AFADL 2022

June 9, 2022
Vannes

Guaranteeing Timed Opacity using Parametric Timed Model Checking

Étienne André[1], Didier Lime[2], Dylan Marinho[1] and Sun Jun[3]

[1] Université de Lorraine, CNRS, Inria, LORIA, Nancy, France
[2] École Centrale de Nantes, LS2N, UMR CNRS 6004, Nantes, France

[3] School of Information Systems, Singapore Management University, Singapore

Paper accepted at ACM Transactions on Software Engineering and Methodology (TOSEM)
Supported by the ANR-NRF French-Singaporean research program ProMiS (ANR-19-CE25-0015)

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

```
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	i	c	k	e	n
attempt	c	h	e	e	s	e	

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	i	c	k	e	n
attempt	c	h	e	e	s	e	

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	i	c	k	e	n
attempt	c	h	e	e	s	e	

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	i	c	k	e	n
attempt	c	h	e	e	s	e	

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	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 problems

Question: can we exhibit secure execution times?

Time-opacity computation

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

Informal problems

Question: can we exhibit secure execution times?

Time-opacity computation

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

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

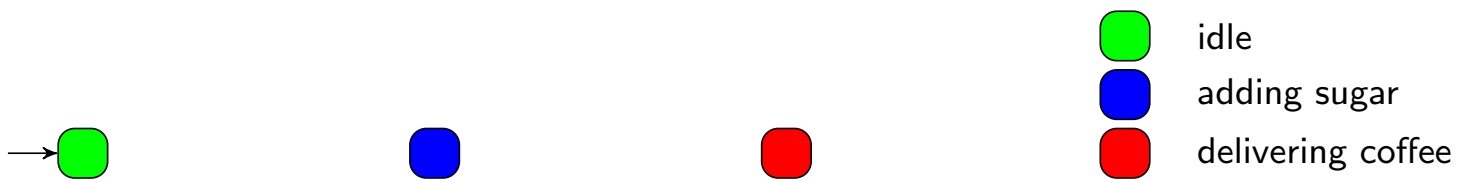
Time-opacity synthesis

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

Outline

Timed automaton (TA)

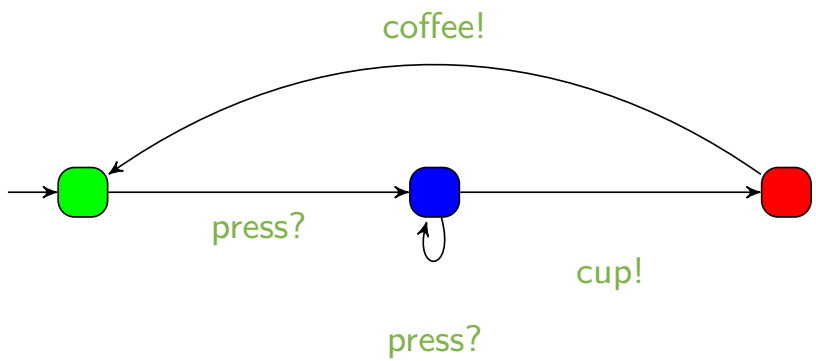
- Finite state automaton (sets of locations)

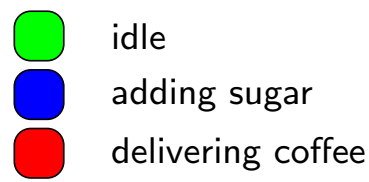


[AD94] Rajeev Alur and David L. Dill. "A theory of timed automata". In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235. ISSN: 0304-3975. DOI: 10.1016/0304-3975(94)90010-8

Timed automaton (TA)

- Finite state automaton (sets of locations and actions)

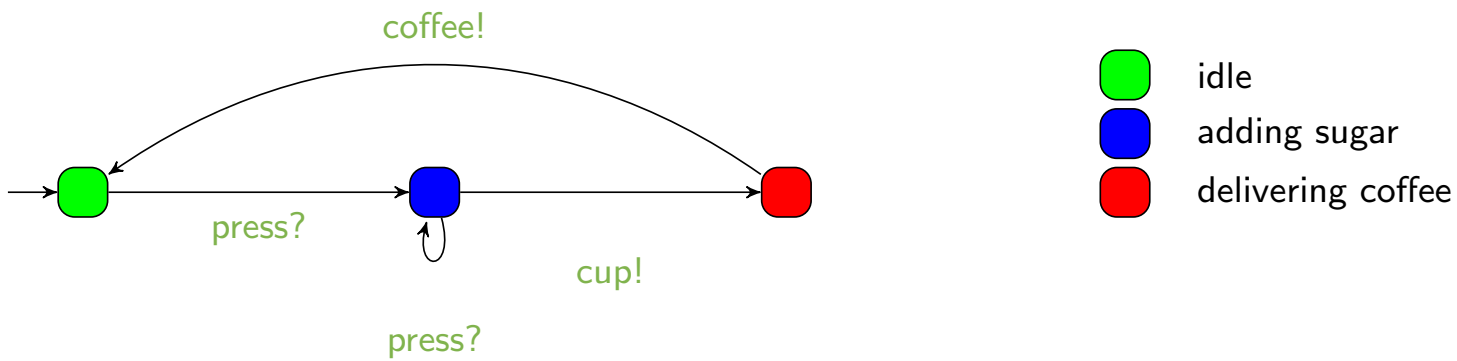




[AD94] Rajeev Alur and David L. Dill. "A theory of timed automata". In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235. ISSN: 0304-3975. DOI: 10.1016/0304-3975(94)90010-8

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

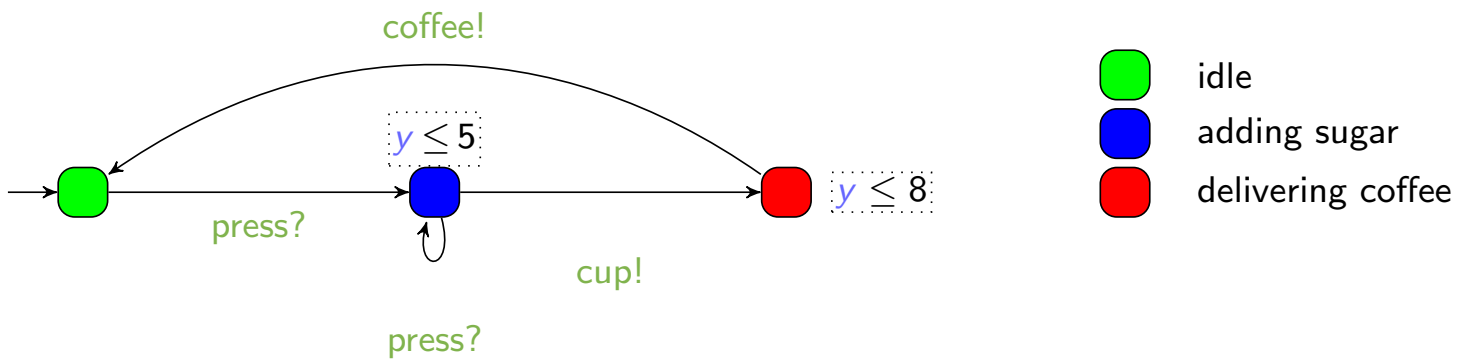


[AD94] Rajeev Alur and David L. Dill. "A theory of timed automata". In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235. ISSN: 0304-3975. DOI: 10.1016/0304-3975(94)90010-8

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

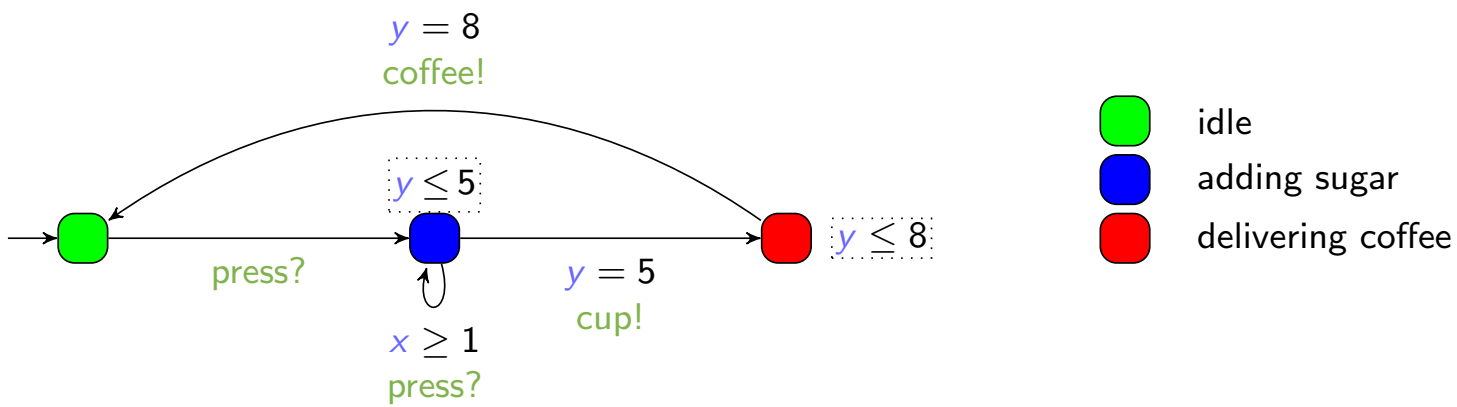


[AD94] Rajeev Alur and David L. Dill. "A theory of timed automata". In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235. ISSN: 0304-3975. DOI: 10.1016/0304-3975(94)90010-8

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

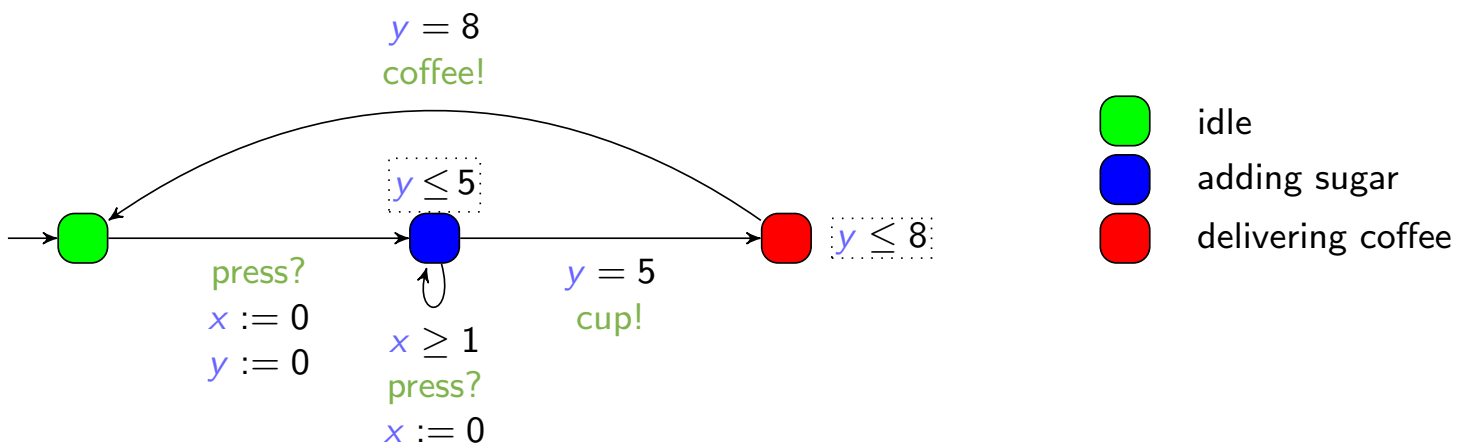


[AD94] Rajeev Alur and David L. Dill. "A theory of timed automata". In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235. ISSN: 0304-3975. DOI: 10.1016/0304-3975(94)90010-8

Timed automaton (TA)

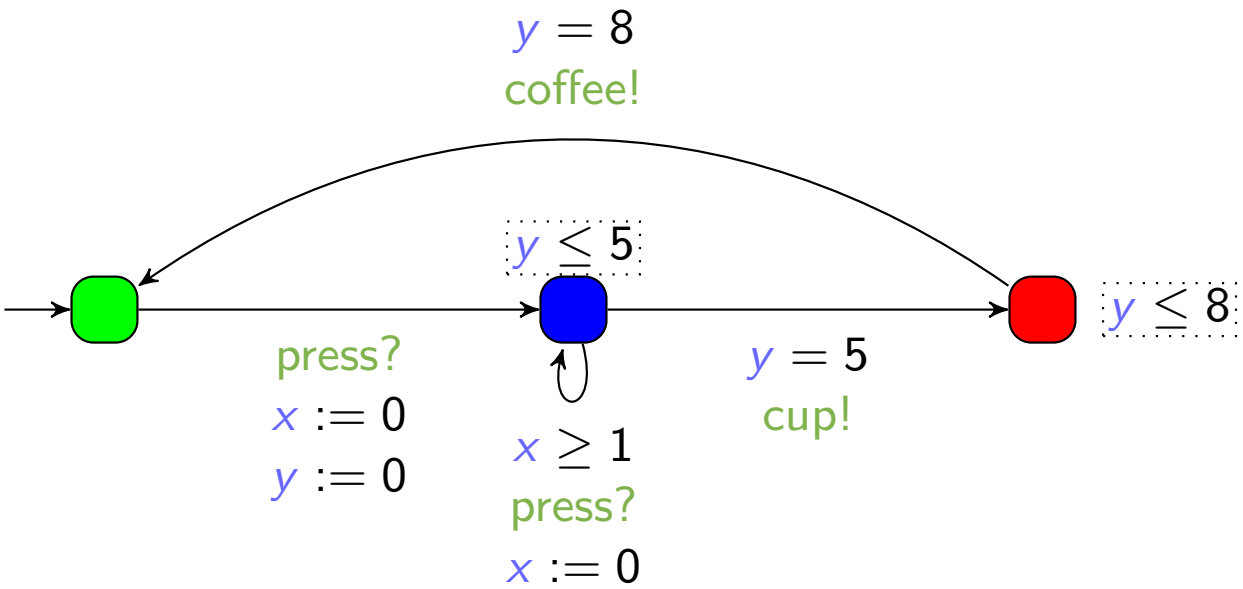
- Finite state automaton (sets of locations and actions) augmented with a set X of clocks [AD94]
  - Real-valued variables evolving linearly at the same rate
  - Can be compared to integer constants in invariants and guards

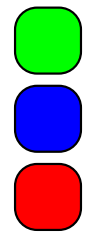
- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property to be verified to enable a transition
  - Clock reset: some of the clocks can be set to 0 along transitions



[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



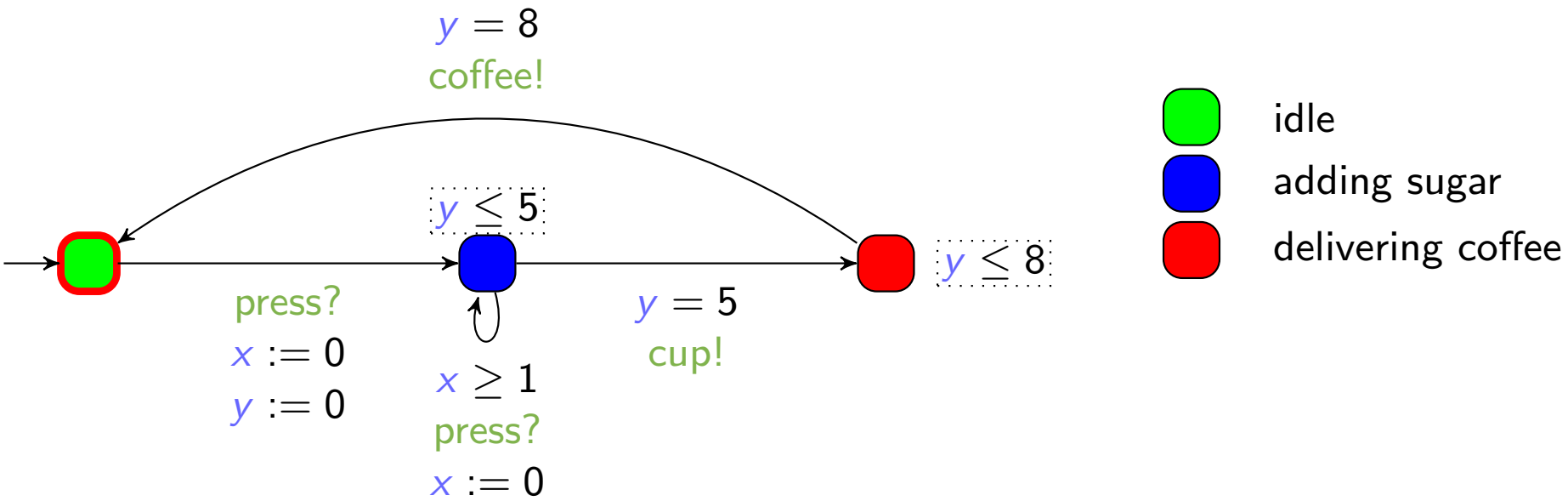


idle

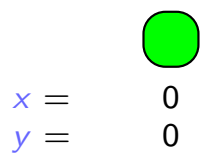
adding sugar

delivering coffee

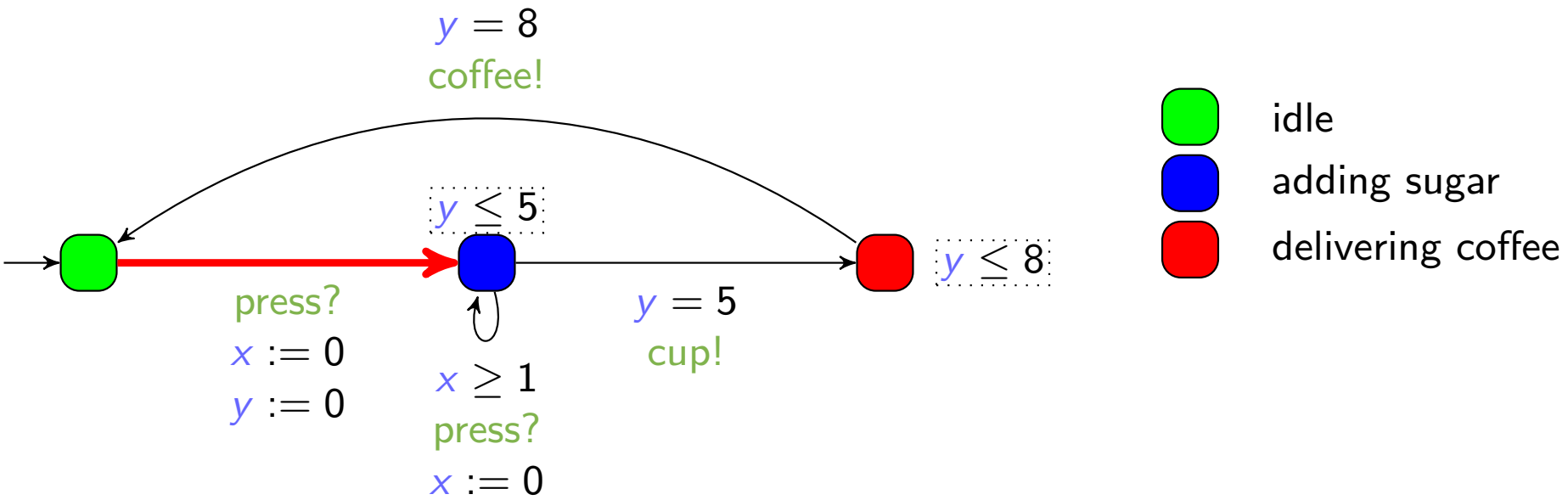
The most critical system: The coffee machine



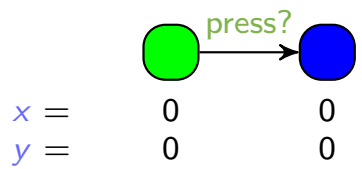
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



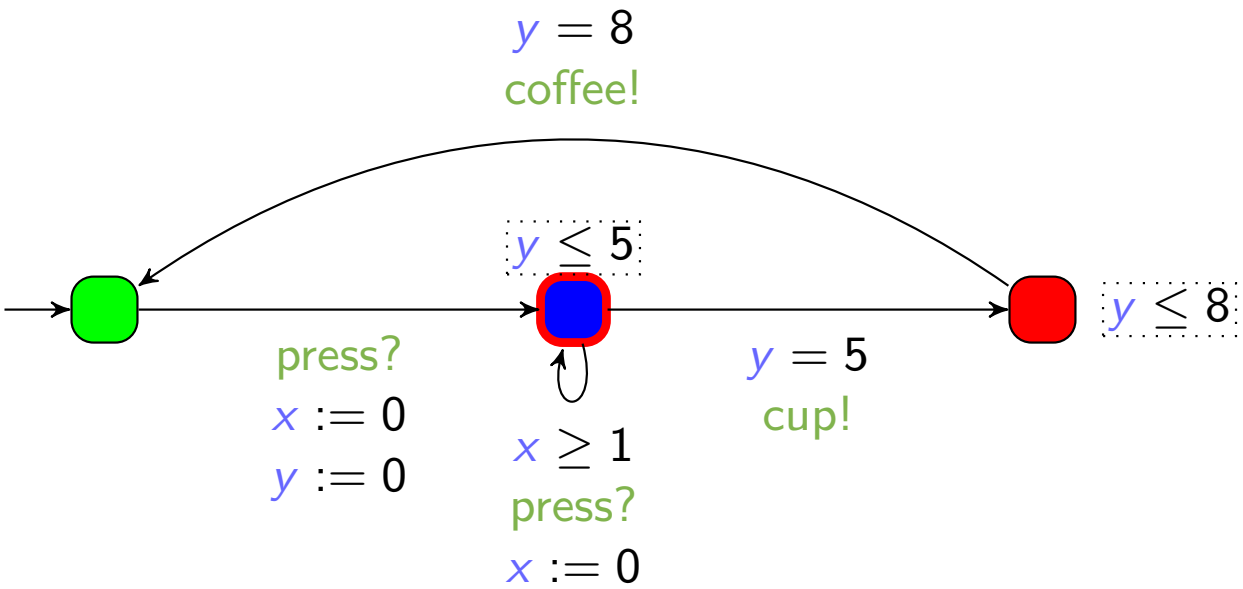
The most critical system: The coffee machine

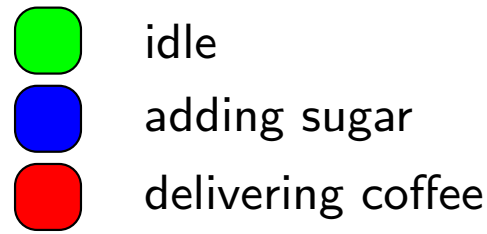


- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar

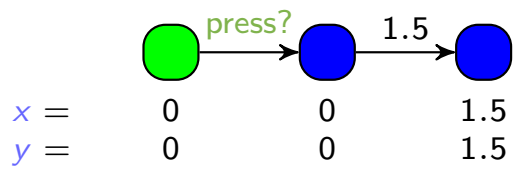


The most critical system: The coffee machine

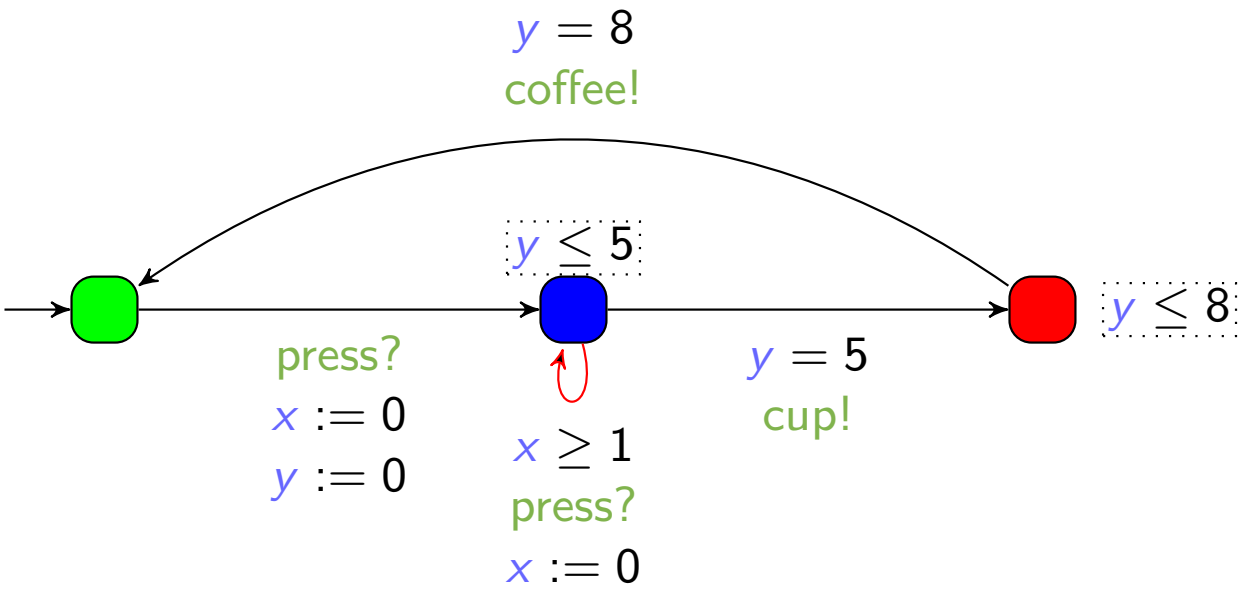


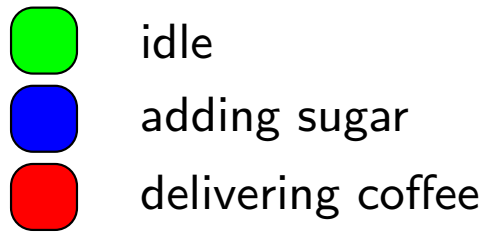


- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar

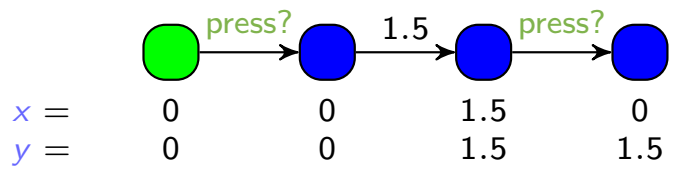


The most critical system: The coffee machine

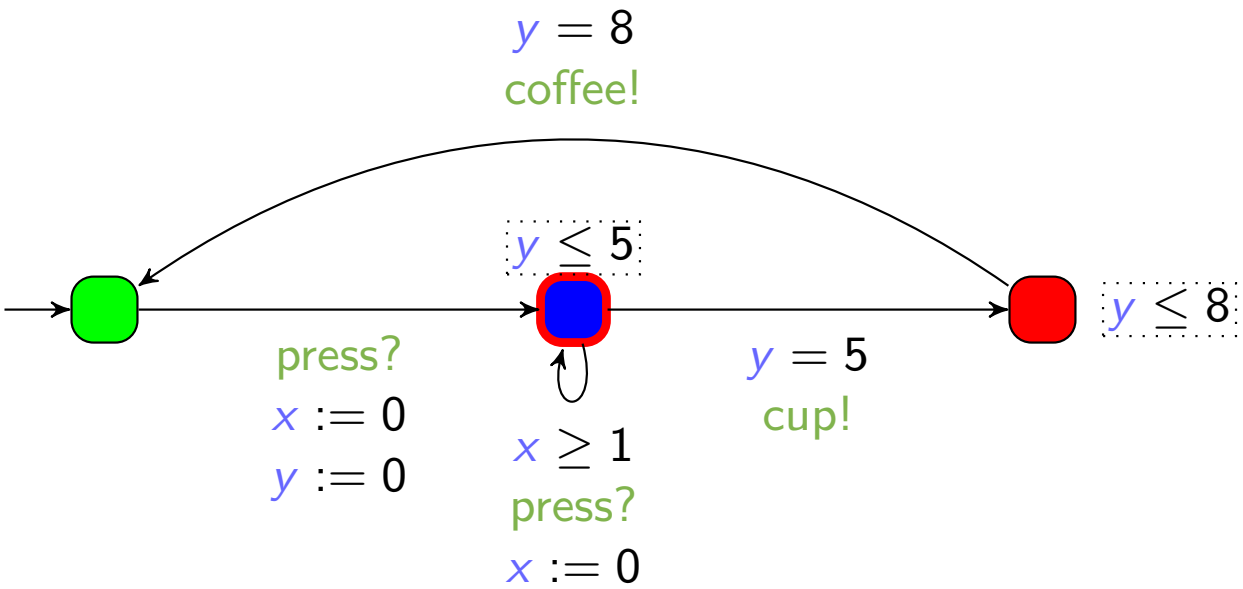


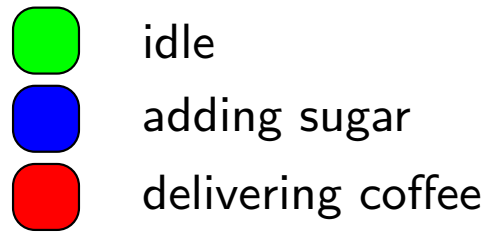


- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar

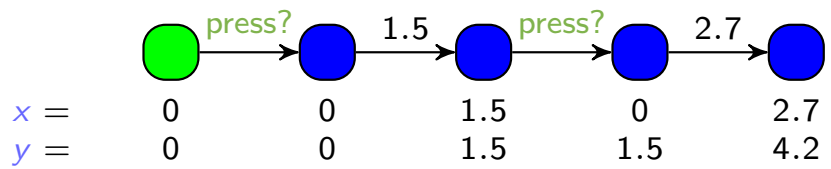


The most critical system: The coffee machine

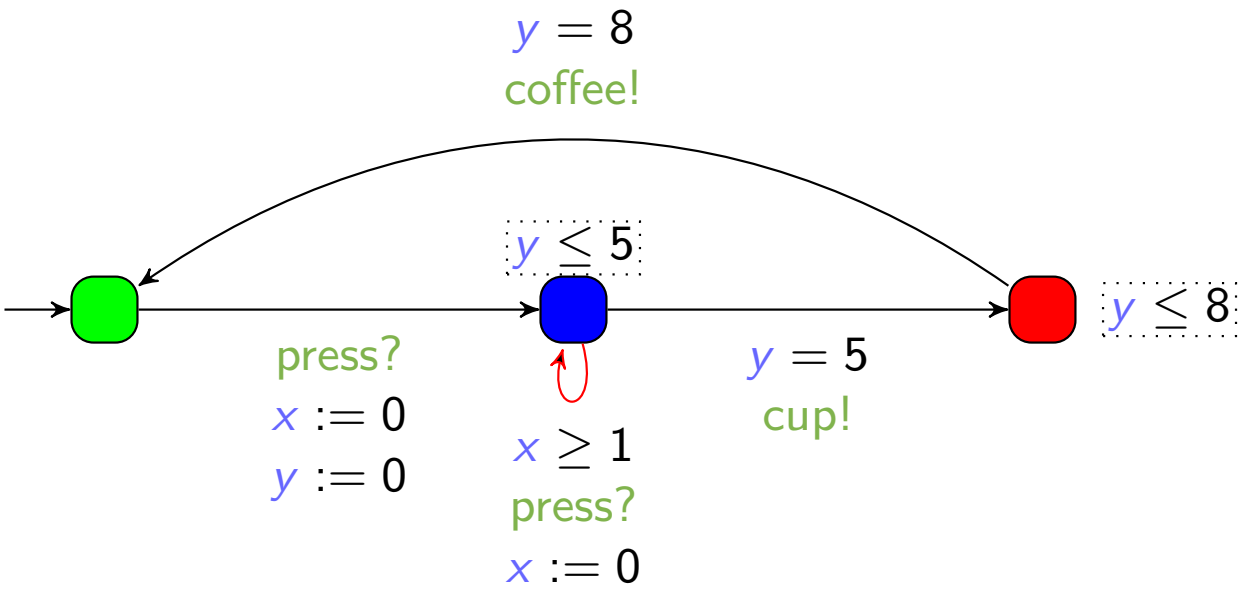


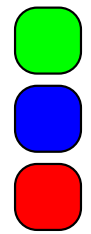


- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



The most critical system: The coffee machine





idle

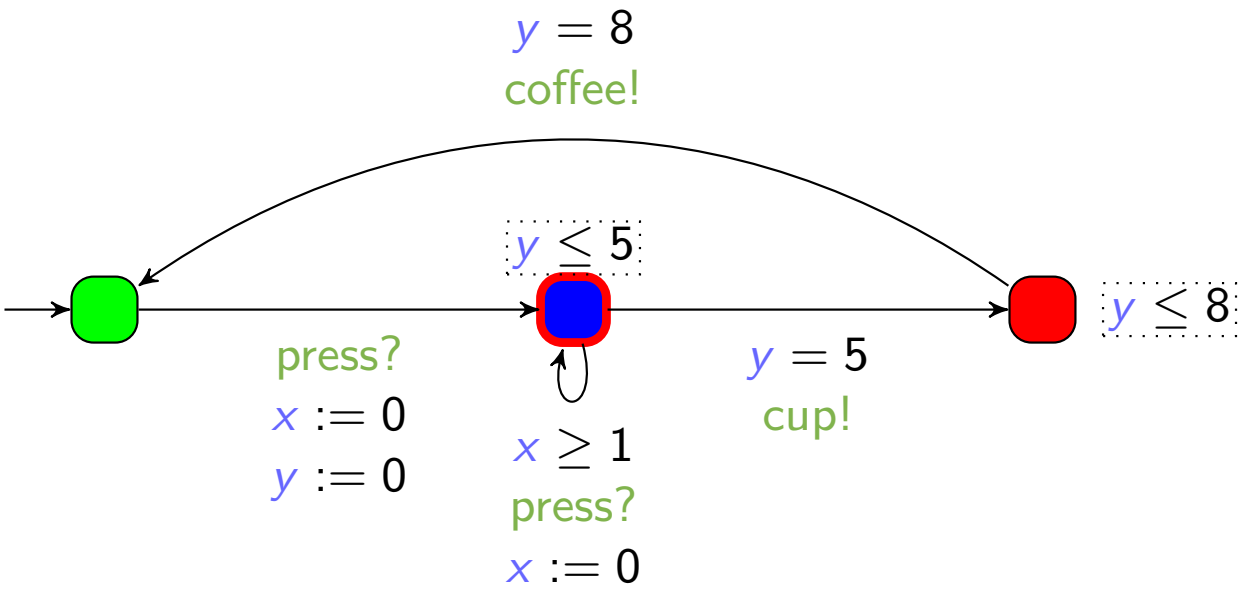
adding sugar

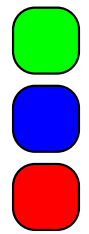
delivering coffee

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

$x =$	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



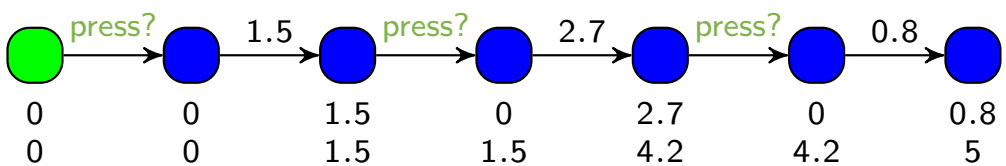


idle

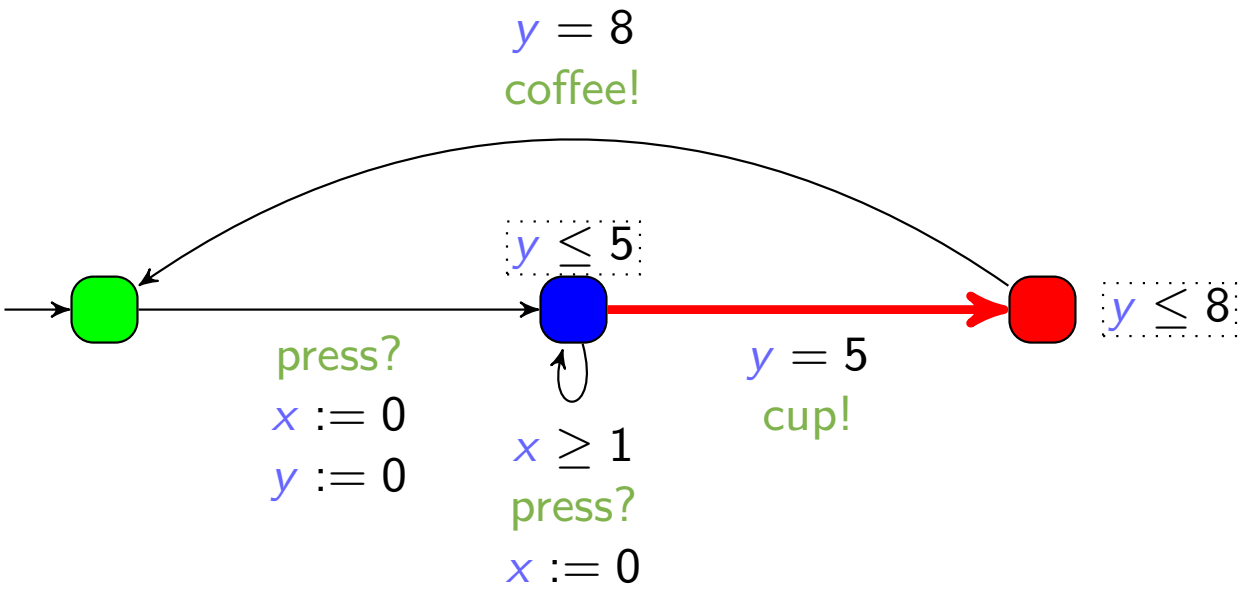
adding sugar

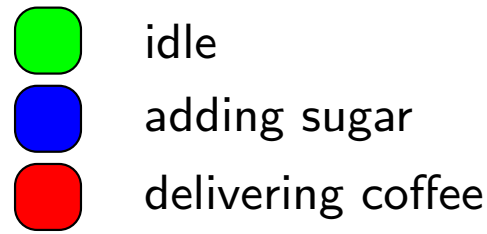
delivering coffee

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

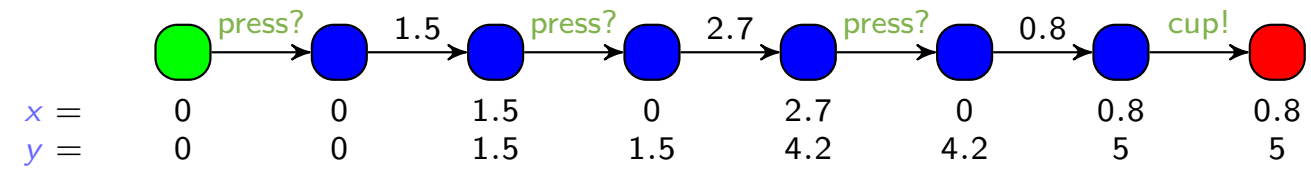


The most critical system: The coffee machine

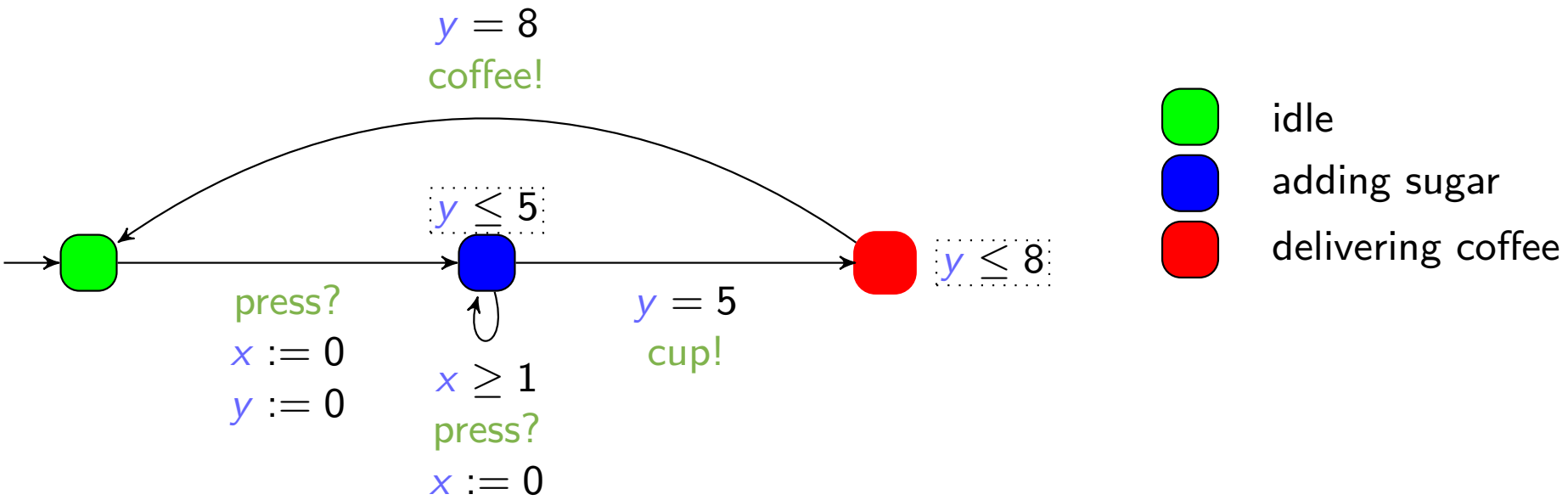




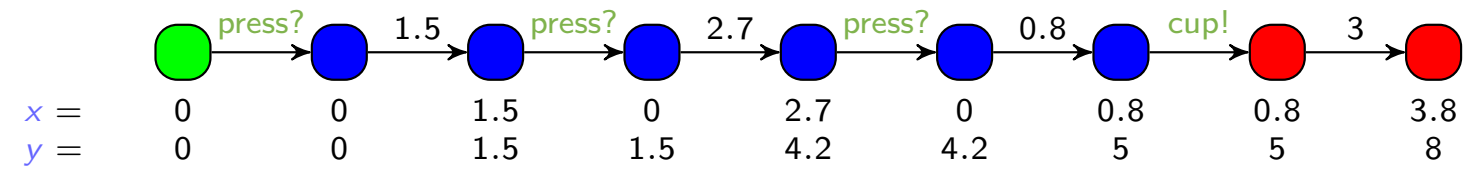
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



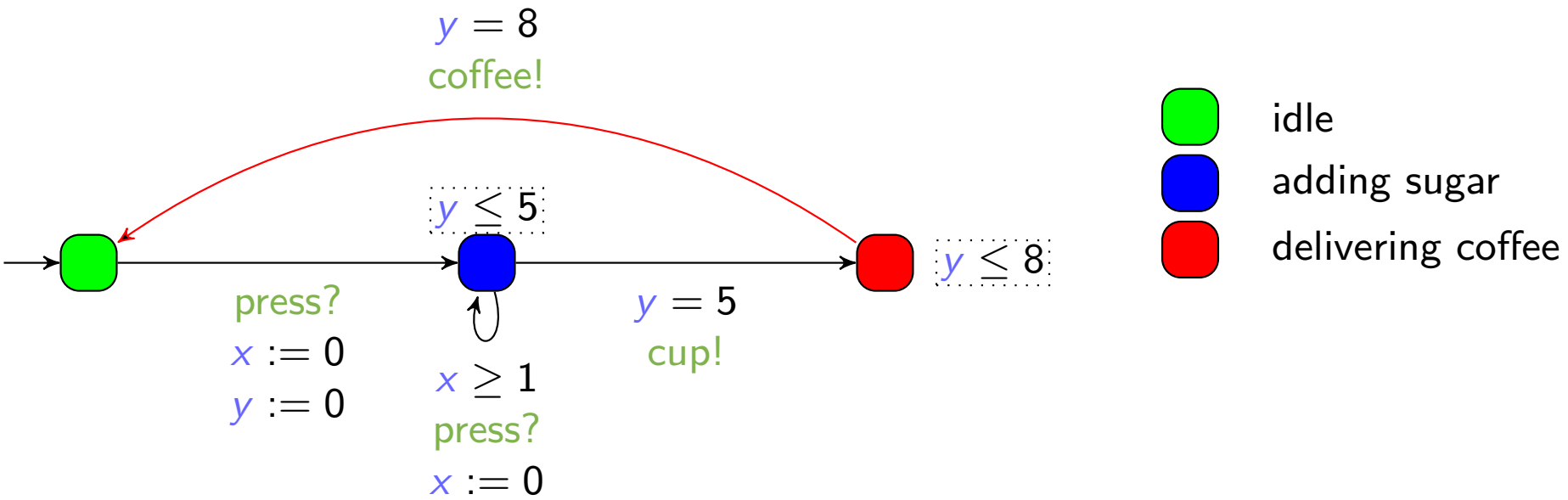
The most critical system: The coffee machine



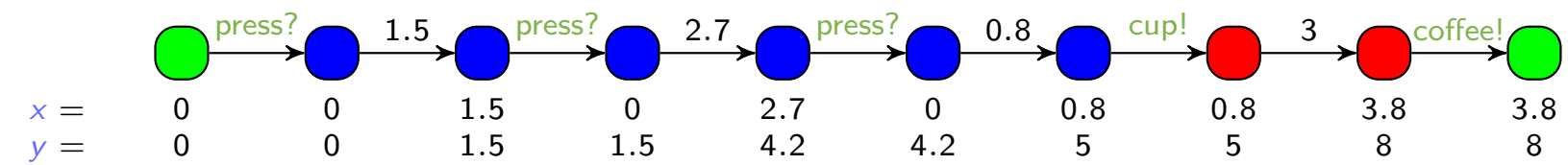
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



The most critical system: The coffee machine



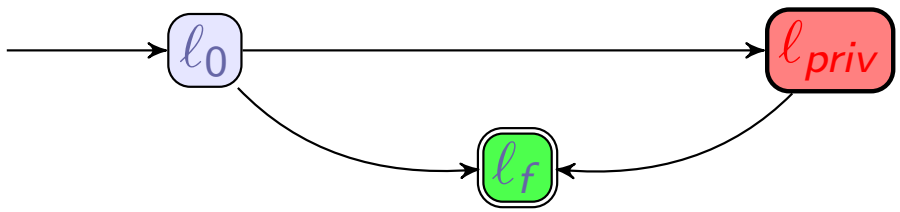
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



Formalization

Hypotheses: [AS19]

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



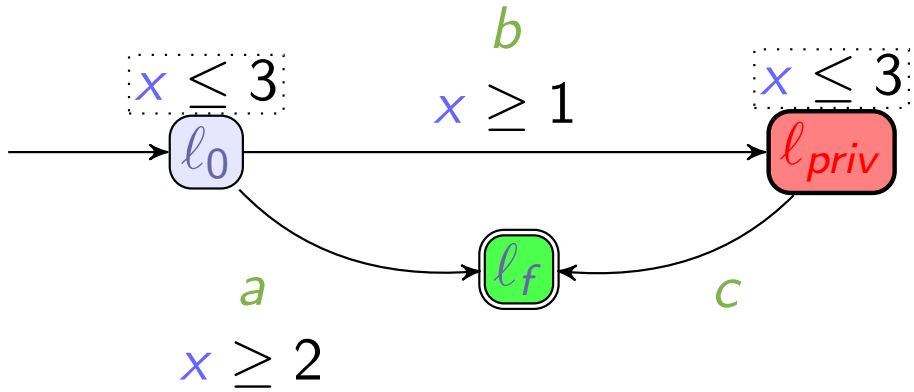
Definition (timed opacity)

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

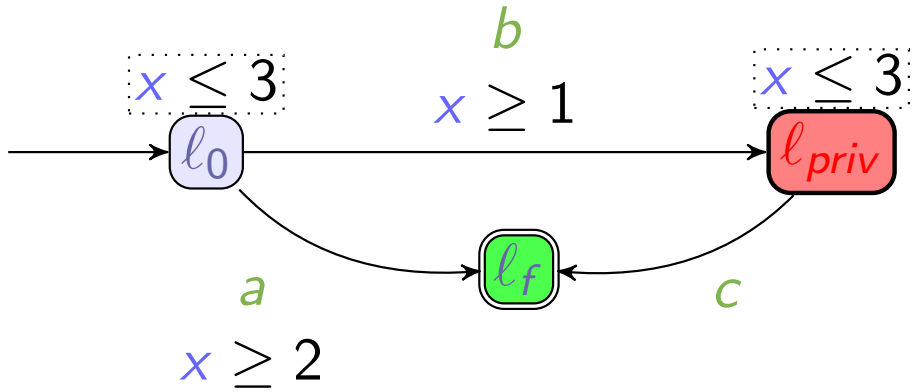
1. one passing by ℓ_{priv}
2. one *not* passing by ℓ_{priv}

[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

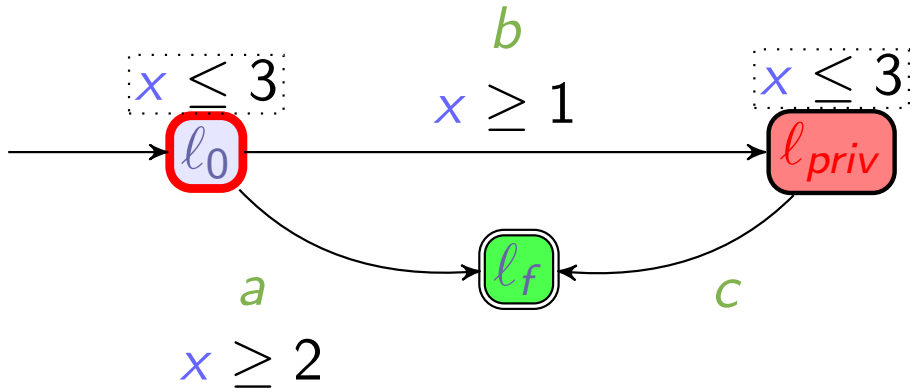


Example

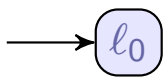


- There exist two runs of duration $d = 2$:

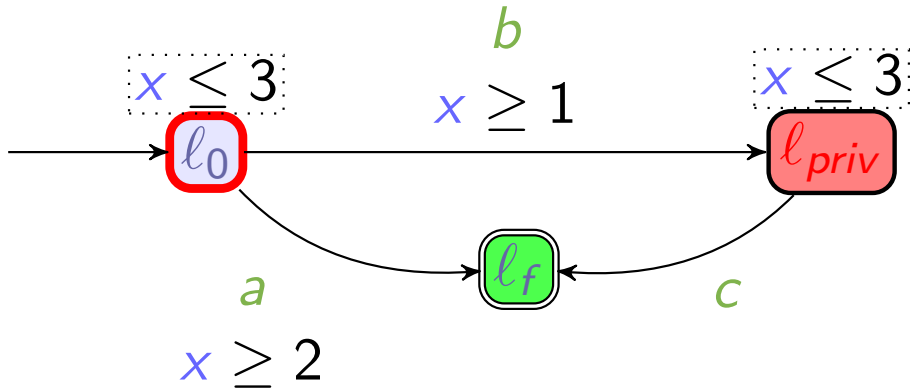
Example



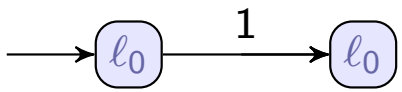
- There exist two runs of duration $d = 2$:



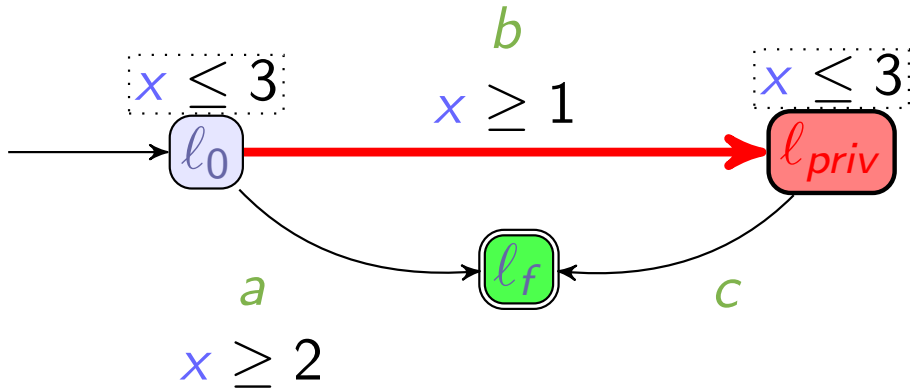
Example



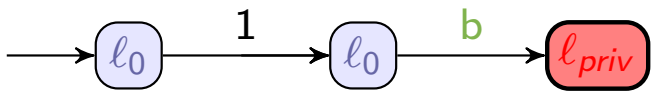
- There exist two runs of duration $d = 2$:



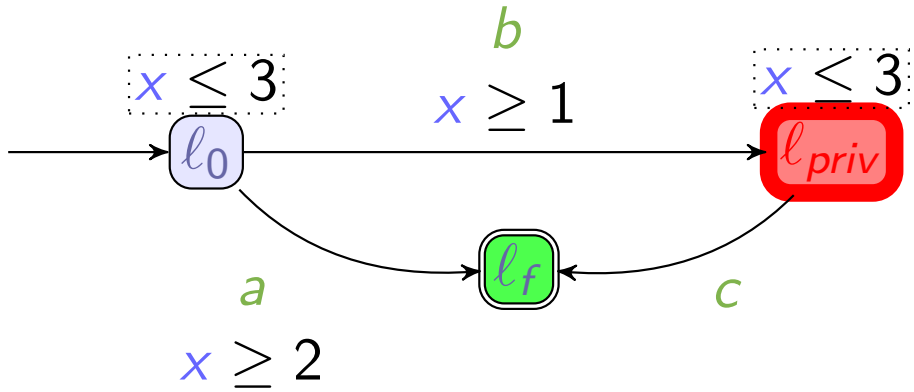
Example



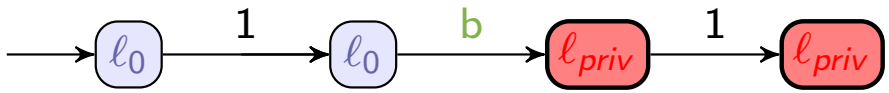
- There exist two runs of duration $d = 2$:



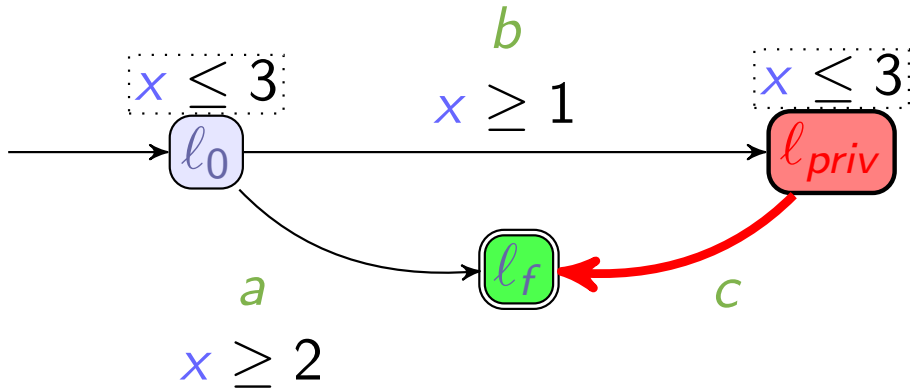
Example



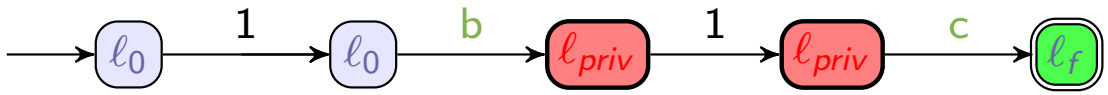
- There exist two runs of duration $d = 2$:



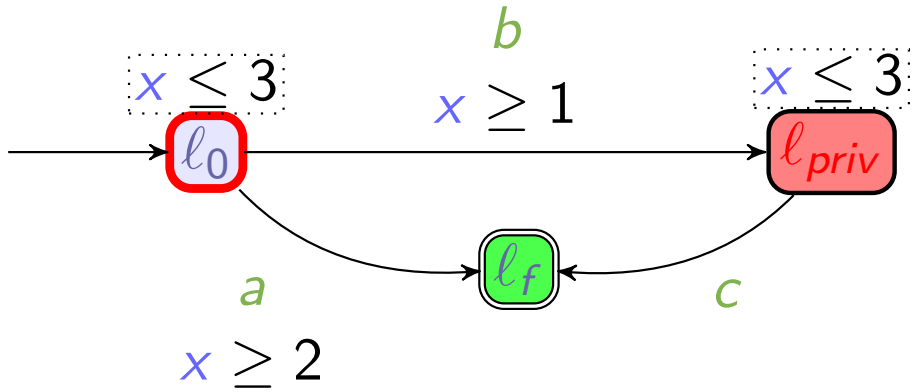
Example



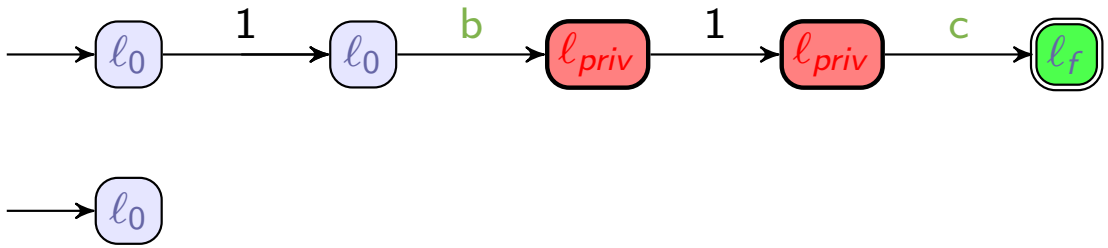
- There exist two runs of duration $d = 2$:



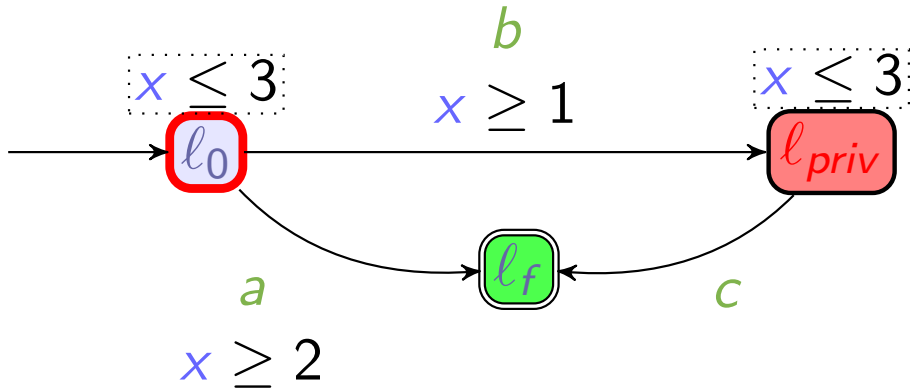
Example



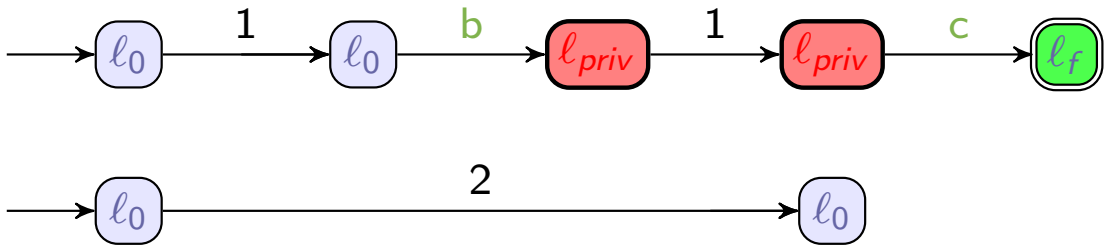
- There exist two runs of duration $d = 2$:



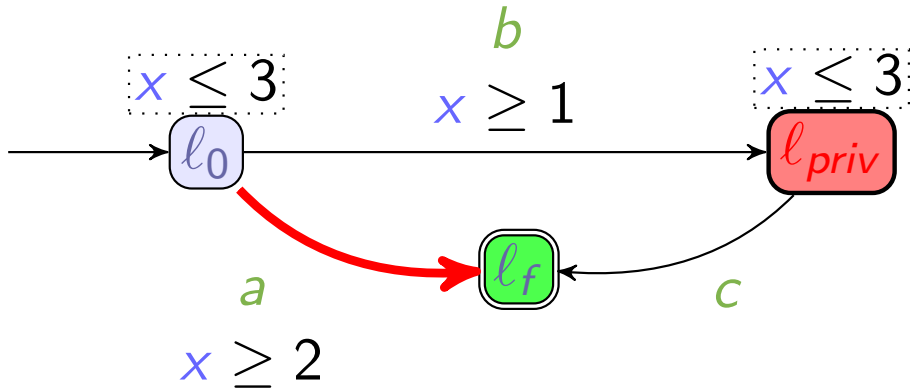
Example



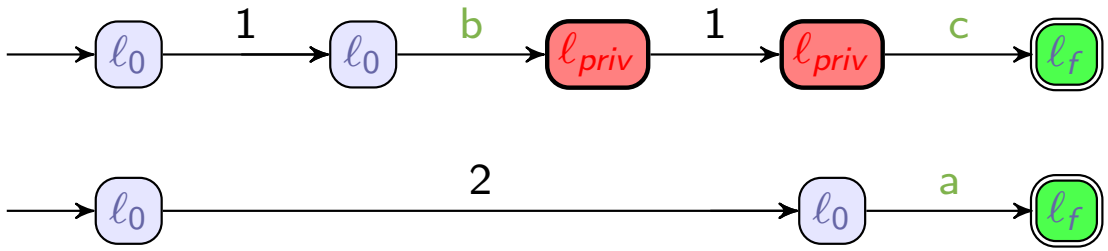
- There exist two runs of duration $d = 2$:



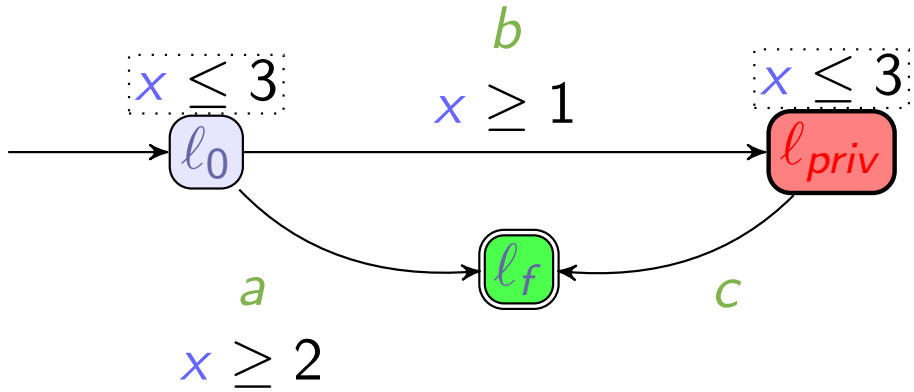
Example



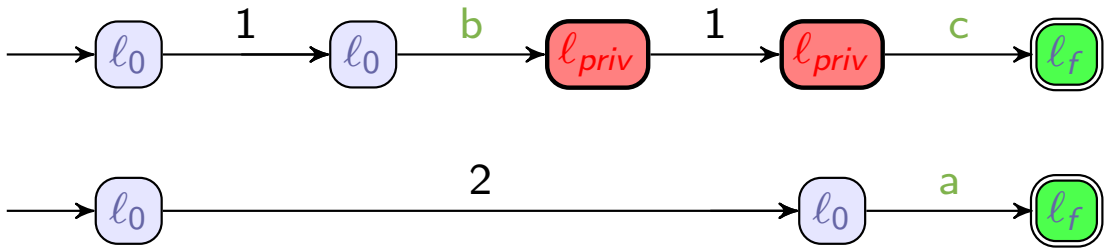
- There exist two runs of duration $d = 2$:



Example

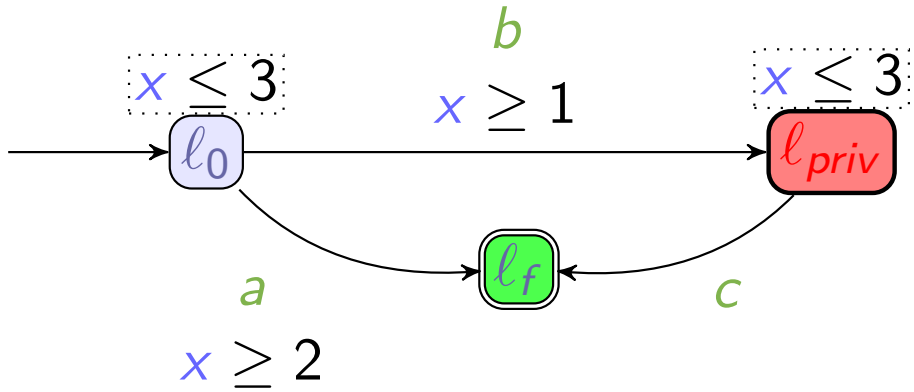


- There exist two runs of duration $d = 2$:

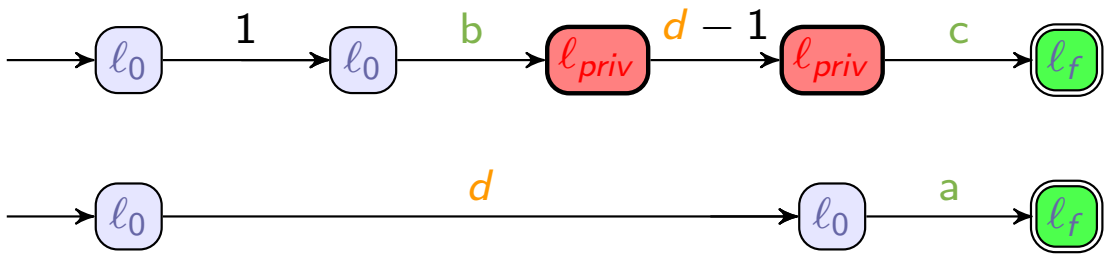


We say that the system is opaque w.r.t. ℓ_{priv} on the way to ℓ_f for a duration $d = 2$

Example

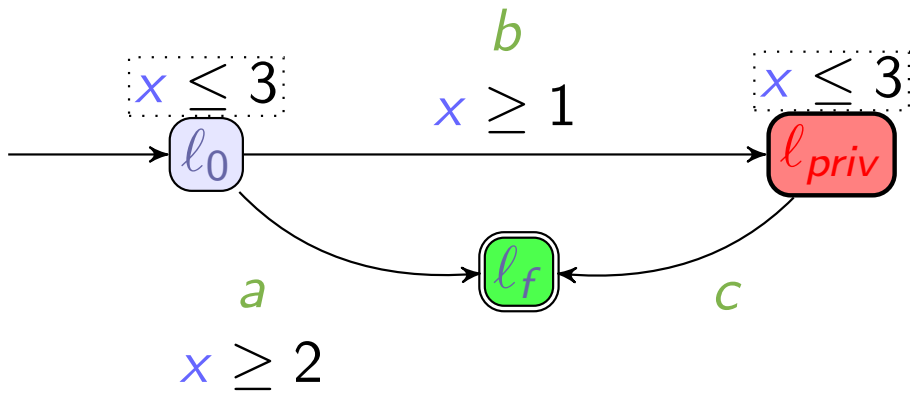


- There exist two runs of duration d for all durations $d \in [2, 3]$:



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

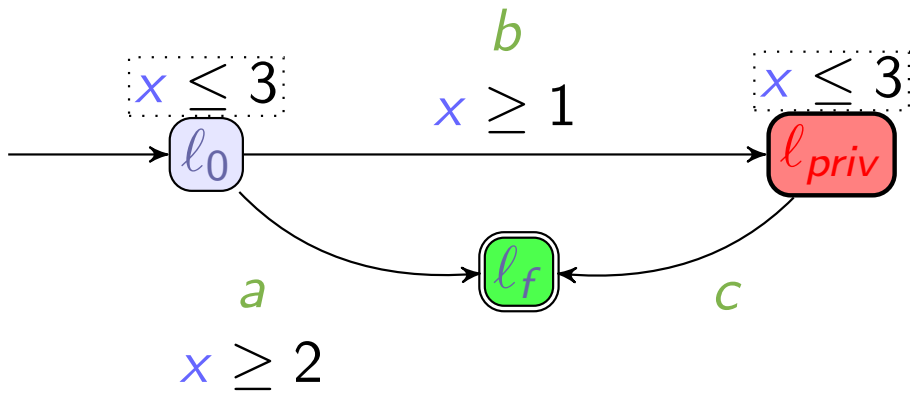
Example



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

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

Example

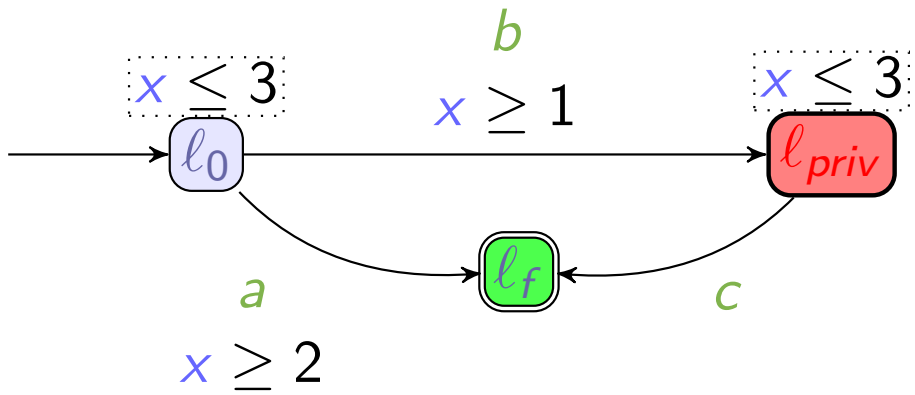


- There exist two runs of duration d for all durations $d \in [2, 3]$:

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

- But

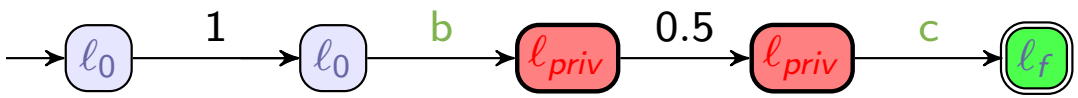
Example



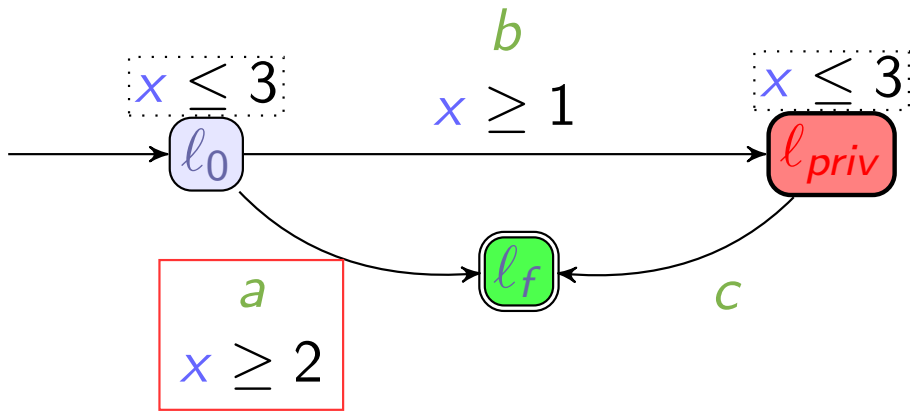
- There exist two runs of duration d for all durations $d \in [2, 3]$:

We say that the system is 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 passing by ℓ_{priv}



Example

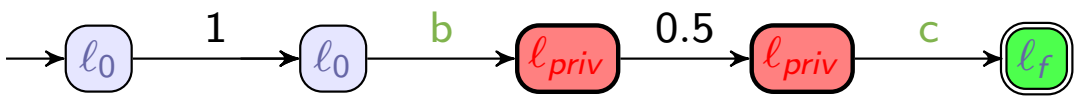


- There exist two runs of duration d for all durations $d \in [2, 3]$:

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

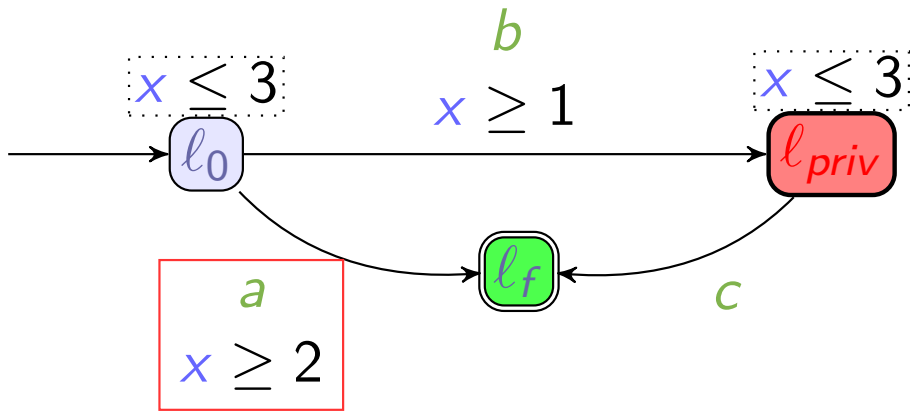
- But

  There exists a run of duration 1.5 passing by ℓ_{priv}



  It is not possible to reach ℓ_f with a path of duration 1.5 not passing by ℓ_{priv}

Example

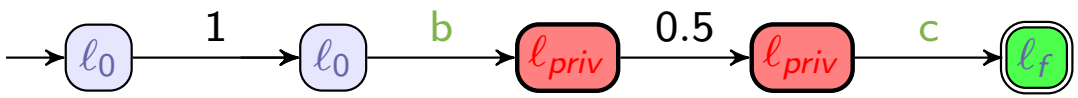


- There exist two runs of duration d for all durations $d \in [2, 3]$:

We say that the system is 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 passing by ℓ_{priv}



  It is not possible to reach ℓ_f with a path of duration 1.5 not passing by ℓ_{priv}

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

Problem 1: timed-opacity computation

Timed-opacity computation problem

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

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

[TOSEM22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. "Guaranteeing Timed Opacity using Parametric Timed Model Checking". In: *TOSEM* (2022). To appear

Problem 1: timed-opacity computation

Timed-opacity computation problem

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

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

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

[TOSEM22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. "Guaranteeing Timed Opacity using Parametric Timed Model Checking". In: *TOSEM* (2022). To appear

Problem 1: timed-opacity computation

Timed-opacity computation problem

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

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

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

Proof: based on the region graph and RA-arithmetic (see [TOSEM22])

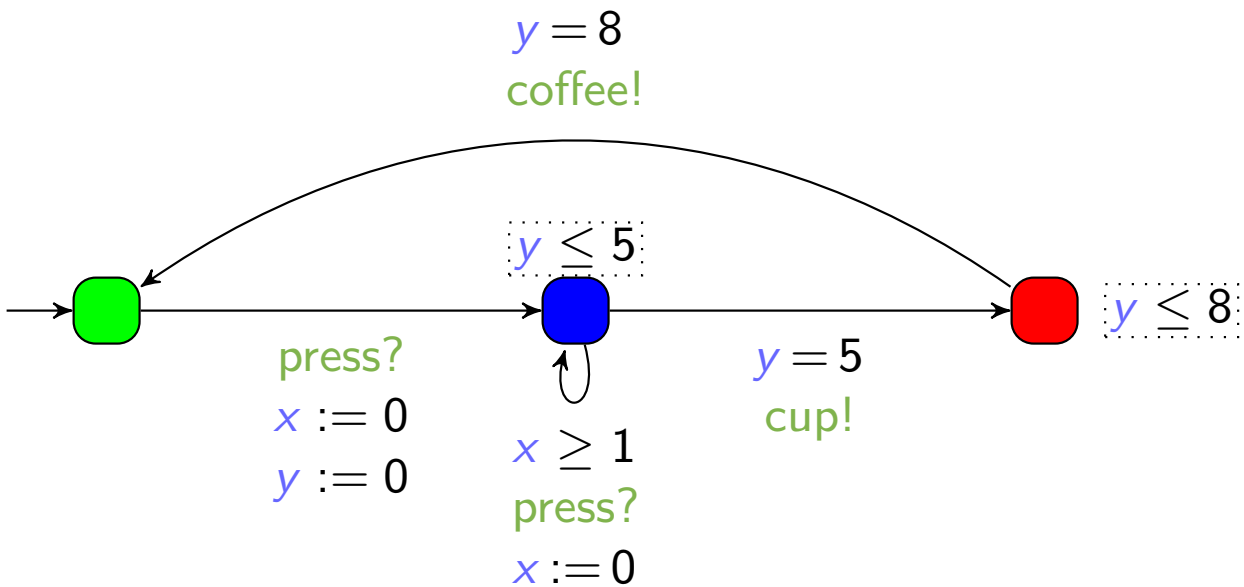
Exact complexity: unproved (EXPSPACE upper bound proved, but exponential hardness seems likely)

[TOSEM22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. "Guaranteeing Timed Opacity using Parametric Timed Model Checking". In: *TOSEM* (2022). To appear

Outline

Parametric Timed Automaton (PTA)

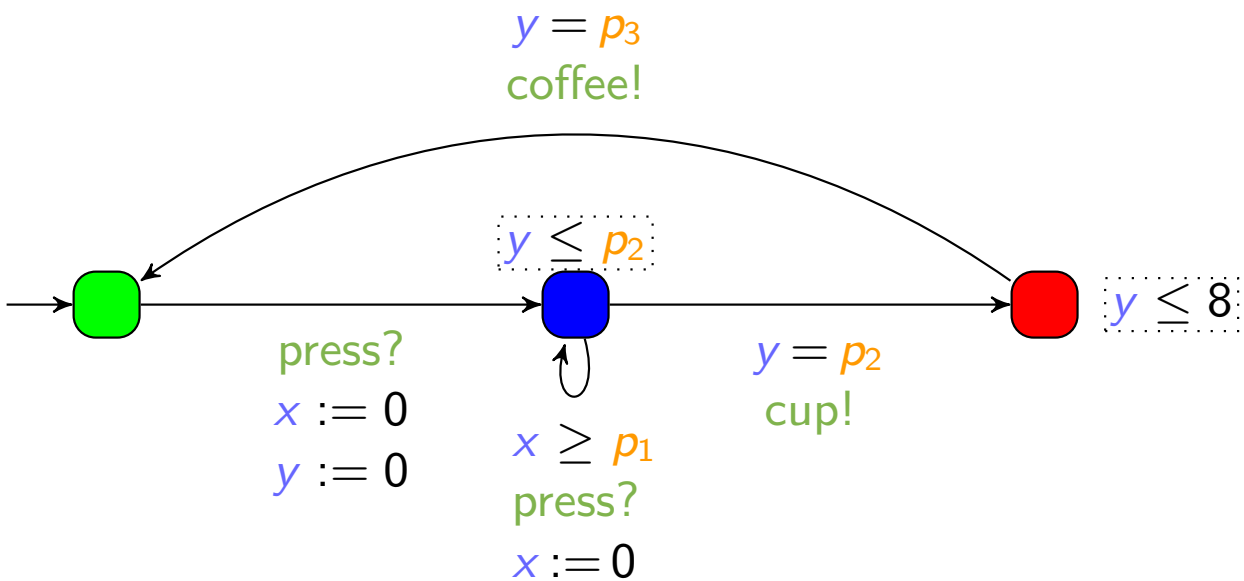
- Timed automaton (sets of locations, actions and clocks)



[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. "Parametric real-time reasoning". In: *STOC*. ed. by S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal. San Diego, California, United States: ACM, 1993, pp. 592–601. ISBN: 0-89791-591-7. 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



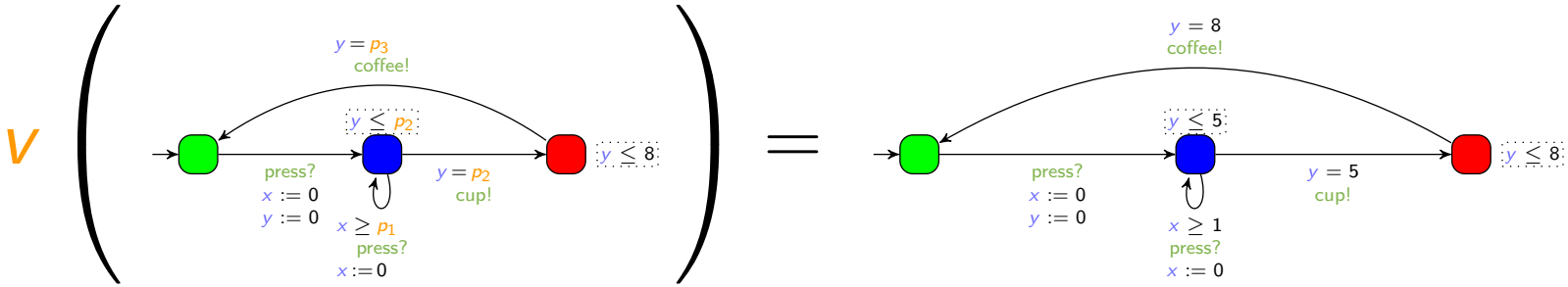
[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. "Parametric real-time reasoning". In: *STOC*. ed. by S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal. San Diego, California, United States: ACM, 1993, pp. 592–601. ISBN: 0-89791-591-7. DOI: 10.1145/167088.167242

Valuation of a PTA = TA

- Given a PTA $\mathcal{A}$ and a parameter valuation v,
  $v(\mathcal{A})$ is the TA where each parameter p is valuated by $v(p)$

Valuation of a PTA = TA

- Given a PTA $\mathcal{A}$ and a parameter valuation v,
  $v(\mathcal{A})$ is the TA where each parameter p is valuated by $v(p)$



$$\text{with } v : \begin{cases} p_1 & \rightarrow & 1 \\ p_2 & \rightarrow & 5 \\ p_3 & \rightarrow & 8 \end{cases}$$

Problem 2: timed-opacity synthesis

Timed-opacity synthesis problem

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

Problem 2: timed-opacity synthesis

Timed-opacity synthesis problem

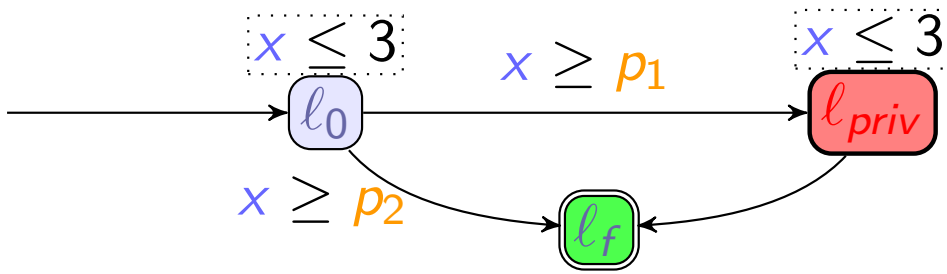
Find parameter valuations v and durations d ("execution times") of runs of $v(\mathcal{A})$ from ℓ_0 to ℓ_f such that the system is opaque w.r.t. ℓ_{priv} on the way to ℓ_f

Problem 2: timed-opacity synthesis

Timed-opacity synthesis problem

Find parameter valuations v and durations d ("execution times") of runs of $v(\mathcal{A})$ from ℓ_0 to ℓ_f such that the system is opaque w.r.t. ℓ_{priv} on the way to ℓ_f

Example:

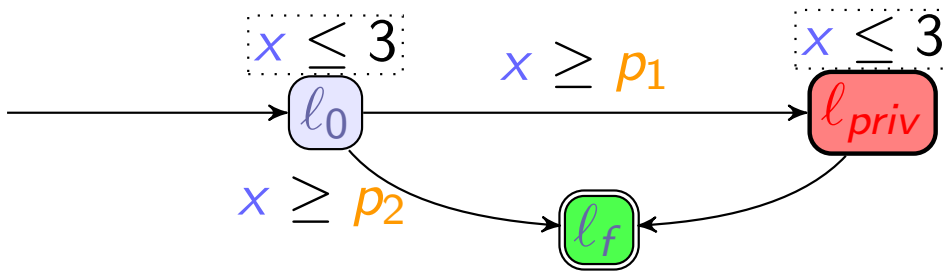


Problem 2: timed-opacity synthesis

Timed-opacity synthesis problem

Find parameter valuations v and durations d ("execution times") of runs of $v(\mathcal{A})$ from ℓ_0 to ℓ_f such that the system is opaque w.r.t. ℓ_{priv} on the way to ℓ_f

Example:



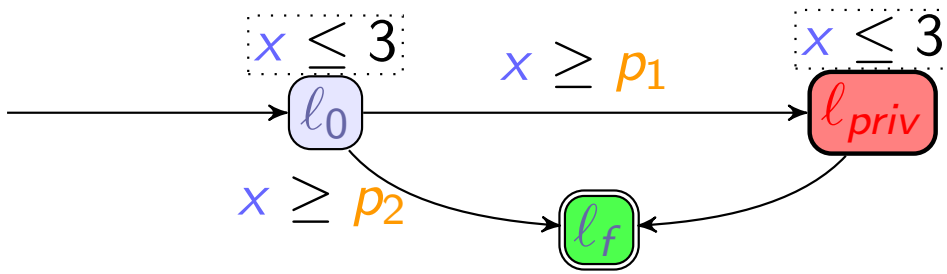
Expected result: $p_1 \leq 3 \wedge p_2 \leq 3 \wedge d \in [p_2, 3]$

Problem 2: timed-opacity synthesis

Timed-opacity synthesis problem

Find parameter valuations v and durations d ("execution times") of runs of $v(\mathcal{A})$ from ℓ_0 to ℓ_f such that the system is opaque w.r.t. ℓ_{priv} on the way to ℓ_f

Example:



Expected result: $p_1 \leq 3 \wedge p_2 \leq 3 \wedge d \in [p_2, 3]$

If $v(p_1) = 1$ and $v(p_2) = 2$: $\top \wedge \top \wedge d \in [2, 3]$

Overview of our theoretical results

- General case: The mere existence of a parameter valuation for which there exists a duration for which timed-opacity is achieved is undecidable

[TOSEM22]

[TOSEM22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. "Guaranteeing Timed Opacity using Parametric Timed Model Checking". In: *TOSEM* (2022). To appear

Overview of our theoretical results

- General case: The mere existence of a parameter valuation for which there exists a duration for which timed-opacity is achieved is undecidable

- Study of a subclass known for being "at the frontier" of decidability (L/U-PTA)
  - The existence of valuations for timed opacity w.r.t. some execution times is decidable
  - The existence of valuations for full timed opacity is undecidable
  - The synthesis is uncomptable in pratice

[TOSEM22]

[TOSEM22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. "Guaranteeing Timed Opacity using Parametric Timed Model Checking". In: *TOSEM* (2022). To appear

Overview of our theoretical results

- General case: The mere existence of a parameter valuation for which there exists a duration for which timed-opacity is achieved is undecidable

- Study of a subclass known for being "at the frontier" of decidability (L/U-PTA)
  - The existence of valuations for timed opacity w.r.t. some execution times is decidable
  - The existence of valuations for full timed opacity is undecidable
  - The synthesis is uncomptable in pratice

[TOSEM22]

We adopt a "best-effort" approach for the general case of PTAs

- Approach not guaranteed to terminate in theory

[TOSEM22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. "Guaranteeing Timed Opacity using Parametric Timed Model Checking". In: *TOSEM* (2022). To appear

Outline

Experimental environment

Algorithms

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

Benchmarks

- Common PTA benchmarks [TAP21]
- 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/`

Experiments: (non-parametric) timed 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)	Opaque?
Fig. 5, [VNN18]	1	1	2	3	3	0.02	$(\times)$
Fig. 1b, [GMR07]	1	1	2	3	1	0.04	$(\times)$
Fig. 2a, [GMR07]	1	1	2	3	1	0.05	$(\times)$
Fig. 2b, [GMR07]	1	1	2	3	1	0.02	$(\times)$
Web privacy problem [Ben+15]	1	2	2	4	1	0.07	$(\times)$
`Coffee`	1	2	2	5	1	0.05	$\surd$
`Fischer-HSRV02`	3	2	6	5	1	5.83	$(\times)$
`STAC:1:n`			2	3	6	0.12	$(\times)$
`STAC:1:v`			2	3	6	0.11	$\times$
`STAC:3:n`			2	3	8	0.72	$\surd$
`STAC:3:v`			2	3	8	0.74	$(\times)$
`STAC:4:n`			2	3	8	6.40	$\times$
`STAC:4:v`			2	3	8	265.52	$\times$
`STAC:5:n`			2	3	6	0.24	$\surd$
`STAC:11A:v`			2	3	8	47.77	$(\times)$
`STAC:11B:v`			2	3	8	59.35	$(\times)$
`STAC:12c:v`			2	3	8	18.44	$\times$
`STAC:12e:n`			2	3	8	0.58	$\times$
`STAC:12e:v`			2	3	8	1.10	$(\times)$
`STAC:14:n`			2	3	8	22.34	$(\times)$

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

Experiments: (parametric) timed-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

Outline

Perspectives

On the theoretical side

- Some restricted problems remain open
  e. g., PTA with one clock
- Study more restritive sub-classes, with the hope to exhibit a decidable one

Perspectives

On the theoretical side

- Some restricted problems remain open
  e. g., PTA with one clock
- Study more restritive sub-classes, with the hope to exhibit a decidable one

On the pratical side

- Have an automatic translation of programs to PTAs
  → Some experiments were done, but on Java programs manually translated to PTAs
- Repairing a non-opaque system
  → Preliminary ideas in [TOSEM22][a], but not fixed

[a] [TOSEM22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. "Guaranteeing Timed Opacity using Parametric Timed Model Checking". In: *TOSEM* (2022). To appear

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.

[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. "Parametric real-time reasoning". In: *STOC*. Ed. by S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal. San Diego, California, United States: ACM, 1993, pp. 592–601. ISBN: 0-89791-591-7. DOI: 10.1145/167088.167242.

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.

[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.

References III

[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.

[TAP21] Étienne André, Dylan Marinho, and Jaco van de Pol. "A Benchmarks Library for Extended Parametric Timed Automata". In: *TAP 2021*. Ed. by Frédéric Loulergue and Franz Wotawa. Vol. 12740. Lecture Notes in Computer Science. Springer, 2021, pp. 39–50. DOI: 10.1007/978-3-030-79379-1_3.

[TOSEM22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. "Guaranteeing Timed Opacity using Parametric Timed Model Checking". In: *TOSEM* (2022). To appear.

References IV

[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. Lecture Notes in Computer Science. Thessaloniki, Greece: Springer, 2018, pp. 28–52. DOI: 10.1007/978-3-319-89722-6_2.