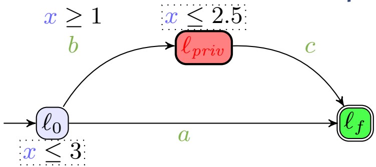


- There exist *(at least)* two runs of duration $d = 2$:

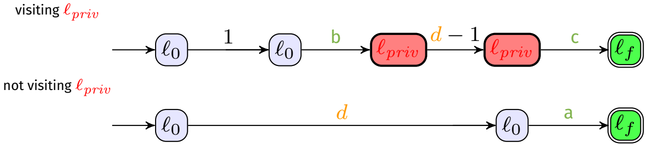


The system is ET-opaque for a duration $d = 2$

Illustrating weak and full execution-time opacity

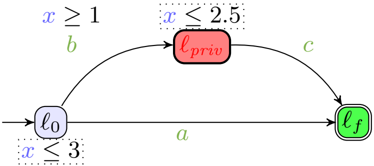


- There exist *(at least)* two runs of duration d for all durations $d \in [1, 2.5]$:



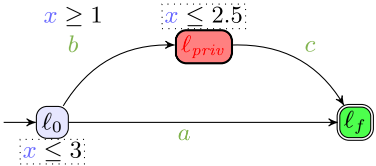
The system is ET-opaque for all durations in $[1, 2.5]$

Illustrating weak and full execution-time opacity



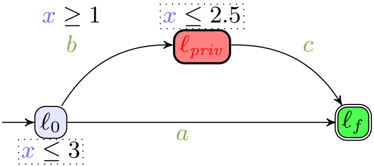
- There exist *(at least)* two runs of duration d for all durations $d \in [1, 2.5]$

Illustrating weak and full execution-time opacity



- There exist *(at least)* two runs of duration d for all durations $d \in [1, 2.5]$
- Generally:
  - private execution times are $[1, 2.5]$
    public execution times are $[0, 3]$

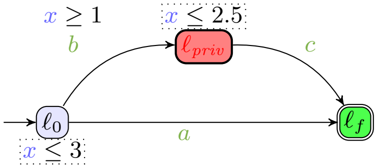
Illustrating weak and full execution-time opacity



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

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

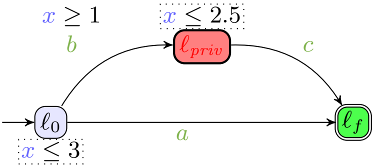
Illustrating weak and full execution-time opacity



- There exist *(at least)* two runs of duration d for all durations $d \in [1, 2.5]$
- Generally:
  - private execution times are $[1, 2.5]$
    public execution times are $[0, 3]$
  - private durations $\subseteq$ public durations

The system is weakly ET-opaque

Illustrating weak and full execution-time opacity

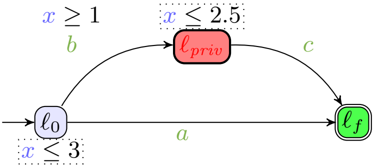


- There exist *(at least)* two runs of duration d for all durations $d \in [1, 2.5]$
- Generally:
  - private execution times are $[1, 2.5]$
    public execution times are $[0, 3]$
  - private durations $\subsetneq$ public durations

The system is weakly ET-opaque

  - private durations $\neq$ public durations

Illustrating weak and full execution-time opacity



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

Execution-time opacity computation can be achieved

Theorem (Computability of execution-time opacity)

The answer to the execution-time opacity computation problem for timed automata can be effectively computed in the form of a finite union of intervals

Proof: based on the region graph (see [And+22])

Exact complexity: unproved (EXPSPACE upper bound proved, but exponential hardness seems likely)

Remark: to be put in perspective with [Cas09]

- undecidability for a less expressive class, for a stronger notion of opacity

[And+22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. "Guaranteeing timed opacity using parametric timed model checking". In: *ACM Transactions on Software Engineering and Methodology* 31.4 (Oct. 2022), pp. 1–36

[Cas09] Franck Cassez. "The Dark Side of Timed Opacity". In: *ISA*. vol. 5576. LNCS. Springer, 2009, pp. 21–30

Full and weak execution-time opacity

Theorem (Full execution-time opacity [And+22])

Full execution-time opacity is decidable for timed automata

Theorem (Weak execution-time opacity [ALM23])

Weak execution-time opacity is decidable for timed automata

[And+22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. "Guaranteeing timed opacity using parametric timed model checking". In: *ACM Transactions on Software Engineering and Methodology* 31.4 (Oct. 2022), pp. 1–36

[ALM23] Étienne André, Engel Lefaucheux, and Dylan Marinho. "Expiring opacity problems in parametric timed automata". In: *ICECCS*. To appear. 2023

Outline

Towards configurable opaque systems…

Problems

- Can we configure some timing constants to guarantee opacity?
- Verification for one set of constants does not usually guarantee the correctness for other values
- Robustness [BMS13]: What happens if 50 is implemented with 49.99?

[BMS13] Patricia Bouyer, Nicolas Markey, and Ocan Sankur. "Robustness in timed automata". In: *RP*. vol. 8169. LNCS. Invited paper. Springer, Sept. 2013, pp. 1–18

Towards configurable opaque systems...

Problems

- Can we configure some timing constants to guarantee opacity?
- Verification for one set of constants does not usually guarantee the correctness for other values
- Robustness [BMS13]: What happens if 50 is implemented with 49.99?

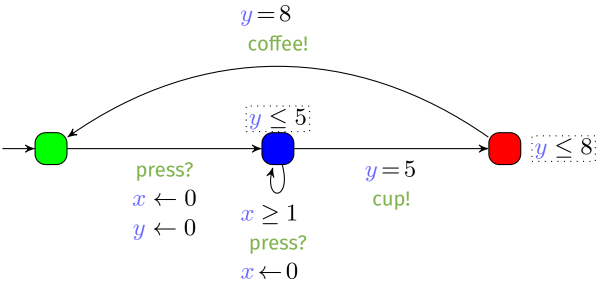
A solution:

- Parameter synthesis
  - Consider that timing constants are unknown constants (parameters)

[BMS13] Patricia Bouyer, Nicolas Markey, and Ocan Sankur. "Robustness in timed automata". In: *RP*. vol. 8169. LNCS. Invited paper. Springer, Sept. 2013, pp. 1–18

Parametric Timed Automaton (PTA)

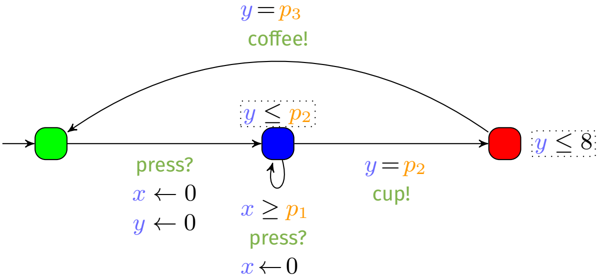
- Timed automaton (sets of locations, actions and clocks)



[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. “Parametric real-time reasoning”. In: *STOC*. ACM, 1993, pp. 592–601

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*. ACM, 1993, pp. 592–601

Two classes of parametric problems

Emptiness problem

Is the set of parameter valuations ensuring the property empty?

Synthesis problem

Synthesize all the parameter valuations ensuring the property

Two classes of parametric problems

Emptiness problem

Is the set of parameter valuations ensuring the property empty?

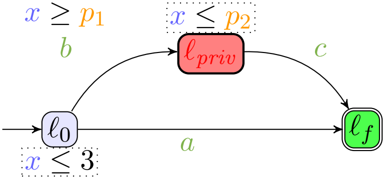
Synthesis problem

Synthesize all the parameter valuations ensuring the property

4 concrete opacity problems:

- Decision problems: weak (resp. full) execution-time opacity emptiness
- Synthesis problems: weak (resp. full) execution-time opacity synthesis

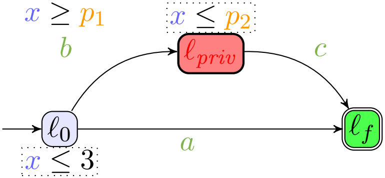
Example



Private	$[p_1, p_2]$
Public	$[0, 3]$

ET-opacity	Emptiness	Synthesis
weak		
full		

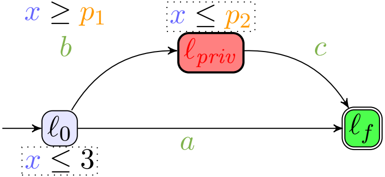
Example



Private	$[p_1, p_2]$
Public	$[0, 3]$

ET-opacity	Emptiness	Synthesis
weak	× ($\exists v$)	
full	× ($\exists v$)	

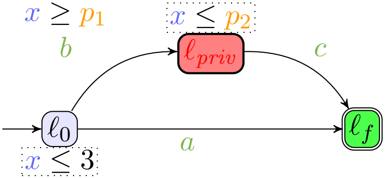
Example



Private	$[p_1, p_2]$
Public	$[0, 3]$

ET-opacity	Emptiness	Synthesis
weak	$\times$ ($\exists v$)	$0 \leq p_1 \wedge p_2 \leq 3 \quad \wedge \quad p_1 \leq p_2$
full	$\times$ ($\exists v$)	

Example



Private	$[p_1, p_2]$
Public	$[0, 3]$

ET-opacity	**Emptiness**	**Synthesis**
weak	$\times_{(\exists v)}$	$0 \leq p_1 \wedge p_2 \leq 3 \quad \wedge \quad p_1 \leq p_2$
full	$\times_{(\exists v)}$	$p_1 = 0 \wedge p_2 = 3$

These valuations give a way to configure the system parameters to formally guarantee execution-time opacity

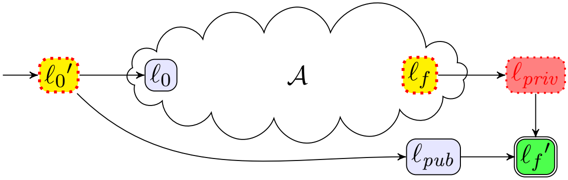
Outline

Execution-time opacity synthesis is (very) difficult

Theorem (Undecidability of execution-time opacity-emptiness)

The mere existence of a parameter valuation such that there exists a duration for which execution-time opacity is achieved is undecidable.

Proof idea: reduction from reachability-emptiness for PTAs [AHV93]



Remark: decidable subclass

(see [And+22])

[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. "Parametric real-time reasoning". In: *STOC*. ACM, 1993, pp. 592–601

[And+22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. "Guaranteeing timed opacity using parametric timed model checking". In: *ACM Transactions on Software Engineering and Methodology* 31.4 (Oct. 2022), pp. 1–36

Undecidability

Theorem (Full execution-time opacity emptiness [And+22])

Full execution-time opacity emptiness is undecidable for parametric timed automata, and even for the subclass of L/U parametric timed automata.

[And+22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. "Guaranteeing timed opacity using parametric timed model checking". In: *ACM Transactions on Software Engineering and Methodology* 31.4 (Oct. 2022), pp. 1–36

Undecidability

Theorem (Full execution-time opacity emptiness [And+22])

Full execution-time opacity emptiness is undecidable for parametric timed automata, and even for the subclass of L/U parametric timed automata.

In the following, we adopt a "best-effort" approach

- Approach not guaranteed to terminate in theory

[And+22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. "Guaranteeing timed opacity using parametric timed model checking". In: *ACM Transactions on Software Engineering and Methodology* 31.4 (Oct. 2022), pp. 1–36

Outline

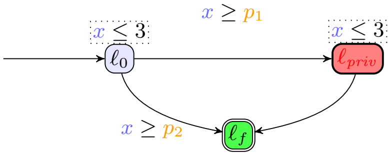
Computing execution-time opacity via reachability synthesis

Big picture:

- Formalism: parametric timed automata
- Our approach:
  1. Perform a (mild) transformation of the PTA
  2. Perform self-composition
  3. Apply parametric timed model checking (reachability-synthesis)
- Tool support: IMITATOR [And21]

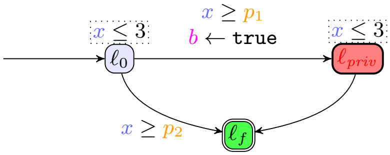
[And21] Étienne André. “IMITATOR 3: Synthesis of timing parameters beyond decidability”. In: *CAV*. vol. 12759. LNCS. Springer, 2021, pp. 1–14

Our transformation of the PTA in 4 overlays



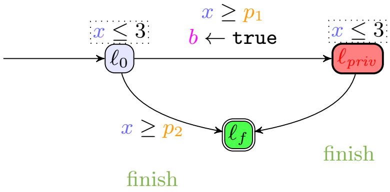
Our transformation of the PTA in 4 overlays

1. Add a Boolean flag b to remember whether ℓ_{priv} was visited



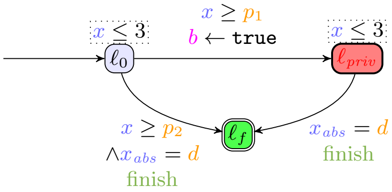
Our transformation of the PTA in 4 overlays

1. Add a Boolean flag b to remember whether ℓ_{priv} was visited
2. Add a synchronization action finish on any transition to ℓ_f



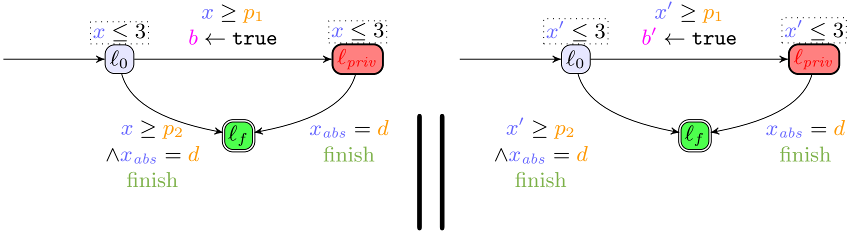
Our transformation of the PTA in 4 overlays

1. Add a Boolean flag b to remember whether ℓ_{priv} was visited
2. Add a synchronization action finish on any transition to ℓ_f
3. Measure the (parametric) duration to ℓ_f thanks to a new clock x_{abs} and a new parameter d



Our transformation of the PTA in 4 overlays

1. Add a Boolean flag b to remember whether ℓ_{priv} was visited
2. Add a synchronization action finish on any transition to ℓ_f
3. Measure the (parametric) duration to ℓ_f thanks to a new clock x_{abs} and a new parameter d
4. Perform self-composition (i. e., a synchronization on shared actions of the PTA with a copy of itself)



Applying reachability-synthesis

We then synthesize all parameter valuations (including d) for which the following discrete state is reachable:

- the original automaton is in ℓ_f with $b = \texttt{true}$
- the copy automaton is in ℓ_f with $b' = \texttt{false}$

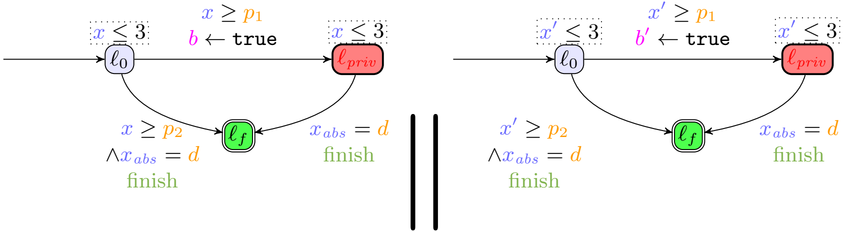
Applying reachability-synthesis

We then synthesize all parameter valuations (including d) for which the following discrete state is reachable:

- the original automaton is in ℓ_f with $b = \texttt{true}$
- the copy automaton is in ℓ_f with $b' = \texttt{false}$

Intuition:

- for the same duration (thanks to the synchronization on finish), we can reach ℓ_f "both" after visiting ℓ_{priv} (i. e., $b = \texttt{true}$) and not visiting ℓ_{priv} (i. e., $b = \texttt{false}$)



Formal proof of correctness: see paper

Outline

Experimental environment

Algorithms

1. Full execution-time opacity: "for a non-parametric TA, is the TA opaque for all execution times?"
2. Execution-time opacity synthesis: "for a PTA, synthesize some parameter valuations and execution times ensuring execution-time opacity"

Benchmarks

- Common PTA benchmarks [AMP21]
- Library of Java programs https://github.com/Apogee-Research/STAC/
  - Manually translated to PTAs
  - User-input variables translated to (non-timing) parameters (supported by IMITATOR)

See experiments at `doi.org/10.5281/zenodo.3251141`

and `imitator.fr/static/ATVA19/`

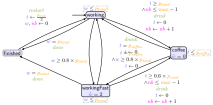
[AMP21] Étienne André, Dylan Marinho, and Jaco van de Pol. "A Benchmarks Library for Extended Timed Automata". In: *TAP*. vol. 12740. LNCS. Springer, 2021, pp. 39–50

Outline

Parameter synthesis using IMITATOR

IMITATOR: a parametric timed model checker

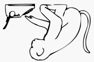
An extended PTA



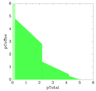
A property

```
#synth AGnot(loc[researcher] = coffee)
```

IMITATOR



Set of parameter valuations



Inputs

Output

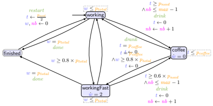
The set of parameter valuations is symbolic

- Symbolic: finite set of linear constraints (polyhedra)

Parameter synthesis using IMITATOR

IMITATOR: a parametric timed model checker

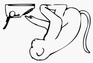
An extended PTA



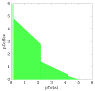
A property

```
#synth AGnot(loc[researcher] = coffee)
```

IMITATOR



Set of parameter valuations



Inputs

Output

The set of parameter valuations is symbolic

- Symbolic: finite set of linear constraints (polyhedra)
- Two categories of properties
  - Synthesis: "(try to) synthesize all valuations for which the property holds"
  - Exhibition: "(try to) synthesize at least one valuation for which the property holds"

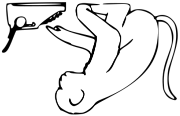
Distribution

Free and open source software: Available under the GNU-GPL license

Distribution:

- Binaries available for Linux platforms (no dependency, no install)
- Docker version
- Integrated as a virtual machine doi.org/10.5281/zenodo.4723415
- Comes with a user manual and an extensive benchmarks library [AMP21]

Try it!



www.imitator.fr

[AMP21] Étienne André, Dylan Marinho, and Jaco van de Pol. "A Benchmarks Library for Extended Timed Automata". In: *TAP*. vol. 12740. LNCS. Springer, 2021, pp. 39–50

Outline

Experiments: (non-parametric) execution-time opacity

Model			Transf. PTA			Result	
Name	$\lvert\mathcal{A}\rvert$	$\lvert X\rvert$	$\lvert\mathcal{A}\rvert$	$\lvert X\rvert$	$\lvert P\rvert$	Time (s)	Vulnerable?
Fig. 5, [VNN18]	1	1	2	3	3	0.02	$(\surd)$
Fig. 1b, [GMR07]	1	1	2	3	1	0.04	$(\surd)$
Fig. 2a,	1	1	2	3	1	0.05	$(\surd)$
Fig. 2b,	1	1	2	3	1	0.02	$(\surd)$
Web privacy problem [Ben+15]	1	2	2	4	1	0.07	$(\surd)$
`Coffee`	1	2	2	5	1	0.05	$\times$
`Fischer-HSRV02`	3	2	6	5	1	5.83	$(\surd)$
`STAC:1:n`			2	3	6	0.12	$(\surd)$
`STAC:1:v`			2	3	6	0.11	$\surd$
`STAC:3:n`			2	3	8	0.72	$\times$
`STAC:3:v`			2	3	8	0.74	$(\surd)$
`STAC:4:n`			2	3	8	6.40	$\surd$
`STAC:4:v`			2	3	8	265.52	$\surd$
`STAC:5:n`			2	3	6	0.24	$\times$
`STAC:11A:v`			2	3	8	47.77	$(\surd)$
`STAC:11B:v`			2	3	8	59.35	$(\surd)$
`STAC:12c:v`			2	3	8	18.44	$\surd$
`STAC:12e:n`			2	3	8	0.58	$\surd$
`STAC:12e:v`			2	3	8	1.10	$(\surd)$
`STAC:14:n`			2	3	8	22.34	$(\surd)$

$\times$ = not vulnerable;
$(\surd)$ = vulnerable, can be repaired;
$\surd$ = vulnerable, cannot be repaired

[VNN18] Panagiotis Vasilikos, Flemming Nielson, and Hanne Riis Nielson. “Secure Information Release in Timed Automata”. In: *POST*. vol. 10804. LNCS. Springer, 2018, pp. 28–52

[GMR07] Guillaume Gardey, John Mullins, and Olivier H. Roux. “Non-Interference Control Synthesis for Security Timed Automata”. In: *Electronic Notes in Theoretical Computer Science* 180.1 (2007), pp. 35–53

[Ben+15] Gilles Benattar, Franck Cassez, Didier Lime, and Olivier H. Roux. “Control and synthesis of non-interferent timed systems”. In: *International Journal of Control* 88.2 (2015), pp. 217–236

Outline

Experiments: (parametric) execution-time opacity synthesis

Model				Transf. PTA			Result	
Name	$\lvert\mathcal{A}\rvert$	$\lvert X\rvert$	$\lvert P\rvert$	$\lvert\mathcal{A}\rvert$	$\lvert X\rvert$	$\lvert P\rvert$	Time (s)	Constraint
Fig. 5, [VNN18]	1	1	0	2	3	4	0.02	K
Fig. 1b, [GMR07]	1	1	0	2	3	3	0.03	K
Fig. 2, [GMR07]	1	1	0	2	3	3	0.05	K
Web privacy problem [Ben+15]	1	2	2	2	4	3	0.07	K
`Coffee`	1	2	3	2	5	4	0.10	$\top$
`Fischer-HSRV02`	3	2	2	6	5	3	7.53	K
`STAC:3:v`			2	2	3	9	0.93	K

K = some valuations make the system non-vulnerable; $\top$ = all valuations make the system non-vulnerable

[VNN18] Panagiotis Vasilikos, Flemming Nielson, and Hanne Riis Nielson. “Secure Information Release in Timed Automata”. In: *POST*. vol. 10804. LNCS. Springer, 2018, pp. 28–52

[GMR07] Guillaume Gardey, John Mullins, and Olivier H. Roux. “Non-Interference Control Synthesis for Security Timed Automata”. In: *Electronic Notes in Theoretical Computer Science* 180.1 (2007), pp. 35–53

[GMR07] Guillaume Gardey, John Mullins, and Olivier H. Roux. “Non-Interference Control Synthesis for Security Timed Automata”. In: *Electronic Notes in Theoretical Computer Science* 180.1 (2007), pp. 35–53

[Ben+15] Gilles Benattar, Franck Cassez, Didier Lime, and Olivier H. Roux. “Control and synthesis of non-interferent timed systems”. In: *International Journal of Control* 88.2 (2015), pp. 217–236

Outline

What if the secret can expire?

Motivation: cache

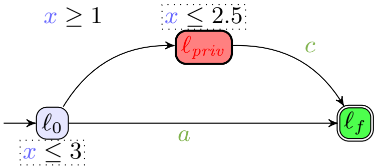
- Deducing that some information was in the cache a long time ago might be useless
- Opacity with an expiration date [Amm+21]

[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. ISSN: 2214-2126

Expiring execution-time opacity

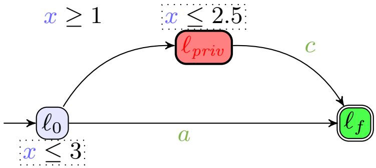
	Secret runs	**Non-secret runs**
ET-opacity	Runs visiting the private location (= private runs)	Runs not visiting the private location (= public runs)
expiring-ET-opacity	Private runs with ℓ_{priv} entered $\leq \Delta$ before the system completion	(i) Public runs and (ii) Private runs with ℓ_{priv} entered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

Example



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

Example



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

Some results [ALM23]

- ☺ Given Δ, we can decide whether a TA is weakly (resp. fully) ET-opaque
- ☺ We can synthesize all Δ for which a TA is weakly ET-opaque
- ☹ The synthesis of all Δ for full ET-opacity remains open
- ☹ The emptiness problems over parametric timed automata are undecidable
  - Even for the L/U-PTA subclass

[ALM23] Étienne André, Engel Lefaucheux, and Dylan Marinho. "Expiring opacity problems in parametric timed automata". In: *ICECCS*. To appear. 2023

Outline

Conclusion

Context: vulnerability by timing-attacks

- Attacker model: observability of the global execution time
- Goal: avoid leaking information on whether some discrete state has been visited

Several decision and computation problems studied for timed automata

- ☺ Mostly decidable

Extension to parametric timed automata

- ☹ Quickly undecidable
- ☺ One procedure for one synthesis problem
- Toolkit: IMITATOR
- Benchmarks: concurrent systems and Java programs

Perspectives

- Theoretical open problems
  - Synthesis of expiring dates for weak expiring opacity
  - Execution-time opacity emptiness remains open for 1 clock
  - Case of U-PTAs or L-PTAs [BL09]
- Algorithmic open problems
  - Weak (resp. full) execution-time opacity synthesis
- Automated translation of Java programs
  - Our translation required non-trivial creativity
  - How to automate it?
  - Finer grain needed for "untimed" instructions: probabilistic timings?
- Reconfiguring a non-opaque system
  - "From PTA parameter tuning back to the original system"
  - In programs: using `Wait` or `Sleep`?

[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

Bibliography

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.

[ALM23] Étienne André, Engel Lefaucheux, and Dylan Marinho. "Expiring opacity problems in parametric timed automata". In: *ICECCS* (June 12–16, 2023). Ed. by Yamine Ait-Ameur and Ferhat Khendek. To appear. Toulouse, France, 2023.

[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. ISSN: 2214-2126. DOI: 10.1016/j.jisa.2021.102926.

[AMP21] Étienne André, Dylan Marinho, and Jaco van de Pol. "A Benchmarks Library for Extended Timed Automata". In: *TAP* (June 21–25, 2021). Ed. by Frédéric Loulergue and Franz Wotawa. Vol. 12740. LNCS. virtual: Springer, 2021, pp. 39–50. DOI: 10.1007/978-3-030-79379-1_3.

[And+22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. "Guaranteeing timed opacity using parametric timed model checking". In: *ACM Transactions on Software Engineering and Methodology* 31.4 (Oct. 2022), pp. 1–36. DOI: 10.1145/3502851.

[And21] Étienne André. "IMITATOR 3: Synthesis of timing parameters beyond decidability". In: *CAV* (July 18–23, 2021). Ed. by Rustan Leino and Alexandra Silva. Vol. 12759. LNCS. virtual: Springer, 2021, pp. 1–14. DOI: 10.1007/978-3-030-81685-8_26.

References II

[Ben+15] Gilles Benattar, Franck Cassez, Didier Lime, and Olivier H. Roux. "Control and synthesis of non-interferent timed systems". In: *International Journal of Control* 88.2 (2015), pp. 217–236. DOI: 10.1080/00207179.2014.944356.

[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.

[BMS13] Patricia Bouyer, Nicolas Markey, and Ocan Sankur. "Robustness in timed automata". In: *RP* (Sept. 25–27, 2013). Ed. by Parosh Aziz Abdulla and Igor Potapov. Vol. 8169. LNCS. Invited paper. Uppsala, Sweden: Springer, Sept. 2013, pp. 1–18. DOI: 10.1007/978-3-642-41036-9_1.

[Cas09] Franck Cassez. "The Dark Side of Timed Opacity". In: *ISA* (June 25–27, 2009). Ed. by Jong Hyuk Park, Hsiao-Hwa Chen, Mohammed Atiquzzaman, Changhoon Lee, Tai-Hoon Kim, and Sang-Soo Yeo. Vol. 5576. LNCS. Seoul, Korea: Springer, 2009, pp. 21–30. DOI: 10.1007/978-3-642-02617-1_3.

[GMR07] Guillaume Gardey, John Mullins, and Olivier H. Roux. "Non-Interference Control Synthesis for Security Timed Automata". In: *Electronic Notes in Theoretical Computer Science* 180.1 (2007), pp. 35–53. DOI: 10.1016/j.entcs.2005.05.046.

[VNN18] Panagiotis Vasilikos, Flemming Nielson, and Hanne Riis Nielson. "Secure Information Release in Timed Automata". In: *POST* (Apr. 14–20, 2018). Ed. by Lujo Bauer and Ralf Küsters. Vol. 10804. LNCS. Thessaloniki, Greece: Springer, 2018, pp. 28–52. DOI: 10.1007/978-3-319-89722-6_2.

Licensing

Source of the graphics used I

Title: Piratey, vector version
Author: Gustavb
Source: `https://commons.wikimedia.org/wiki/File:Piratey,_vector_version.svg`
License: CC by-sa

Title: Expired
Author: RRZEicons
Source: `https://commons.wikimedia.org/wiki/File:Expired.svg`
License: CC by-sa

License of this document