NASA Formal Methods 2022

26 May 2022

# Exemplifying parametric timed specifications over signals with bounded behavior

Étienne André, Masaki Waga, Natsuki Urabe and Ichiro Hasuo

Université de Lorraine, CNRS, Inria, LORIA, Nancy, France
Kyoto University, Kyoto, Japan
National Institute of Informatics, Tokyo, Japan
The Graduate University for Advanced Studies, SOKENDAI, Tokyo, Japan

# Outline

# Context: Specification

- Formal methods
  - Extremely useful for assessing the validity of specifications
  - Require some domain-specific technical expertise

# Context: Specification

- Formal methods
  - Extremely useful for assessing the validity of specifications
  - Require some domain-specific technical expertise

**Problem: writing correct specifications**

Even domain experts may do mistakes when writing models or properties.

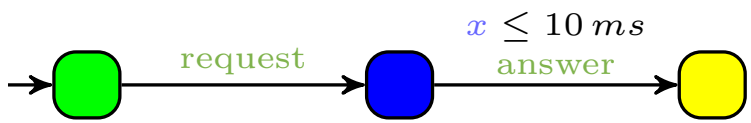
# Context: Specification

- Formal methods
  - Extremely useful for assessing the validity of specifications
  - Require some domain-specific technical expertise

**Problem: writing correct specifications**

Even domain experts may do mistakes when writing models or properties.

Example: “a request is followed by an answer exactly 10 ms later”



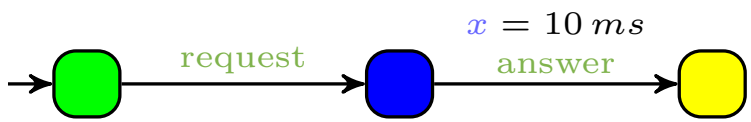
# Context: Specification

- Formal methods
  - Extremely useful for assessing the validity of specifications
  - Require some domain-specific technical expertise

**Problem: writing correct specifications**

Even domain experts may do mistakes when writing models or properties.

Example: “a request is followed by an answer exactly 10 ms later”



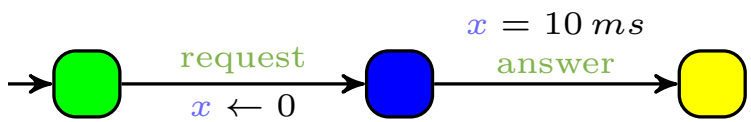
# Context: Specification

- Formal methods
  - Extremely useful for assessing the validity of specifications
  - Require some domain-specific technical expertise

**Problem: writing correct specifications**

Even domain experts may do mistakes when writing models or properties.

Example: “a request is followed by an answer exactly 10 ms later”



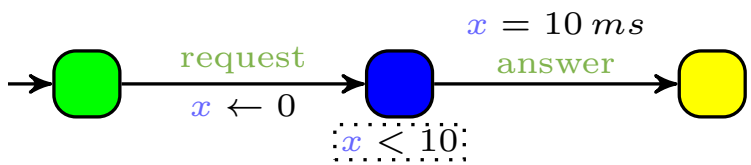
# Context: Specification

- Formal methods
  - Extremely useful for assessing the validity of specifications
  - Require some domain-specific technical expertise

**Problem: writing correct specifications**

Even domain experts may do mistakes when writing models or properties.

Example: “a request is followed by an answer exactly 10 ms later”



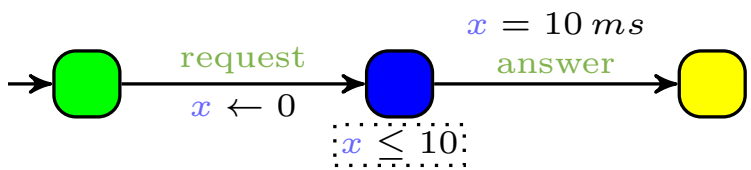
# Context: Specification

- Formal methods
  - Extremely useful for assessing the validity of specifications
  - Require some domain-specific technical expertise

**Problem: writing correct specifications**

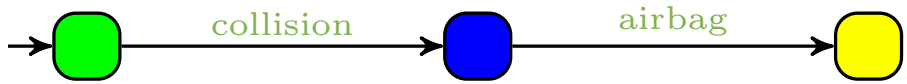
Even domain experts may do mistakes when writing models or properties.

Example: “a request is followed by an answer exactly 10 ms later”



# Context: Specifying quantitative properties

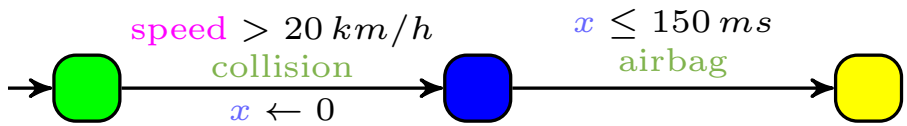
specifications



# Context: Specifying quantitative properties

Quantitative specifications

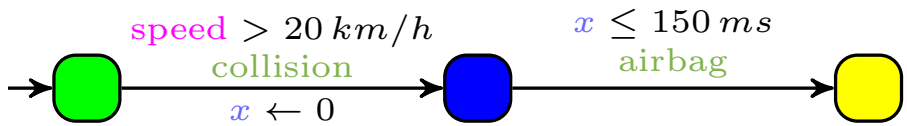
- Constants representing time
- Continuous signals (speed, temperature…)



# Context: Specifying quantitative properties

Quantitative specifications

- Constants representing time
- Continuous signals (speed, temperature…)



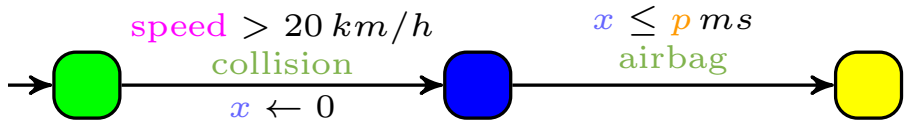

Problem: timed formal methods have some restrictions

- Need for more abstraction
  - Timed constants can be known with only finite precision
  - …or be even completely unknown

# Context: Specifying quantitative properties

Quantitative specifications

- Constants representing time (possibly parametric)
- Continuous signals (speed, temperature…)




Problem: timed formal methods have some restrictions

- Need for more abstraction
  - Timed constants can be known with only finite precision
  - …or be even completely unknown
- Idea: reason with parameters (unknown constants)
  1. Verify a system featuring unknown constants
  2. Synthesize correct parameter valuations

# Objectives: Assist designers

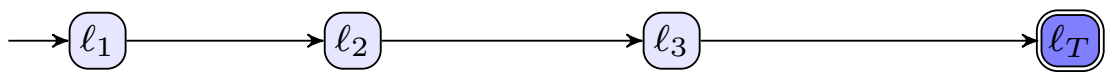
## Objectives

1. Propose a formalism for parametric timed specifications over signals
2. Automatically generate concrete executions exemplifying quantitative specifications

# Outline

# PTAS: parametric timed automata over signals

- 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

[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. "Parametric real-time reasoning". In: *STOC*. ACM, 1993, pp. 592–601. ISBN: 0-89791-591-7

# PTAS: parametric timed automata over signals

- 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

[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. “Parametric real-time reasoning”. In: *STOC*. ACM, 1993, pp. 592–601. ISBN: 0-89791-591-7

# PTAS: parametric timed automata over signals

- Finite-state automaton (sets of locations and actions) augmented with
  - a set 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

[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. "Parametric real-time reasoning". In: *STOC*. ACM, 1993, pp. 592–601. ISBN: 0-89791-591-7

# PTAS: parametric timed automata over signals

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

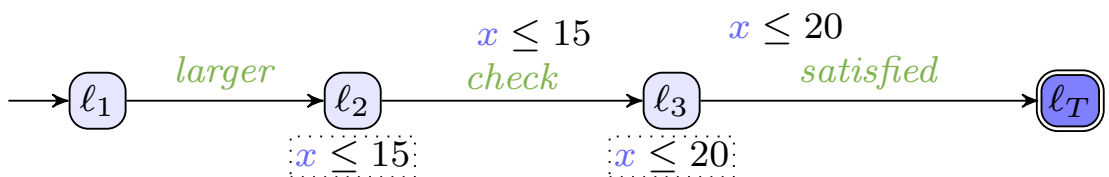


[AD94] Rajeev Alur and David L. Dill. "A theory of timed automata". In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235. ISSN: 0304-3975

[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. "Parametric real-time reasoning". In: *STOC*. ACM, 1993, pp. 592–601. ISBN: 0-89791-591-7

# PTAS: parametric timed automata over signals

- Finite-state automaton (sets of locations and actions) augmented with
  - a set of clocks [AD94]
    - Real-valued variables evolving linearly at the same rate
    - Can be compared to integer constants in invariants and guards
    - Location invariant: property to be verified to stay at a location
    - Transition guard: property to be verified to enable a transition



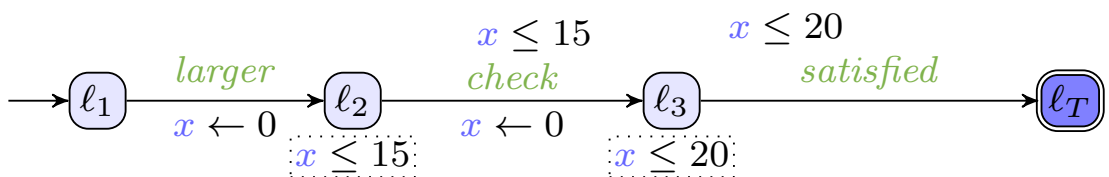


[AD94] Rajeev Alur and David L. Dill. "A theory of timed automata". In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235. ISSN: 0304-3975

[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. "Parametric real-time reasoning". In: *STOC*. ACM, 1993, pp. 592–601. ISBN: 0-89791-591-7

# PTAS: parametric timed automata over signals

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



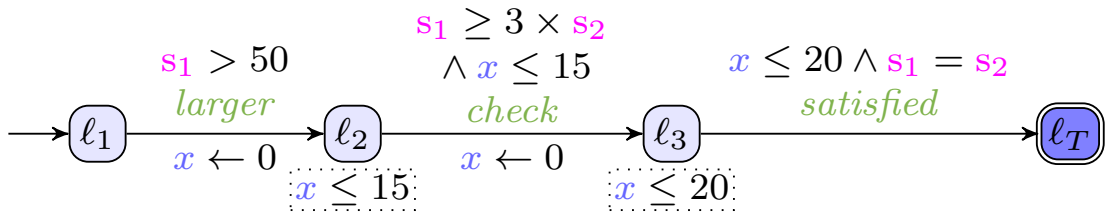


[AD94] Rajeev Alur and David L. Dill. "A theory of timed automata". In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235. ISSN: 0304-3975

[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. "Parametric real-time reasoning". In: *STOC*. ACM, 1993, pp. 592–601. ISBN: 0-89791-591-7

# PTAS: parametric timed automata over signals

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




"Whenever signal $s_1$ is larger than 50, then within at most 15 time units, it holds that $s_1 \geq 3 \times s_2$ and then, within at most 20 more time units, both signals are equal ($s_1 = s_2$)".

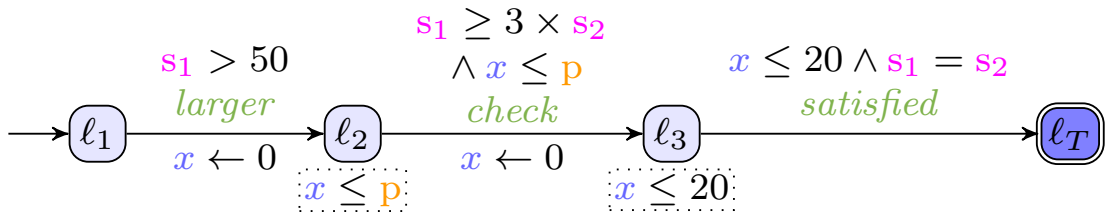
---


[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

[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. "Parametric real-time reasoning". In: *STOC*. ACM, 1993, pp. 592–601. ISBN: 0-89791-591-7

# PTAS: parametric timed automata over signals

- Finite-state automaton (sets of locations and actions) augmented with
  - a set of clocks [AD94]
    - Real-valued variables evolving linearly at the same rate
    - Can be compared to integer constants in invariants and guards
    - 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
  - a set of continuous signals
  - a set of (timing) parameters [AHV93]




"Whenever signal $s_1$ is larger than 50, then within at most $p$ time units, it holds that $s_1 \geq 3 \times s_2$ and then, within at most 20 more time units, both signals are equal ($s_1 = s_2$)".

---


[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

[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. "Parametric real-time reasoning". In: *STOC*. ACM, 1993, pp. 592–601. ISBN: 0-89791-591-7

# Outline

# Bounding behaviors

Looking for concrete behaviors:

- Do we want arbitrary behaviors, or more specific behaviors?
  - e. g., “look for a concrete run only in a scenario when a car decelerates”

# Bounding behaviors

Looking for concrete behaviors:

- Do we want arbitrary behaviors, or more specific behaviors?
  - e. g., “look for a concrete run only in a scenario when a car decelerates”

Crux: constrain the evolution of each signal along the exhibited run using an automaton

# Bounding behaviors

Looking for concrete behaviors:

- Do we want arbitrary behaviors, or more specific behaviors?
  - e. g., “look for a concrete run only in a scenario when a car decelerates”

Crux: constrain the evolution of each signal along the exhibited run using an automaton

Also useful for our approach to remain in the scope of linear constraints (polyhedra)

# Signal bounding automata

Here, each signal may be constrained by a signal bounding automaton (SBA)

- Timed automata augmented with arbitrary (rational) rates for signals
- Subclass of rectangular hybrid automata [Hen96]

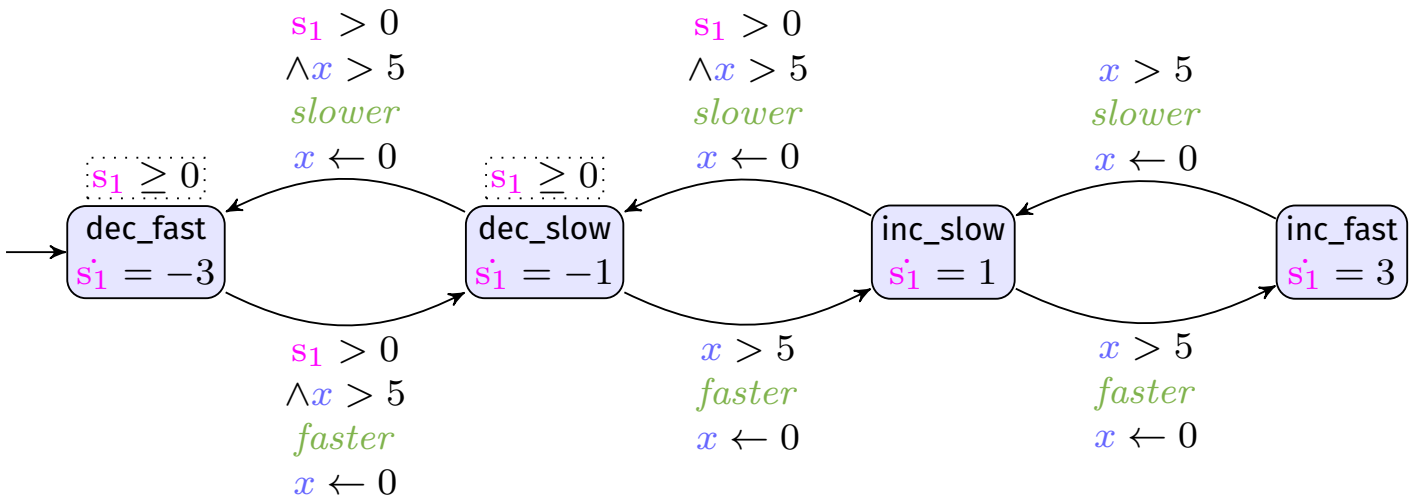
[Hen96] Thomas A. Henzinger. "The Theory of Hybrid Automata". In: *LICS*. IEEE Computer Society, 1996, pp. 278–292

# Signal bounding automata

Here, each signal may be constrained by a signal bounding automaton (SBA)

- Timed automata augmented with arbitrary (rational) rates for signals
- Subclass of rectangular hybrid automata [Hen96]

Example of SBA:



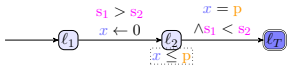

- Signal $s_1$ is always non-negative
- It can increase or decrease fast ($\dot{s_1} = \pm 3$) or slow ($\dot{s_1} = \pm 1$)
- Changes of dynamics never occur faster than every 5 time units

[Hen96] Thomas A. Henzinger. "The Theory of Hybrid Automata". In: *LICS*. IEEE Computer Society, 1996, pp. 278–292

# Our general approach

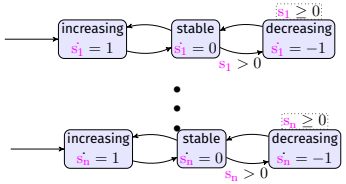
**Specification**

A PTAS $\mathcal{A}$ with $n$ signals




**Bounding behavior**

$n$ SBAs $\mathcal{A}_i, i \in \{1, \ldots, n\}$



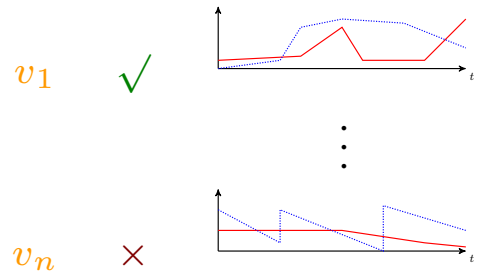

**Inputs**

A PLMA

$\mathcal{A} \parallel \mathcal{A}_1 \parallel \cdots \parallel \mathcal{A}_n$

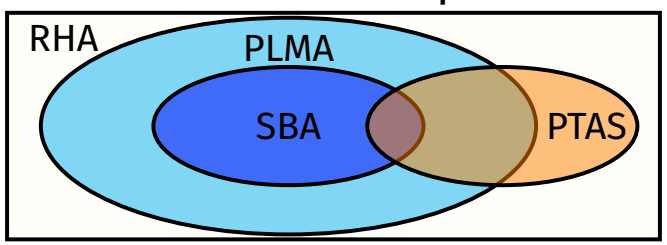
**Exemplification**

Set of concrete runs




**Outputs**

Formalisms manipulated




RHA = rectangular hybrid automata
PLMA = parametric linear multi-rate automata
PTAS = parametric timed automata with signals
SBA = signal bounding automata

# Methodology

1. Symbolic exploration of the state space
   - Underlying data structure: polyhedra over clocks, signals, parameters
2. Reachability analysis
3. Exhibition of a symbolic run
4. (Backward) reconstruction of a concrete run

Heuristics-based “best-effort” approach

(see paper for technical details)

# Outline

# Implementation

## Implementation in IMITATOR [And21]

- Model checker for extensions of parametric timed automata
- Our exemplification approach supports the full IMITATOR syntax
  - including extensions (global variables, etc.)
- Polyhedra operations performed using the Parma Polyhedra Library [BMZ08]
- Output: set of runs in the JSON syntax + (basic) graphical outputs

Reproductibility artifact available on `10.5281/zenodo.6382893`

(`v.3.3-alpha` "Cheese Caramel au beurre salé")

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

[BMZ08] Roberto Bagnara, Hill Patricia M., and Enea Zaffanella. "The Parma Polyhedra Library: Toward a Complete Set of Numerical Abstractions for the Analysis and Verification of Hardware and Software Systems". In: *Science of Computer Programming* 72.1–2 (2008), pp. 3–21

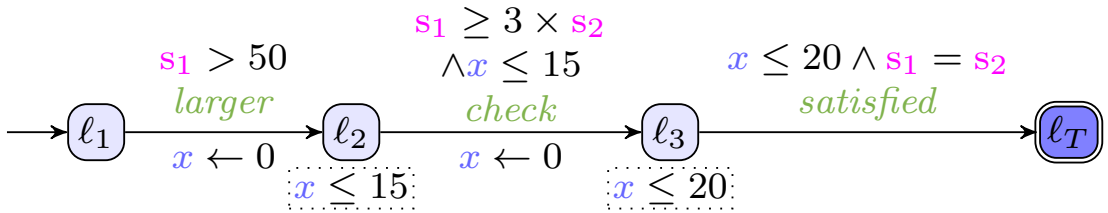
# Proof of concept

Specification: “Whenever signal $\mathrm{s}_1$ is larger than 50, then within at most 15 time units, it holds that $\mathrm{s}_1 \geq 3 \times \mathrm{s}_2$ and then, within at most 20 more time units, both signals are equal”.

# Proof of concept

Specification: “Whenever signal $s_1$ is larger than 50, then within at most 15 time units, it holds that $s_1 \geq 3 \times s_2$ and then, within at most 20 more time units, both signals are equal”.

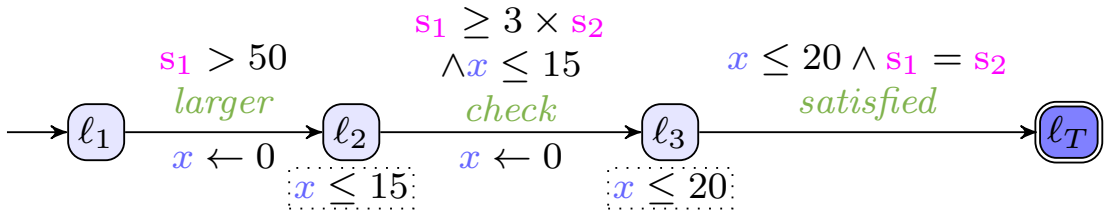
Encoding:



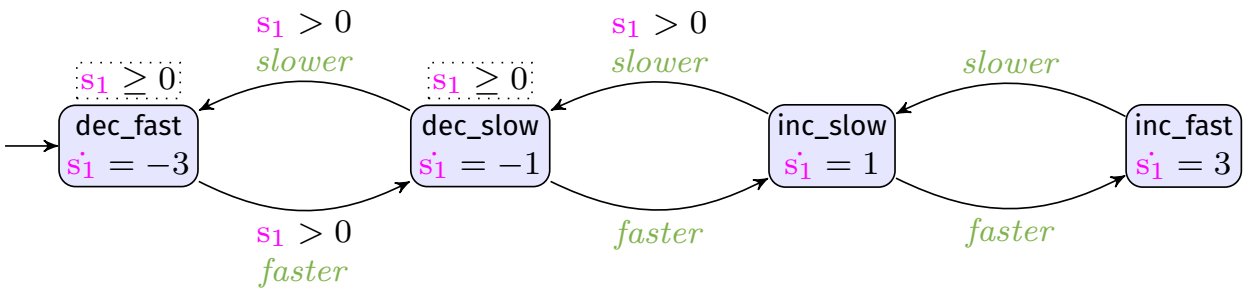
# Proof of concept

Specification: “Whenever signal $s_1$ is larger than 50, then within at most 15 time units, it holds that $s_1 \geq 3 \times s_2$ and then, within at most 20 more time units, both signals are equal”.

Encoding:



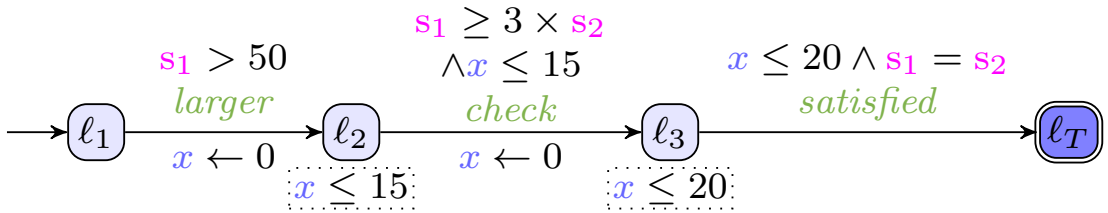

Signal bounding automata:



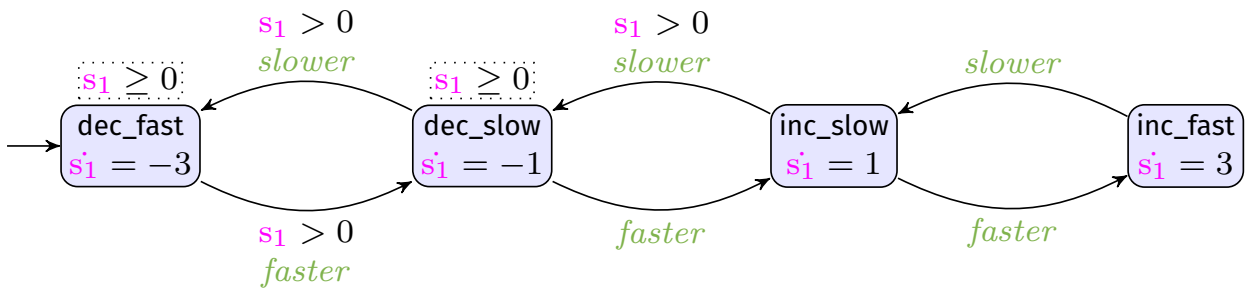
# Proof of concept

Specification: "Whenever signal $s_1$ is larger than 50, then within at most 15 time units, it holds that $s_1 \geq 3 \times s_2$ and then, within at most 20 more time units, both signals are equal".

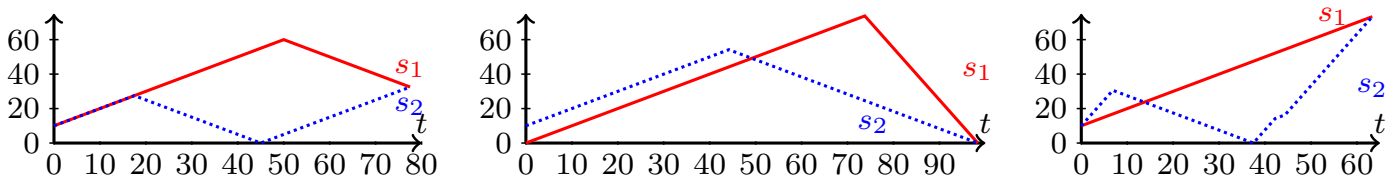
Encoding:




Signal bounding automata:




Automatic generation of 3 possible runs:



# Outline

# Conclusion

Best-effort (heuristic) approach to exemplify parametric specifications over signals

Input formalism: parametric linear multi-rate automata

- real-valued continuous variables with a piecewise-constant rate
- TA clocks
- timing parameters

Crux: signal bounding automata to limit the admissible continuous behavior

Implementation in IMITATOR

- Fully automated process

We also generate negative (impossible) executions (see paper)

# Perspectives

- Enhance efficiency using heuristics (e. g., [AA22])
- Extension to liveness/fairness
- Extension to logics such as LTL, MITL or STL [RHM17][PLK18]
- Expressiveness of our input formalism
- Providing some coverage guarantees
  - sufficient number of positive and negative runs
  - “cornercase” runs
- Evaluation on students or engineers who are not familiar with formal specifications
  - Complexity of the automata theory?

---


[AA22] Johan Arcile and Étienne André. “Zone extrapolations in parametric timed automata”. In: *NFM*. vol. 13260. LNCS. Springer, 2022, pp. 451–469

[RHM17] Hendrik Roehm, Thomas Heinz, and Eva Charlotte Mayer. “STLInspector: STL Validation with Guarantees”. In: *CAV, Part I*. vol. 10426. LNCS. Springer, 2017, pp. 225–232

[PLK18] Pavithra Prabhakar, Ratan Lal, and James Kapinski. “Automatic Trace Generation for Signal Temporal Logic”. In: *RTSS*. IEEE Computer Society, 2018, pp. 208–217

# Bibliography

# References I


Johan Arcile and Étienne André. “Zone extrapolations in parametric timed automata”. In: *NFM* (May 24–27, 2022). Ed. by Klaus Havelund, Jyo Deshmukh, and Ivan Perez. Vol. 13260. LNCS. Caltech, Pasadena, CA, USA: Springer, 2022, pp. 451–469. DOI: 10.1007/978-3-031-06773-0_24.

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.

Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. “Parametric real-time reasoning”. In: *STOC* (May 16–18, 1993). Ed. by S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal. San Diego, California, United States: ACM, 1993, pp. 592–601. ISBN: 0-89791-591-7. DOI: 10.1145/167088.167242.

Étienne André, Masaki Waga, Natsuki Urabe, and Ichiro Hasuo. “Exemplifying parametric timed specifications over signals with bounded behavior”. In: *NFM* (May 24–27, 2022). Ed. by Klaus Havelund, Jyo Deshmukh, and Ivan Perez. Vol. 13260. LNCS. Caltech, Pasadena, CA, USA: Springer, 2022. DOI: 10.1007/978-3-031-06773-0_25.

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

# References II


Roberto Bagnara, Hill Patricia M., and Enea Zaffanella. “The Parma Polyhedra Library: Toward a Complete Set of Numerical Abstractions for the Analysis and Verification of Hardware and Software Systems”. In: *Science of Computer Programming* 72.1–2 (2008), pp. 3–21. DOI: 10.1016/j.scico.2007.08.001.

Thomas A. Henzinger. “The Theory of Hybrid Automata”. In: *LICS* (July 27–30, 1996). New Brunswick, New Jersey, USA: IEEE Computer Society, 1996, pp. 278–292. DOI: 10.1109/LICS.1996.561342.

Pavithra Prabhakar, Ratan Lal, and James Kapinski. “Automatic Trace Generation for Signal Temporal Logic”. In: *RTSS* (Dec. 11–Nov. 14, 2018). Nashville, TN, USA: IEEE Computer Society, 2018, pp. 208–217. DOI: 10.1109/RTSS.2018.00038.

Hendrik Roehm, Thomas Heinz, and Eva Charlotte Mayer. “STLInspector: STL Validation with Guarantees”. In: *CAV, Part I* (July 24–28, 2017). Ed. by Rupak Majumdar and Viktor Kuncak. Vol. 10426. LNCS. Heidelberg, Germany: Springer, 2017, pp. 225–232. DOI: 10.1007/978-3-319-63387-9_11.

# Licensing

# License of this document



Authors: **Étienne André**