Automata with Timers

To be published at FORMATS 2023

Véronique Bruyère, Guillermo A. Pérez, Gaëtan Staquet, Frits W. Vaandrager

Theoretical computer science
Computer Science Department
Science Faculty
University of Mons

Formal Techniques in Software Engineering
Computer Science Department
Science Faculty
University of Antwerp

July 25, 2023

UMONS Université de Mons

fnrs LA LIBERTÉ DE CHERCHER

University of Antwerp

Many computer systems have timing constraints:

- Network protocols;
- Schedulers;
- Embedded systems;
- In general, real-time systems.

1

Many computer systems have timing constraints:

- Network protocols;
- Schedulers;
- Embedded systems;
- In general, real-time systems.

Well-known model for these systems: timed automata.[1]

[1] Baier and Katoen, *Principles of model checking*, 2008; Clarke et al., *Handbook of Model Checking*, 2018

Many computer systems have timing constraints:

- Network protocols;
- Schedulers;
- Embedded systems;
- In general, real-time systems.

Well-known model for these systems: timed automata.[1]

In short: finite automata augmented with clocks that can be reset or used in guards along transitions and states.

[1] Baier and Katoen, *Principles of model checking*, 2008; Clarke et al., *Handbook of Model Checking*, 2018

Many computer systems have timing constraints:

- Network protocols;
- Schedulers;
- Embedded systems;
- In general, real-time systems.

Well-known model for these systems: timed automata.[1]

In short: finite automata augmented with clocks that can be reset or used in guards along transitions and states.

BUT timed automata are hard to construct and understand.

[1] Baier and Katoen, *Principles of model checking*, 2008; Clarke et al., *Handbook of Model Checking*, 2018

We focus on properties that can be represented with timers: automata with timers.

Timed automata

-
-
-
-
-

Automata with timers

-
-
-
-
-

2

We focus on properties that can be represented with timers: automata with timers.

Timed automata

- Clocks go from 0 to infinity;
-
-
-
-

Automata with timers

- Timers go from a value set by the transition to 0;
-
-
-
-

2

We focus on properties that can be represented with timers: automata with timers.

Timed automata

- Clocks go from 0 to infinity;
- We know the current value of the clocks;
-
-
-

Automata with timers

- Timers go from a value set by the transition to 0;
- We do not know the current value of the timers;
-
-
-

2

We focus on properties that can be represented with timers: automata with timers.

Timed automata	Automata with timers
▶ Clocks go from 0 to infinity;	▶ Timers go from a value set by the transition to 0;
▶ We know the current value of the clocks;	▶ We do not know the current value of the timers;
▶ Timed automata are more expressive;	▶ Automata with timers are more restrictive;
▶	▶
▶	▶

2

We focus on properties that can be represented with timers: automata with timers.

Timed automata

- Clocks go from 0 to infinity;
- We know the current value of the clocks;
- Timed automata are more expressive;
- Learning (à la Angluin[2]) timed automata is challenging;
-

Automata with timers

- Timers go from a value set by the transition to 0;
- We do not know the current value of the timers;
- Automata with timers are more restrictive;
- Future work: learning algorithm;
-

[2] Angluin, "Learning Regular Sets from Queries and Counterexamples", 1987

We focus on properties that can be represented with timers: automata with timers.

Timed automata	**Automata with timers**
▶ Clocks go from 0 to infinity;	▶ Timers go from a value set by the transition to 0;
▶ We know the current value of the clocks;	▶ We do not know the current value of the timers;
▶ Timed automata are more expressive;	▶ Automata with timers are more restrictive;
▶ Learning (à la Angluin[2]) timed automata is challenging;	▶ Future work: learning algorithm;
▶ Well-known model.	▶ This work studies some properties of automata with timers.

[2] Angluin, "Learning Regular Sets from Queries and Counterexamples", 1987

An automaton with timers (AT) is a tuple $\mathcal{A} = (X, I, Q, q_0, \chi, \delta)$ where

- X is the set of timers,
- I is the set of actions,

Figure 1: An AT.

An automaton with timers (AT) is a tuple $\mathcal{A} = (X, I, Q, q_0, \chi, \delta)$ where

- X is the set of timers,
- I is the set of actions,
- Q is the finite set of states,
- $q_0 \in Q$ is the initial state,

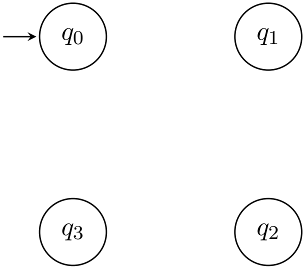


Figure 1: An AT.

An automaton with timers (AT) is a tuple $\mathcal{A} = (X, I, Q, q_0, \chi, \delta)$ where

- X is the set of timers,
- I is the set of actions,
- Q is the finite set of states,
- $q_0 \in Q$ is the initial state,
- $\chi : Q \to \mathcal{P}(X)$ gives the active timers of each state,

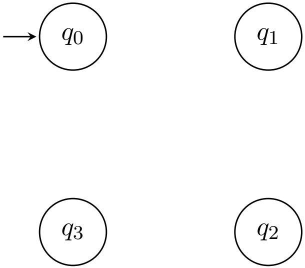


Figure 1: An AT.

An automaton with timers (AT) is a tuple $\mathcal{A} = (X, I, Q, q_0, \chi, \delta)$ where

- X is the set of timers,
- I is the set of actions,
- Q is the finite set of states,
- $q_0 \in Q$ is the initial state,
- $\chi : Q \to \mathcal{P}(X)$ gives the active timers of each state,
- δ is the transition function.

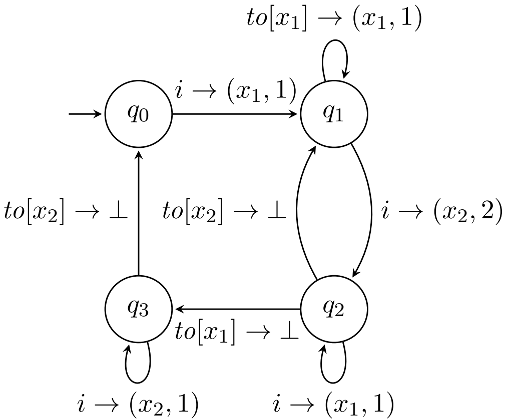


Figure 1: An AT.

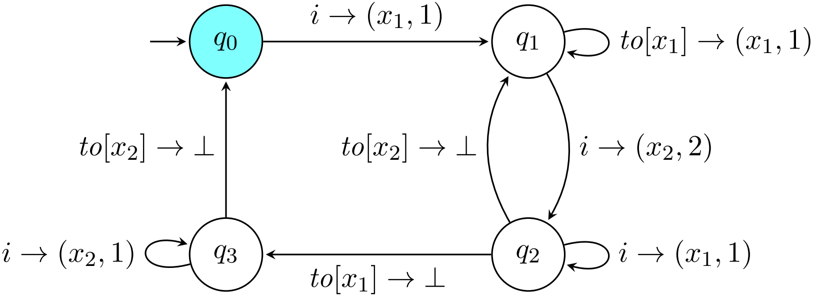


Figure 2: The same AT.

$(q_0, \emptyset)$

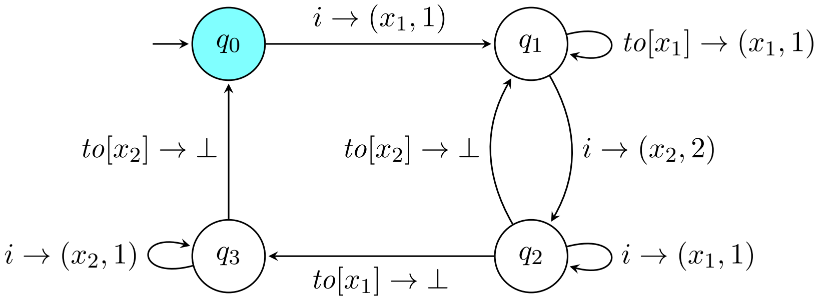


Figure 2: The same AT.

$$(q_0, \emptyset) \xrightarrow{1} (q_0, \emptyset)$$

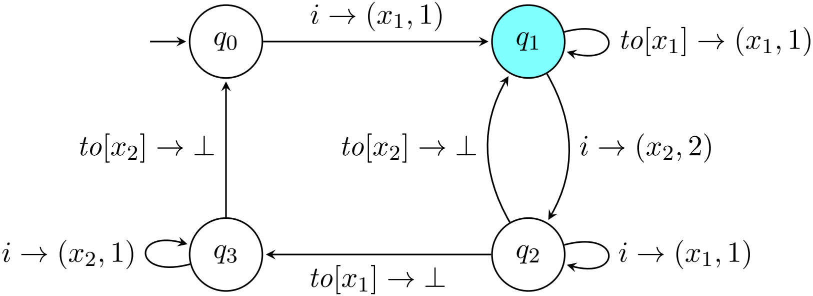


Figure 2: The same AT.

$$(q_0, \emptyset) \xrightarrow{1} (q_0, \emptyset) \xrightarrow[x_1,1]{i} (q_1, x_1 = 1)$$

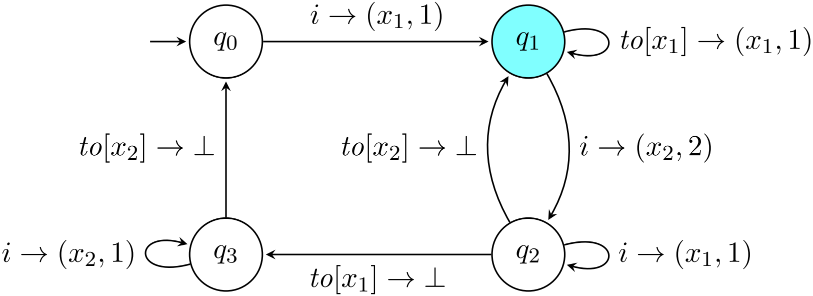


Figure 2: The same AT.

$$(q_0, \emptyset) \xrightarrow{1} (q_0, \emptyset) \xrightarrow[x_1,1]{i} (q_1, x_1 = 1) \xrightarrow{1} (q_1, x_1 = 0)$$

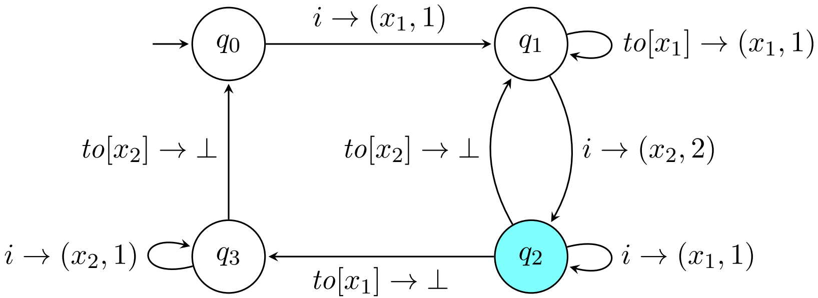


Figure 2: The same AT.

$$(q_0, \emptyset) \xrightarrow{1} (q_0, \emptyset) \xrightarrow[x_1,1]{i} (q_1, x_1 = 1) \xrightarrow{1} (q_1, x_1 = 0) \xrightarrow[x_2,2]{i} (q_2, x_1 = 0, x_2 = 2)$$

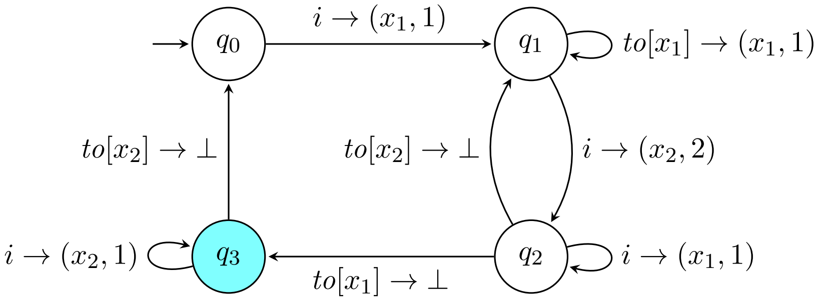


Figure 2: The same AT.

$$(q_0, \emptyset) \xrightarrow{1} (q_0, \emptyset) \xrightarrow[x_1,1]{i} (q_1, x_1 = 1) \xrightarrow{1} (q_1, x_1 = 0) \xrightarrow[x_2,2]{i} (q_2, x_1 = 0, x_2 = 2)$$

$$\xrightarrow{0} (q_2, x_1 = 0, x_2 = 2) \xrightarrow[\bot]{to[x_1]} (q_3, x_2 = 2)$$

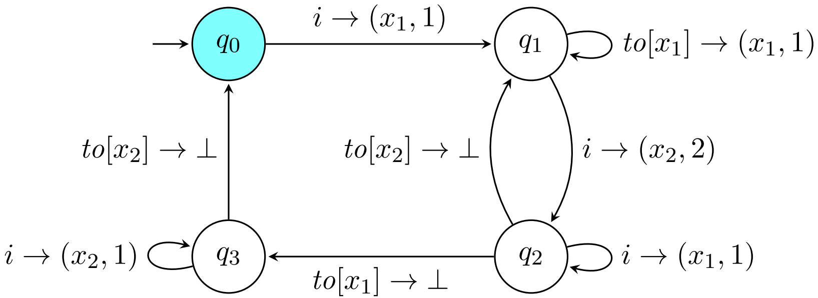


Figure 2: The same AT.

$$(q_0, \emptyset) \xrightarrow{1} (q_0, \emptyset) \xrightarrow[x_1,1]{i} (q_1, x_1 = 1) \xrightarrow{1} (q_1, x_1 = 0) \xrightarrow[x_2,2]{i} (q_2, x_1 = 0, x_2 = 2)$$
$$\xrightarrow{0} (q_2, x_1 = 0, x_2 = 2) \xrightarrow[\bot]{to[x_1]} (q_3, x_2 = 2) \xrightarrow{2} (q_3, x_2 = 0) \xrightarrow[\bot]{to[x_2]} (q_0, \emptyset) \xrightarrow{0.5} (q_0, \emptyset).$$

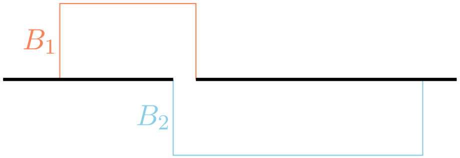


Figure 3: Block representation of the execution.

$$(q_0, \emptyset) \xrightarrow{1} (q_0, \emptyset) \xrightarrow[x_1, 1]{i} (q_1, x_1 = 1) \xrightarrow{1} (q_1, x_1 = 0) \xrightarrow[x_2, 2]{i} (q_2, x_1 = 0, x_2 = 2)$$

$$\xrightarrow{0} (q_2, x_1 = 0, x_2 = 2) \xrightarrow[\bot]{to[x_1]} (q_3, x_2 = 2) \xrightarrow{2} (q_3, x_2 = 0) \xrightarrow[\bot]{to[x_2]} (q_0, \emptyset) \xrightarrow{0.5} (q_0, \emptyset).$$

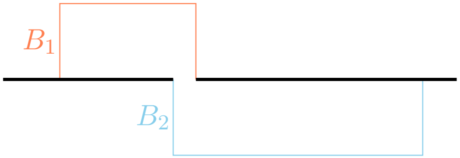


Figure 3: Block representation of the execution.

$$(q_0, \emptyset) \xrightarrow{1} (q_0, \emptyset) \xrightarrow[x_1,1]{i} (q_1, x_1 = 1) \xrightarrow{1} (q_1, x_1 = 0) \xrightarrow[x_2,2]{i} (q_2, x_1 = 0, x_2 = 2)$$

$$\xrightarrow{0} (q_2, x_1 = 0, x_2 = 2) \xrightarrow[\bot]{to[x_1]} (q_3, x_2 = 2) \xrightarrow{2} (q_3, x_2 = 0) \xrightarrow[\bot]{to[x_2]} (q_0, \emptyset) \xrightarrow{0.5} (q_0, \emptyset).$$

We have concurrent actions.

We can avoid this concurrency and still see the same sequence of actions.

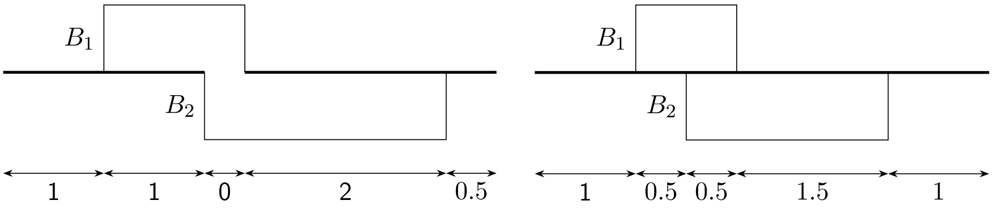


Figure 4: Idea: wiggle delays between actions.

We can avoid this concurrency and still see the same sequence of actions.

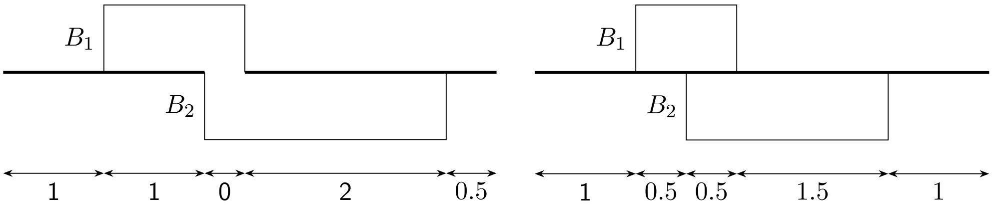


Figure 4: Idea: wiggle delays between actions.

Is it always possible?

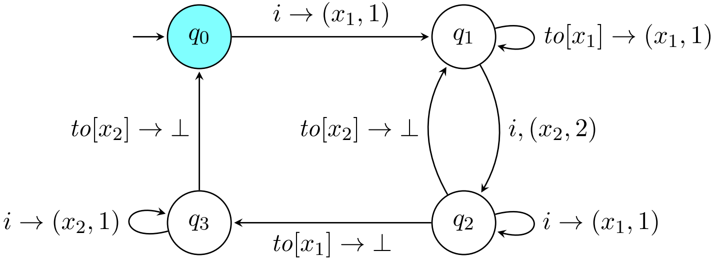


Figure 5: The same AT.

$(q_0, \emptyset)$

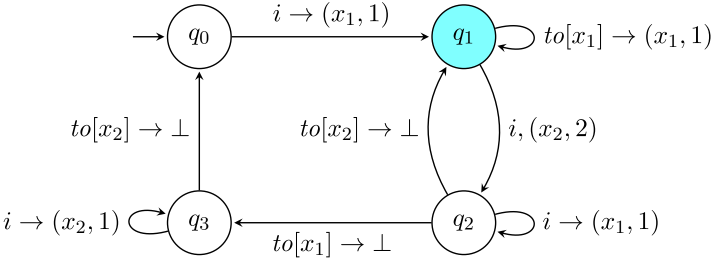


Figure 5: The same AT.

$$(q_0, \emptyset) \xrightarrow{1} (q_0, \emptyset) \xrightarrow[x_1,1]{i} (q_1, x_1 = 1)$$

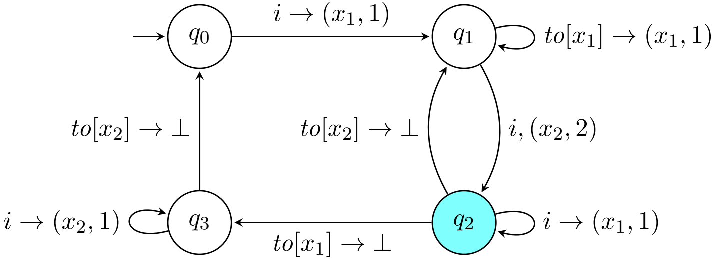


Figure 5: The same AT.

$$(q_0, \emptyset) \xrightarrow{1} (q_0, \emptyset) \xrightarrow[x_1, 1]{i} (q_1, x_1 = 1) \xrightarrow{0} (q_1, x_1 = 1) \xrightarrow[x_2, 2]{i} (q_2, x_1 = 1, x_2 = 2)$$

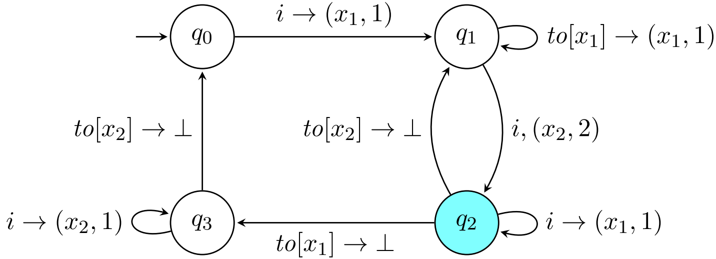


Figure 5: The same AT.

$$(q_0, \emptyset) \xrightarrow{1} (q_0, \emptyset) \xrightarrow[x_1,1]{i} (q_1, x_1 = 1) \xrightarrow{0} (q_1, x_1 = 1) \xrightarrow[x_2,2]{i} (q_2, x_1 = 1, x_2 = 2)$$
$$\xrightarrow{1} (q_2, x_1 = 0, x_2 = 1) \xrightarrow[x_1,1]{i} (q_2, x_1 = 1, x_2 = 1)$$

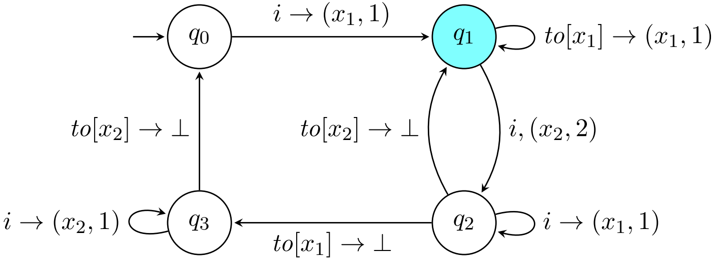


Figure 5: The same AT.

$$(q_0, \emptyset) \xrightarrow{1} (q_0, \emptyset) \xrightarrow[x_1,1]{i} (q_1, x_1 = 1) \xrightarrow{0} (q_1, x_1 = 1) \xrightarrow[x_2,2]{i} (q_2, x_1 = 1, x_2 = 2)$$

$$\xrightarrow{1} (q_2, x_1 = 0, x_2 = 1) \xrightarrow[x_1,1]{i} (q_2, x_1 = 1, x_2 = 1) \xrightarrow{1} (q_2, x_1 = 0, x_2 = 0)$$

$$\xrightarrow[\bot]{to[x_2]} (q_1, x_1 = 0)$$

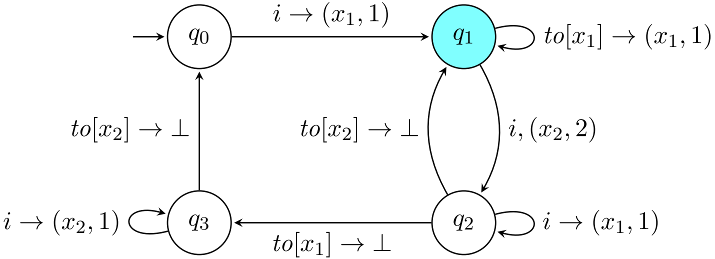


Figure 5: The same AT.

$$(q_0, \emptyset) \xrightarrow{1} (q_0, \emptyset) \xrightarrow[x_1,1]{i} (q_1, x_1 = 1) \xrightarrow{0} (q_1, x_1 = 1) \xrightarrow[x_2,2]{i} (q_2, x_1 = 1, x_2 = 2)$$

$$\xrightarrow{1} (q_2, x_1 = 0, x_2 = 1) \xrightarrow[x_1,1]{i} (q_2, x_1 = 1, x_2 = 1) \xrightarrow{1} (q_2, x_1 = 0, x_2 = 0)$$

$$\xrightarrow[\bot]{to[x_2]} (q_1, x_1 = 0) \xrightarrow{0} (q_1, x_1 = 0) \xrightarrow[x_1,1]{to[x_1]} (q_1, x_1 = 1) \xrightarrow{0.5} (q_1, x_1 = 0.5).$$

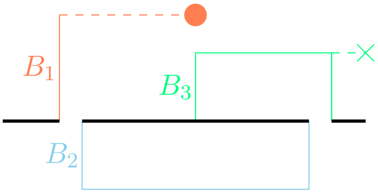


Figure 6: Block representation of the timed run.

$$(q_0, \emptyset) \xrightarrow{1} (q_0, \emptyset) \xrightarrow[x_1,1]{i} (q_1, x_1 = 1) \xrightarrow{0} (q_1, x_1 = 1) \xrightarrow[x_2,2]{i} (q_2, x_1 = 1, x_2 = 2)$$

$$\xrightarrow{1} (q_2, x_1 = 0, x_2 = 1) \xrightarrow[x_1,1]{i} (q_2, x_1 = 1, x_2 = 1) \xrightarrow{1} (q_2, x_1 = 0, x_2 = 0)$$

$$\xrightarrow[\bot]{to[x_2]} (q_1, x_1 = 0) \xrightarrow{0} (q_1, x_1 = 0) \xrightarrow[x_1,1]{to[x_1]} (q_1, x_1 = 1) \xrightarrow{0.5} (q_1, x_1 = 0.5).$$

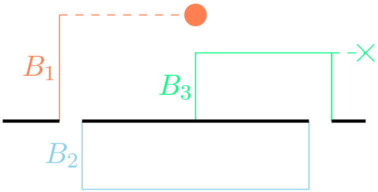


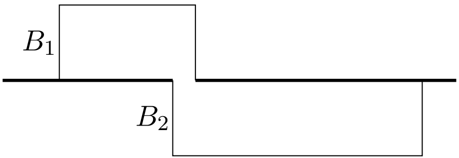
Figure 6: Block representation of the timed run.

$$(q_0, \emptyset) \xrightarrow{1} (q_0, \emptyset) \xrightarrow[x_1,1]{i} (q_1, x_1 = 1) \xrightarrow{0} (q_1, x_1 = 1) \xrightarrow[x_2,2]{i} (q_2, x_1 = 1, x_2 = 2)$$

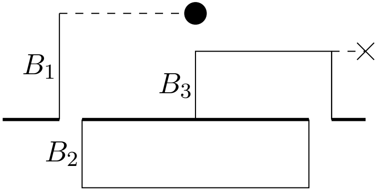
$$\xrightarrow{1} (q_2, x_1 = 0, x_2 = 1) \xrightarrow[x_1,1]{i} (q_2, x_1 = 1, x_2 = 1) \xrightarrow{1} (q_2, x_1 = 0, x_2 = 0)$$

$$\xrightarrow[\bot]{to[x_2]} (q_1, x_1 = 0) \xrightarrow{0} (q_1, x_1 = 0) \xrightarrow[x_1,1]{to[x_1]} (q_1, x_1 = 1) \xrightarrow{0.5} (q_1, x_1 = 0.5).$$

We cannot avoid this concurrency and still see the same sequence of actions.

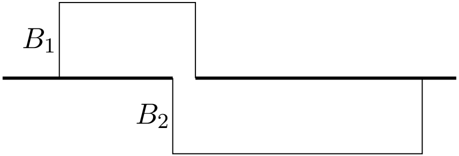


(a) Can be avoided.

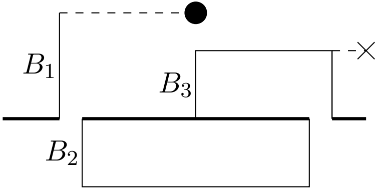


(b) Can not be avoided.

Figure 7: Some concurrency can be avoided, some not.



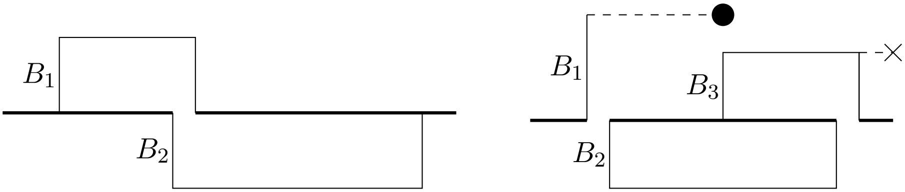
(a) Can be avoided.



(b) Can not be avoided.

Figure 7: Some concurrency can be avoided, some not.

Can we characterize when it is possible to remove the concurrency?



(a) Can be avoided. (b) Can not be avoided.

Figure 7: Some concurrency can be avoided, some not.

Can we characterize when it is possible to remove the concurrency?

Yes... But there is not enough time!

We studied two problems.

We studied two problems.

Theorem 1 (Contribution)

Fix an automaton and a state q. Deciding whether there exists an execution of the automaton that reaches q is PSPACE-*complete.*

We studied two problems.

Theorem 1 (Contribution)

Fix an automaton and a state q. Deciding whether there exists an execution of the automaton that reaches q is PSPACE*-complete.*

Theorem 2 (Contribution)

Deciding whether an AT contains an execution in which some concurrency can not be avoided is PSPACE*-hard and in* 3EXP.

We studied two problems.

Theorem 1 (Contribution)

Fix an automaton and a state q. Deciding whether there exists an execution of the automaton that reaches q is PSPACE*-complete.*

Theorem 2 (Contribution)

Deciding whether an AT contains an execution in which some concurrency can not be avoided is PSPACE*-hard and in* 3EXP*.*

Thank you!

For all details, see Bruyère et al., "Automata with Timers", 2023.

References I

- Angluin, Dana. "Learning Regular Sets from Queries and Counterexamples". In: *Inf. Comput.* 75.2 (1987), pp. 87–106. DOI: 10.1016/0890-5401(87)90052-6.
- Baier, Christel and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008. ISBN: 978-0-262-02649-9.
- Bruyère, Véronique et al. "Automata with Timers". In: *CoRR* abs/2305.07451 (2023). DOI: 10.48550/arXiv.2305.07451. arXiv: 2305.07451. URL: https://doi.org/10.48550/arXiv.2305.07451.
- Clarke, Edmund M. et al., eds. *Handbook of Model Checking*. Springer, 2018. ISBN: 978-3-319-10574-1. DOI: 10.1007/978-3-319-10575-8.