Séminaire Ve, LoVe, LIPN

November 18th, 2022
Villetaneuse, France

strategFTO: Untimed control for timed opacity

Étienne André[1,2], Shapagat Bolat[2], 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, Nancy, France

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

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

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	i	c	k	e	n
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	i	c	k	e	n
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	i	c	k	e	n
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	i	c	k	e	n
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	i	c	k	e	n
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`

Informal problem

Question: can we exhibit secure execution times?

Timed-opacity computation

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

Outline

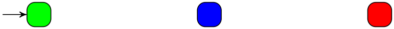
Preliminaries: Timed Opacity: Formalism and Preliminary results

Contribution: (Untimed) Control for timed opacity

Perspectives

Timed automaton (TA)

- Finite state automaton (sets of locations)



🟩 idle

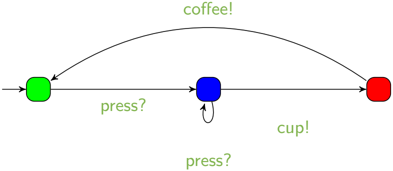
🟦 adding sugar

🟥 delivering coffee

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

Timed automaton (TA)

- Finite state automaton (sets of locations and actions)

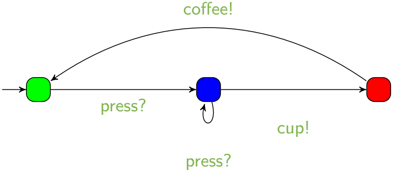


🟩 idle
🟦 adding sugar
🟥 delivering coffee

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



🟩 idle

🟦 adding sugar

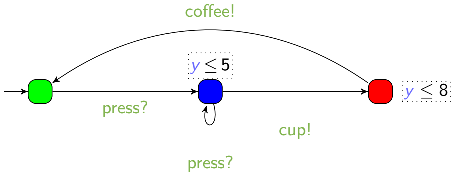
🟥 delivering coffee

[AD94] Rajeev Alur and David L. Dill. “A theory of timed automata”. In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235. ISSN: 0304-3975. 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



coffee!

$y \leq 5$

$y \leq 8$

press?

cup!

press?

🟩 idle

🟦 adding sugar

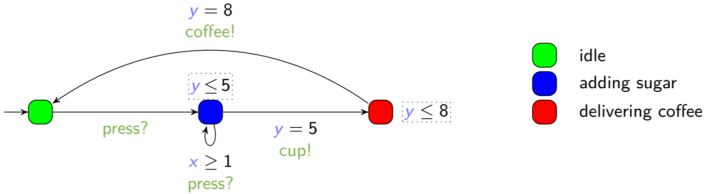
🟥 delivering coffee

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

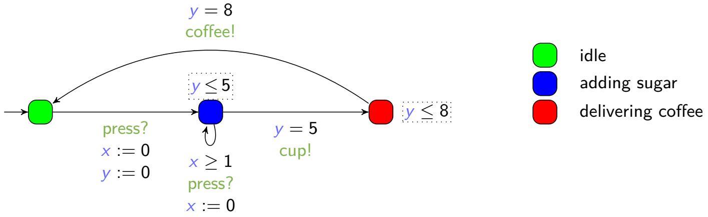


[AD94] Rajeev Alur and David L. Dill. “A theory of timed automata”. In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235. ISSN: 0304-3975. DOI: 10.1016/0304-3975(94)90010-8

Timed automaton (TA)

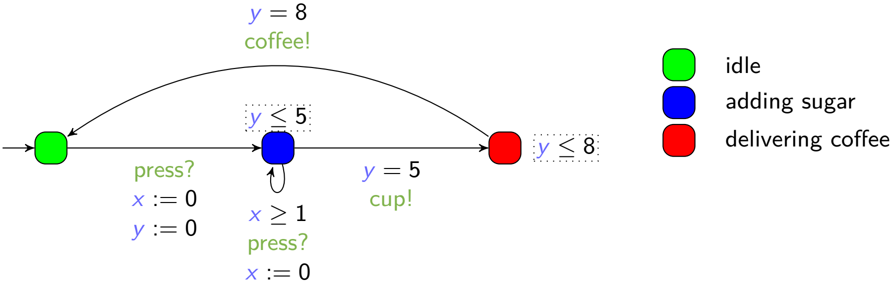
- Finite state automaton (sets of locations and actions) augmented with a set X of clocks [AD94]
  - Real-valued variables evolving linearly at the same rate
  - Can be compared to integer constants in invariants and guards

- 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

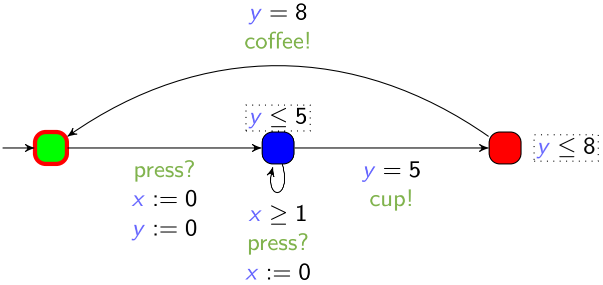


[AD94] Rajeev Alur and David L. Dill. “A theory of timed automata”. In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235. ISSN: 0304-3975. DOI: 10.1016/0304-3975(94)90010-8

The most critical system: The coffee machine



The most critical system: The coffee machine



🟩 idle

🟦 adding sugar

🟥 delivering coffee

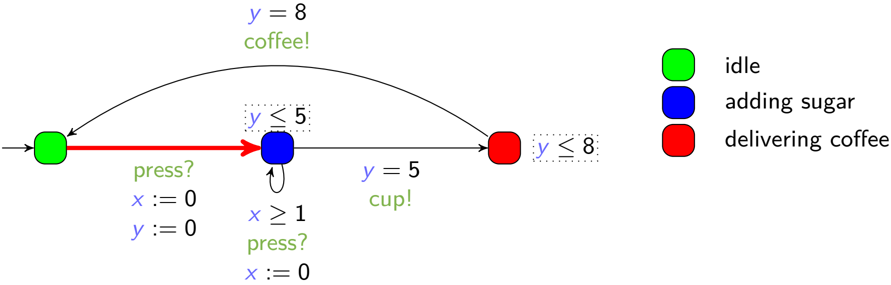
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar

🟩

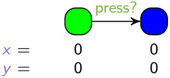
$x =$ 0

$y =$ 0

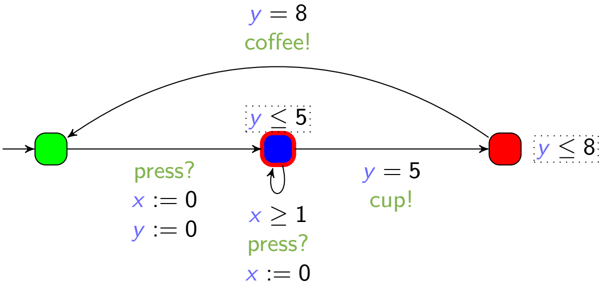
The most critical system: The coffee machine



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



The most critical system: The coffee machine

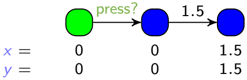


🟩 idle

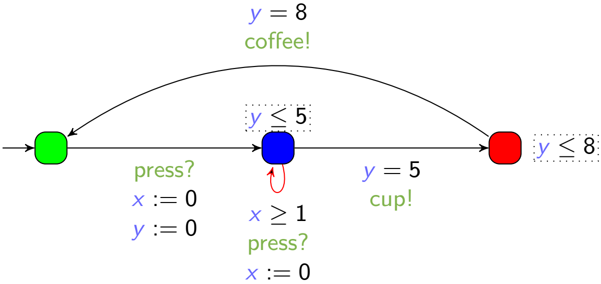
🟦 adding sugar

🟥 delivering coffee

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

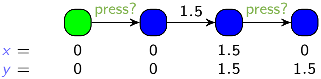


The most critical system: The coffee machine

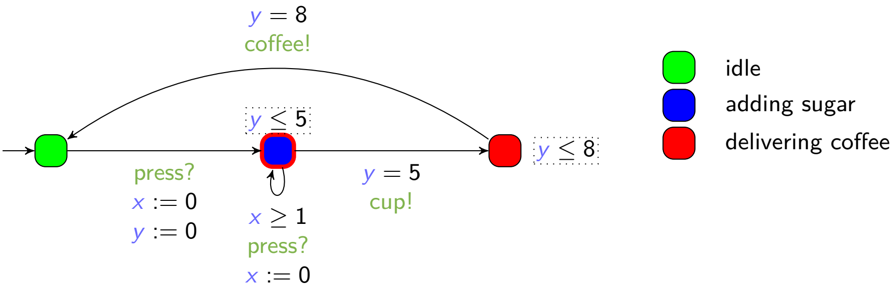


🟩 idle
🟦 adding sugar
🟥 delivering coffee

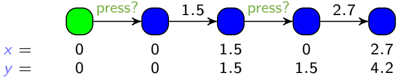
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



The most critical system: The coffee machine

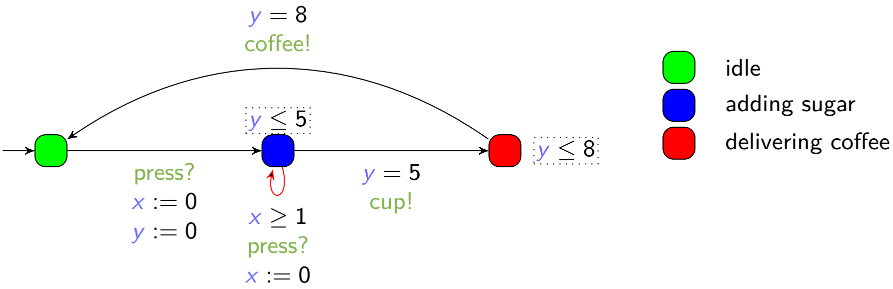


- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar

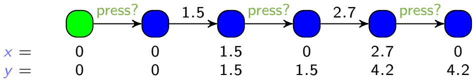


$x =$	0	0	1.5	0	2.7
$y =$	0	0	1.5	1.5	4.2

The most critical system: The coffee machine

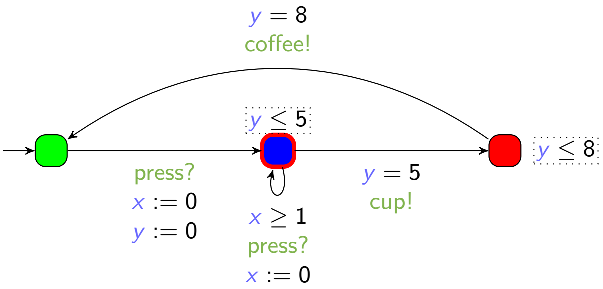


- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



$x =$	0	0	1.5	0	2.7	0
$y =$	0	0	1.5	1.5	4.2	4.2

The most critical system: The coffee machine

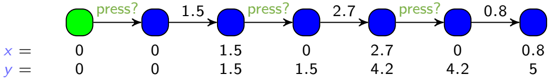


🟩 idle

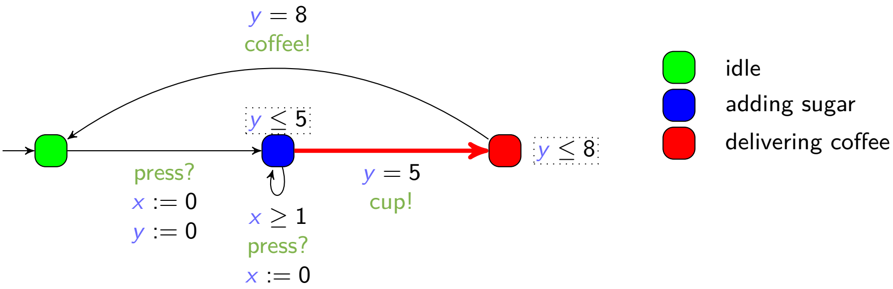
🟦 adding sugar

🟥 delivering coffee

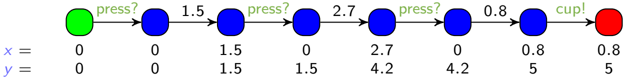
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



The most critical system: The coffee machine

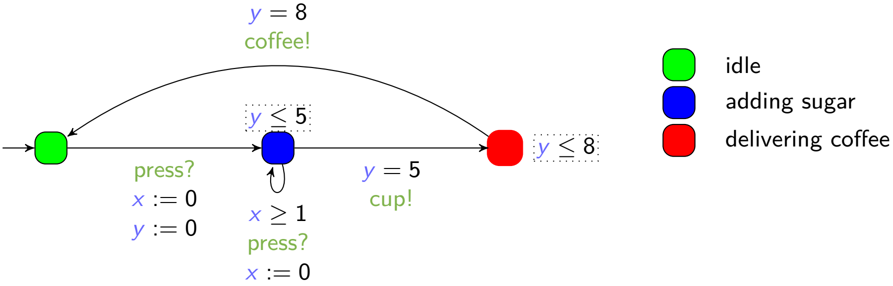


- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar

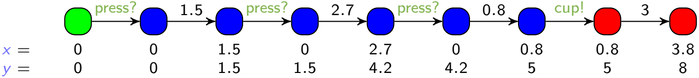


$x =$	0	0	1.5	0	2.7	0	0.8	0.8
$y =$	0	0	1.5	1.5	4.2	4.2	5	5

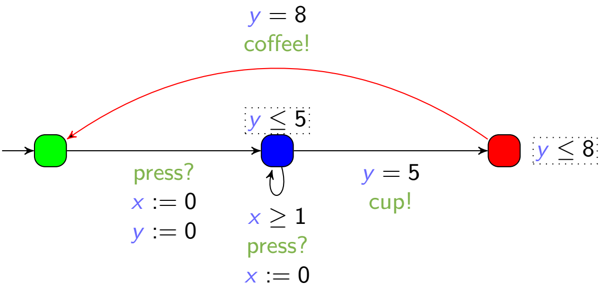
The most critical system: The coffee machine



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


press?
1.5
press?
2.7
press?
0.8
cup!
3
$x =$ 0 0 1.5 0 2.7 0 0.8 0.8 3.8
$y =$ 0 0 1.5 1.5 4.2 4.2 5 5 8

The most critical system: The coffee machine

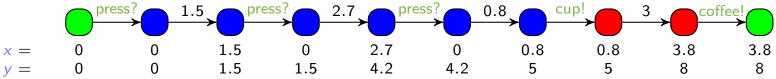


🟩 idle

🟦 adding sugar

🟥 delivering coffee

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



press?
1.5
press?
2.7
press?
0.8
cup!
3
coffee!
$x =$ 0 0 1.5 0 2.7 0 0.8 0.8 3.8 3.8
$y =$ 0 0 1.5 1.5 4.2 4.2 5 5 8 8

Outline

Preliminaries: Timed Opacity: Formalism and Preliminary results

- Timed Opacity formalization
- Computation problem and results

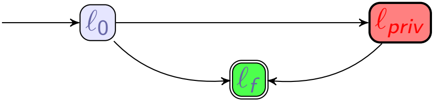
Contribution: (Untimed) Control for timed opacity

Perspectives

Formalization

Hypotheses: [AS19]

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



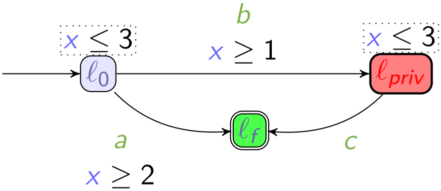
Definition (timed opacity)

The system is timed-opaque w.r.t. ℓ_{priv} on the way to ℓ_f for a duration d if there exist at least two runs to ℓ_f of duration d

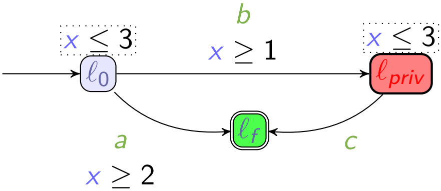
1. one passing by ℓ_{priv}
2. one *not* passing by ℓ_{priv}

[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

Example

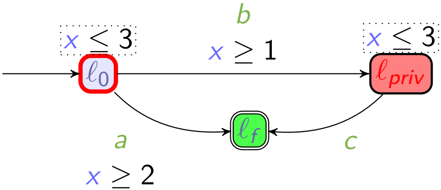


Example



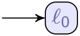
- There exist (at least) two runs of duration $d = 2$:

Example

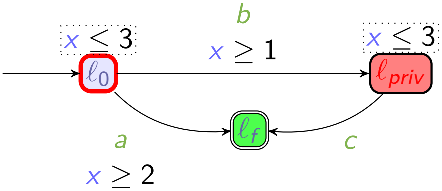


- There exist (at least) two runs of duration $d = 2$:

visiting ℓ_{priv}

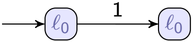


Example

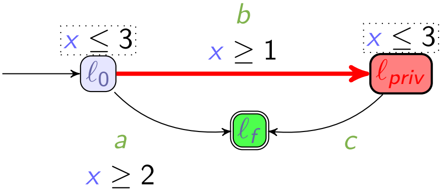


- There exist (at least) two runs of duration $d = 2$:

visiting ℓ_{priv}

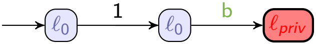


Example

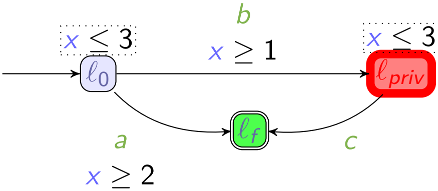


- There exist (at least) two runs of duration $d = 2$:

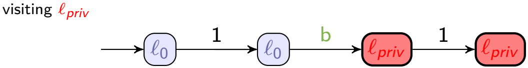
visiting ℓ_{priv}



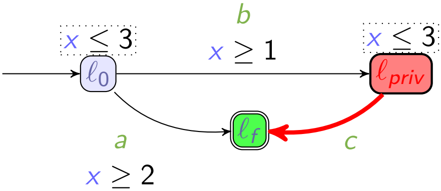
Example



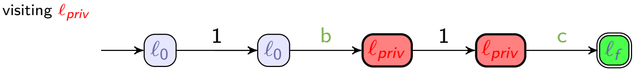
- There exist (at least) two runs of duration $d = 2$:



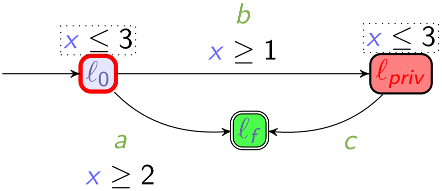
Example



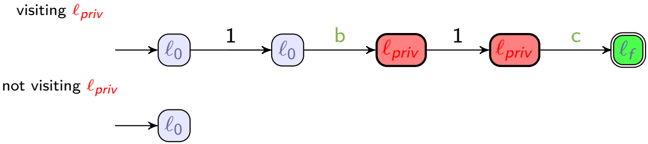
- There exist (at least) two runs of duration $d = 2$:



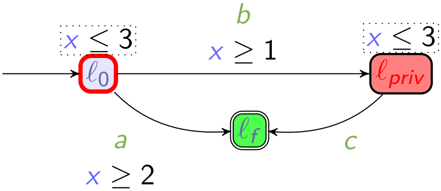
Example



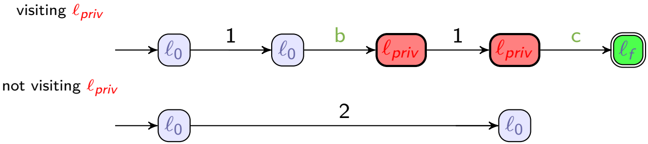
- There exist (at least) two runs of duration $d = 2$:



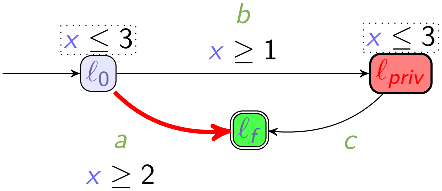
Example



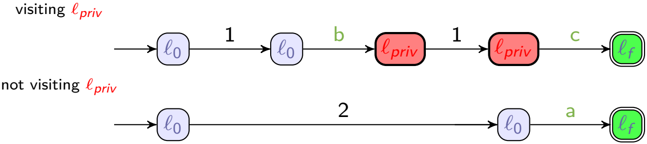
- There exist (at least) two runs of duration $d = 2$:



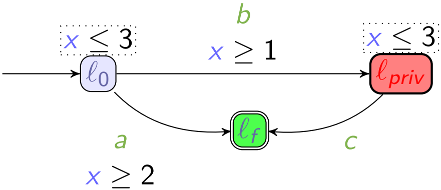
Example



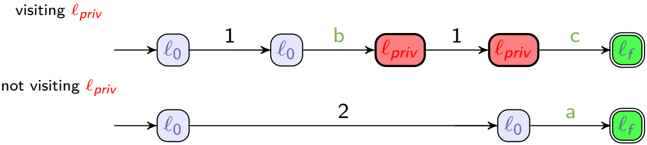
- There exist (at least) two runs of duration $d = 2$:



Example

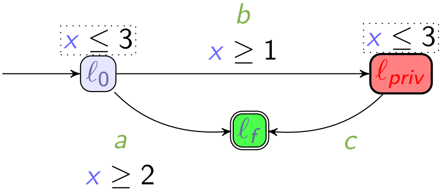


- There exist (at least) two runs of duration $d = 2$:

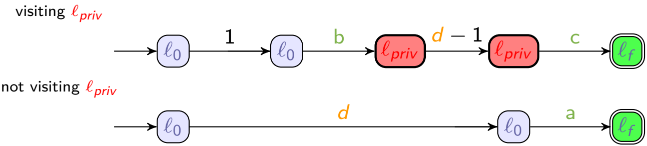


We say that the system is timed-opaque w.r.t. ℓ_{priv} on the way to ℓ_f for a duration $d = 2$

Example

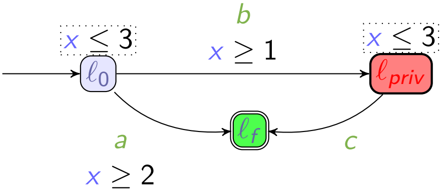


- There exist (at least) two runs of duration d for all durations $d \in [2, 3]$:



We say that the system is timed-opaque w.r.t. ℓ_{priv} on the way to ℓ_f for all durations in [2,3]

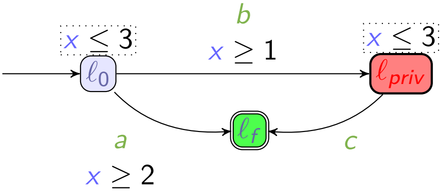
Example



- There exist (at least) two runs of duration d for all durations $d \in [2, 3]$:

We say that the system is timed-opaque w.r.t. ℓ_{priv} on the way to ℓ_f for all durations in [2,3]

Example

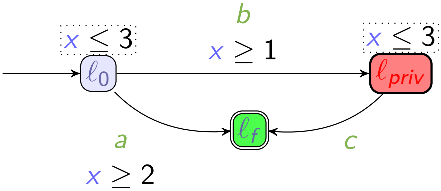


- There exist (at least) two runs of duration d for all durations $d \in [2, 3]$:

We say that the system is timed-opaque w.r.t. ℓ_{priv} on the way to ℓ_f for all durations in [2,3]

- But

Example

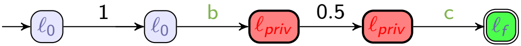


- There exist (at least) two runs of duration d for all durations $d \in [2, 3]$:

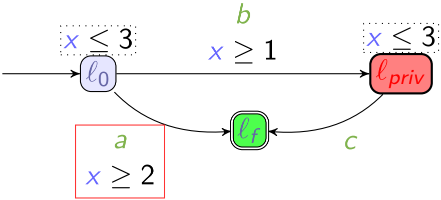
We say that the system is timed-opaque w.r.t. ℓ_{priv} on the way to ℓ_f for all durations in [2,3]

- But

  There exists a run of duration 1.5 reaching ℓ_f and visiting ℓ_{priv}



Example

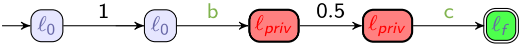


- There exist (at least) two runs of duration d for all durations $d \in [2, 3]$:

We say that the system is timed-opaque w.r.t. ℓ_{priv} on the way to ℓ_f for all durations in [2,3]

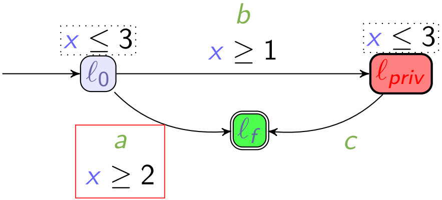
- But

  There exists a run of duration 1.5 reaching ℓ_f and visiting ℓ_{priv}



  There exists no run of duration 1.5 reaching ℓ_f and *not* visiting ℓ_{priv}

Example

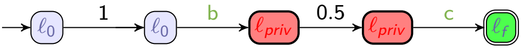


- There exist (at least) two runs of duration d for all durations $d \in [2, 3]$:

We say that the system is timed-opaque w.r.t. ℓ_{priv} on the way to ℓ_f for all durations in [2,3]

- But

There exists a run of duration 1.5 reaching ℓ_f and visiting ℓ_{priv}



There exists no run of duration 1.5 reaching ℓ_f and *not* visiting ℓ_{priv}

We say that the system is *not* fully timed-opaque w.r.t. ℓ_{priv} on the way to ℓ_f

Outline

Preliminaries: Timed Opacity: Formalism and Preliminary results

Timed Opacity formalization

Computation problem and results

Contribution: (Untimed) Control for timed opacity

Perspectives

Problem: timed-opacity computation

[TOSEM22]

Timed-opacity computation problem

Find durations d (“execution times”) of runs from ℓ_0 to ℓ_f such that the system is timed-opaque w.r.t. ℓ_{priv} on the way to ℓ_f

Theorem The durations d such that the system is timed-opaque can be effectively computed and defined

[Wei99] Volker Weispfenning. “Mixed Real-Integer Linear Quantifier Elimination”. In: *ISSAC* (July 29–31, 1999). Ed. by Keith O. Geddes, Bruno Salvy, and Samuel S. Dooley. Vancouver, BC, Canada: Association for Computing Machinery, 1999, pp. 129–136. DOI: 10.1145/309831.309888

[TOSEM22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. “Guaranteeing Timed Opacity Using Parametric Timed Model Checking”. In: *ACM Trans. Softw. Eng. Methodol.* (Nov. 2022). ISSN: 1049-331X. DOI: 10.1145/3502851

Problem: timed-opacity computation

[TOSEM22]

Timed-opacity computation problem

Find durations d ("execution times") of runs from ℓ_0 to ℓ_f such that the system is timed-opaque w.r.t. ℓ_{priv} on the way to ℓ_f

Theorem The durations d such that the system is timed-opaque can be effectively computed and defined

Corollary Asking if a TA is timed-opaque for all its execution times is decidable

[Wei99] Volker Weispfenning. "Mixed Real-Integer Linear Quantifier Elimination". In: *ISSAC* (July 29–31, 1999). Ed. by Keith O. Geddes, Bruno Salvy, and Samuel S. Dooley. Vancouver, BC, Canada: Association for Computing Machinery, 1999, pp. 129–136. DOI: 10.1145/309831.309888

[TOSEM22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. "Guaranteeing Timed Opacity Using Parametric Timed Model Checking". In: *ACM Trans. Softw. Eng. Methodol.* (Nov. 2022). ISSN: 1049-331X. DOI: 10.1145/3502851

Problem: timed-opacity computation

[TOSEM22]

Timed-opacity computation problem

Find durations d ("execution times") of runs from ℓ_0 to ℓ_f such that the system is timed-opaque w.r.t. ℓ_{priv} on the way to ℓ_f

Theorem The durations d such that the system is timed-opaque can be effectively computed and defined

Corollary Asking if a TA is timed-opaque for all its execution times is decidable

Proof: based on the region graph and RA-arithmetic [Wei99]

[Wei99] Volker Weispfenning. "Mixed Real-Integer Linear Quantifier Elimination". In: *ISSAC* (July 29–31, 1999). Ed. by Keith O. Geddes, Bruno Salvy, and Samuel S. Dooley. Vancouver, BC, Canada: Association for Computing Machinery, 1999, pp. 129–136. DOI: 10.1145/309831.309888

[TOSEM22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. "Guaranteeing Timed Opacity Using Parametric Timed Model Checking". In: *ACM Trans. Softw. Eng. Methodol.* (Nov. 2022). ISSN: 1049-331X. DOI: 10.1145/3502851

Outline

Preliminaries: Timed Opacity: Formalism and Preliminary results

Contribution: (Untimed) Control for timed opacity

Perspectives

Context & Informal problem

[And+22]

- √ We can decide computation and decision problems for timed opacity
- × What to do if the model is not (fully) timed-opaque?

[And+22] Étienne André, Shapagat Bolat, Engel Lefaucheux, and Dylan Marinho. "strategFTO: Untimed control for timed opacity". In: *Formal Techniques for Safety-Critical Systems - FTSCS 2022, Auckland, New Zeland, December 5-10, 2022, Proceedings*. Ed. by Cyrille Artho and Peter Ölveczky. 2022

Context & Informal problem [And+22]

- √ We can decide computation and decision problems for timed opacity
- × What to do if the model is not (fully) timed-opaque?

Full timed opacity control

Is it possible to disable some user actions to make the system fully timed-opaque?

[And+22] Étienne André, Shapagat Bolat, Engel Lefaucheux, and Dylan Marinho. "strategFTO: Untimed control for timed opacity". In: *Formal Techniques for Safety-Critical Systems - FTSCS 2022, Auckland, New Zeland, December 5-10, 2022, Proceedings*. Ed. by Cyrille Artho and Peter Ölveczky. 2022

Untimed control

Goal

Exhibit a controller guaranteeing the system to be fully timed-opaque

i. e., a subset of the actions to be kept, while other controllable actions are disabled

Untimed control

Goal

Exhibit a controller guaranteeing the system to be fully timed-opaque

i. e., a subset of the actions to be kept, while other controllable actions are disabled

We distinguish two kinds of actions:

- uncontrollable: required by the system or dependent on another agent

  → e. g., action dealing with a correct or incorrect password
- controllable: that can be disabled

Outline

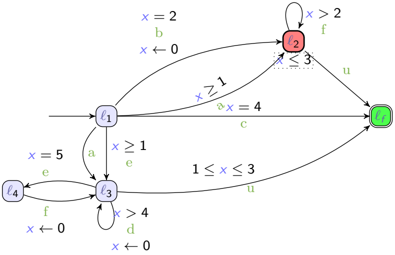
Preliminaries: Timed Opacity: Formalism and Preliminary results

Contribution: (Untimed) Control for timed opacity

- A running example
- Our tool
- Proof of concept

Perspectives

A running example



Uncontrollable u

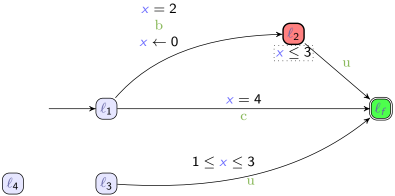
Controllable a, b, c, d, e, f

Is the system fully timed-opaque?

- Passing by ℓ_2: $[1, 5]$
- Not passing by ℓ_2: $[1, 3] \cup [4, 4] \cup [5, +\text{inf})$

$\Rightarrow$ Not fully timed-opaque

A running example



Uncontrollable u

Controllable a, b, c, d, e, f

Allowed u + b, c

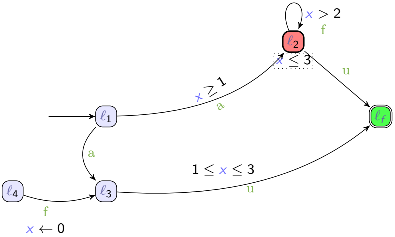
Disabled a, d, e, f

Is the system fully timed-opaque?

- Passing by ℓ_2: $[2, 5]$
- Not passing by ℓ_2: $[4, 4]$

$\Rightarrow$ Not fully timed-opaque

A running example



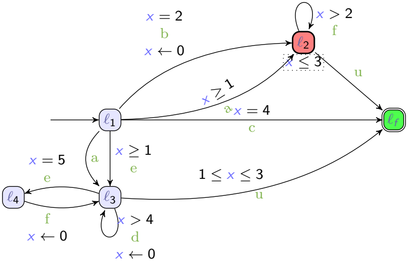
Uncontrollable	u
Controllable	a, b, c, d, e, f
Allowed	u + a, f
Disabled	b, c, d, e

Is the system fully timed-opaque?

- Passing by ℓ_2: $[1, 3]$
- Not passing by ℓ_2: $[1, 3]$

$\Rightarrow$ Fully timed-opaque

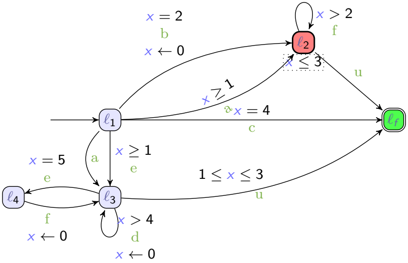
A running example



It can be shown that the set of *sets of actions to allow* is

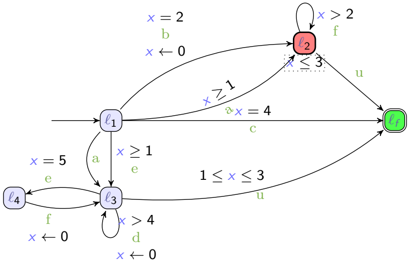
$\{u, a\}$ $\{u, a, e\}$ $\{u, a, f\}$

A running example



It can be shown that the set of fully timed-opaque strategies is
$\{u, a\}$ $\{u, a, e\}$ $\{u, a, f\}$

A running example



It can be shown that the set of fully timed-opaque strategies is

$$\underbrace{\{u, a\}}_{\text{minimal}} \qquad \underbrace{\{u, a, e\} \qquad \{u, a, f\}}_{\text{maximal}}$$

Outline

Preliminaries: Timed Opacity: Formalism and Preliminary results

Contribution: (Untimed) Control for timed opacity

- A running example
- Our tool
- Proof of concept

Perspectives

strategFTO

- an automated open-source tool written in Java

  https://github.com/DylanMarinho/Controlling-TA

- iteratively constructs strategies
  - computes the private and public execution times (using IMITATOR[And21])
  - checks full timed opacity by checking their equality (using PolyOp[1])
    - Method: by considering execution times as a timing parameter, and performing parameter synthesis

[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. Lecture Notes in Computer Science. virtual: Springer, 2021, pp. 1–14. DOI: 10.1007/978-3-030-81685-8_26

[1] https://github.com/etienneandre/PolyOp

Outline

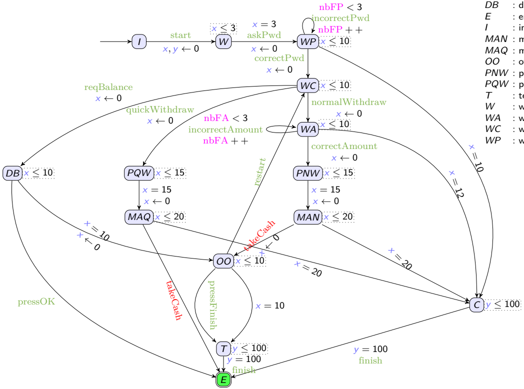
Preliminaries: Timed Opacity: Formalism and Preliminary results

Contribution: (Untimed) Control for timed opacity

- A running example
- Our tool
- Proof of concept

Perspectives

A Proof of concept benchmark: an ATM



C	: cancelling
DB	: displaying balance
E	: end
I	: initial
MAN	: money available normal
MAQ	: money available quick
OO	: other operations
PNW	: preparing normal withdrawal
PQW	: preparing quick withdrawal
T	: terminating
W	: waiting
WA	: waiting for amount
WC	: waiting for choice
WP	: waiting for password

Uncontrollable actions	correctAmount, correctPwd, incorrectAmount, incorrectPwd, pressFinish
Controllable system actions	askPwd, finish, start
Controllable user actions	reqBalance, normalWithdraw, pressOK, quickWithdraw, restart
Secret	takeCash

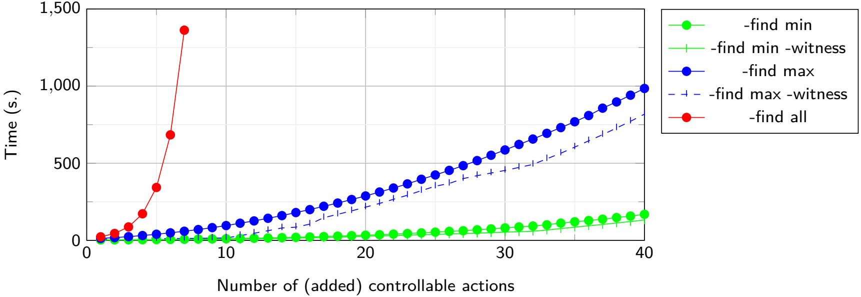
Proof of concept

[And+22]

Actions to disable **Option**	synthMinControl `-find min`	witnessMinControl `-find min -witness`	synthMaxControl `-find max`	witnessMaxControl `-find max -witness`	synthControl `-find all`
restart, pressOK			√	√	√
restart, reqBalance			√		√
restart, pressOK, quickWithdraw					√
restart, pressOK, reqBalance					√
restart, quickWithdraw, reqBalance					√
restart, pressOK, quickWithdraw, reqBalance	√	√			√

[And+22] Étienne André, Shapagat Bolat, Engel Lefaucheux, and Dylan Marinho. "strategFTO: Untimed control for timed opacity". In: *Formal Techniques for Safety-Critical Systems - FTSCS 2022, Auckland, New Zeland, December 5-10, 2022, Proceedings*. Ed. by Cyrille Artho and Peter Ölveczky. 2022

Scalability



Methodology: add to the ATM model an increasing number of self-loop transitions

Outline

Preliminaries: Timed Opacity: Formalism and Preliminary results

Contribution: (Untimed) Control for timed opacity

Perspectives

Perspectives

Theory

- Use symbolic reasoning

  → Instead of a simple enumeration

- Extend the method to timed control

Perspectives

Theory

- Use symbolic reasoning

  → Instead of a simple enumeration

- Extend the method to timed control

Algorithmic and implementation

- Automatic translation of programs to timed automata
- Repairing a non timed-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. ISSN: 0304-3975. DOI: 10.1016/0304-3975(94)90010-8.

[And+22] Étienne André, Shapagat Bolat, Engel Lefaucheux, and Dylan Marinho. “strategFTO: Untimed control for timed opacity”. In: *Formal Techniques for Safety-Critical Systems - FTSCS 2022, Auckland, New Zeland, December 5-10, 2022, Proceedings*. Ed. by Cyrille Artho and Peter Ölveczky. 2022.

References II

[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. Lecture Notes in Computer Science. virtual: Springer, 2021, pp. 1–14. DOI: 10.1007/978-3-030-81685-8_26.

[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.

References III

[TOSEM22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. “Guaranteeing Timed Opacity Using Parametric Timed Model Checking”. In: *ACM Trans. Softw. Eng. Methodol.* (Nov. 2022). ISSN: 1049-331X. DOI: 10.1145/3502851.

[Wei99] Volker Weispfenning. “Mixed Real-Integer Linear Quantifier Elimination”. In: *ISSAC* (July 29–31, 1999). Ed. by Keith O. Geddes, Bruno Salvy, and Samuel S. Dooley. Vancouver, BC, Canada: Association for Computing Machinery, 1999, pp. 129–136. DOI: 10.1145/309831.309888.