RP 2015

September 22nd, 2015
Warszawa

Integer-Complete Parameter Synthesis for Bounded Parametric Timed Automata

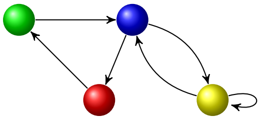
Étienne André[1,2], Didier Lime[2], Olivier H. Roux[2]

[1] LIPN, Université Paris 13, Sorbonne Paris Cité, CNRS, France
[2] École Centrale de Nantes, IRCCyN, CNRS, UMR 6597, France

PROJET FINANCÉ PAR L'ANR - PROJECT FUNDED BY THE ANR

Context: Formal verification of timed systems

- Model checking



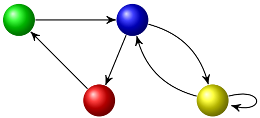
A model of the system

 is unreachable

A property to be satisfied

Context: Formal verification of timed systems

- Model checking



$\overset{?}{\models}$

 is unreachable

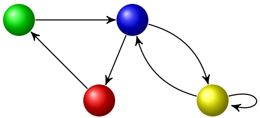
A model of the system

A property to be satisfied

- Question: does the model of the system satisfy the property?

Context: Formal verification of timed systems

- Model checking



$\overset{?}{\models}$

 is unreachable

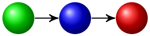
A model of the system

A property to be satisfied

- Question: does the model of the system satisfy the property?

Yes

No



Counterexample

Beyond model checking: parameter synthesis

- Timed systems are characterized by a set of timing constants
  - "The packet transmission lasts for 50 ms"
  - "The sensor reads the value every 10 s"
- Verification for one set of constants does not usually guarantee the correctness for other values
- Challenges
  - Numerous verifications: is the system correct for any value within $[40;60]$?
  - Optimization: until what value can we increase 10?
  - Robustness [Markey, 2011]: What happens if 50 is implemented with 49.99?

Beyond model checking: parameter synthesis

- Timed systems are characterized by a set of timing constants
  - "The packet transmission lasts for $50\,ms$"
  - "The sensor reads the value every $10\,s$"

- Verification for one set of constants does not usually guarantee the correctness for other values

- Challenges
  - Numerous verifications: is the system correct for any value within $[40; 60]$?
  - Optimization: until what value can we increase 10?
  - Robustness [Markey, 2011]: What happens if 50 is implemented with 49.99?

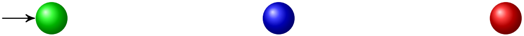
- Parameter synthesis
  - Consider that timing constants are unknown constants (parameters)
  - Find good values for the parameters

Outline

Outline

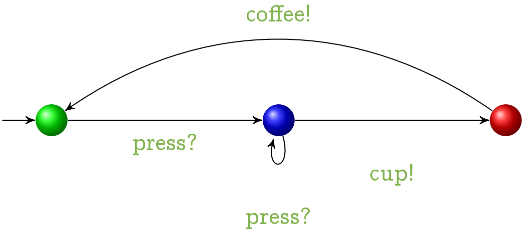
Timed automaton (TA)

- Finite state automaton (sets of locations)



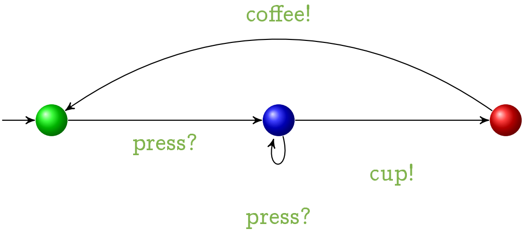
Timed automaton (TA)

- Finite state automaton (sets of locations and actions)



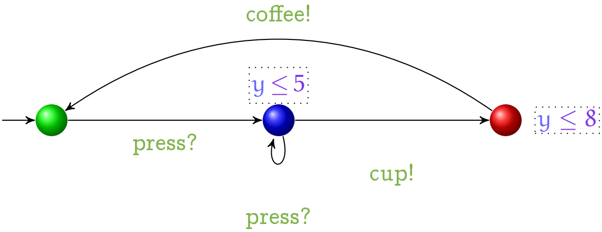
Timed automaton (TA)

- Finite state automaton (sets of locations and actions) augmented with a set X of clocks [Alur and Dill, 1994]
  - Real-valued variables evolving linearly at the same rate



Timed automaton (TA)

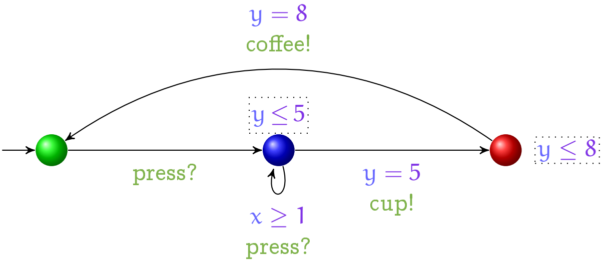
- Finite state automaton (sets of locations and actions) augmented with a set X of clocks [Alur and Dill, 1994]
  - Real-valued variables evolving linearly at the same rate
- Features
  - Location invariant: property to be verified to stay at a location



Timed automaton (TA)

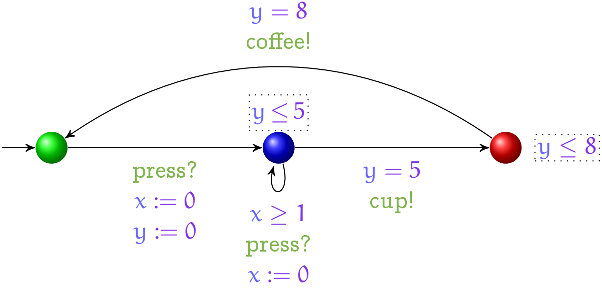
- Finite state automaton (sets of locations and actions) augmented with a set X of clocks [Alur and Dill, 1994]
  - Real-valued variables evolving linearly at the same rate

- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property to be verified to enable a transition



Timed automaton (TA)

- Finite state automaton (sets of locations and actions) augmented with a set X of clocks [Alur and Dill, 1994]
  - Real-valued variables evolving linearly at the same rate
- 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 at each transition



Concrete semantics of timed automata

- Concrete state of a TA: pair (l, w), where
  - l is a location,
  - w is a valuation of each clock

- Concrete run: alternating sequence of concrete states and actions or elapsing of time
  - Possible concrete runs for the coffee machine

Concrete semantics of timed automata

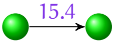
- Concrete state of a TA: pair (l, w), where
  - l is a location,
  - w is a valuation of each clock

- Concrete run: alternating sequence of concrete states and actions or elapsing of time
  - Possible concrete runs for the coffee machine
    - Coffee with no sugar

x	0
y	0

Concrete semantics of timed automata

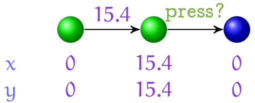
- Concrete state of a TA: pair (l, w), where
  - l is a location,
  - w is a valuation of each clock
- Concrete run: alternating sequence of concrete states and actions or elapsing of time
  - Possible concrete runs for the coffee machine
    - Coffee with no sugar



x	0	15.4
y	0	15.4

Concrete semantics of timed automata

- Concrete state of a TA: pair (l, w), where
  - l is a location,
  - w is a valuation of each clock
- Concrete run: alternating sequence of concrete states and actions or elapsing of time
  - Possible concrete runs for the coffee machine
    - Coffee with no sugar



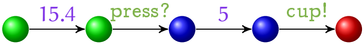
Concrete semantics of timed automata

- Concrete state of a TA: pair (l, w), where
  - l is a location,
  - w is a valuation of each clock
- Concrete run: alternating sequence of concrete states and actions or elapsing of time
  - Possible concrete runs for the coffee machine
    - Coffee with no sugar

x	0	15.4	0	5
y	0	15.4	0	5

Concrete semantics of timed automata

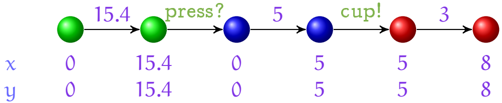
- Concrete state of a TA: pair (l, w), where
  - l is a location,
  - w is a valuation of each clock
- Concrete run: alternating sequence of concrete states and actions or elapsing of time
  - Possible concrete runs for the coffee machine
    - Coffee with no sugar



x	0	15.4	0	5	5
y	0	15.4	0	5	5

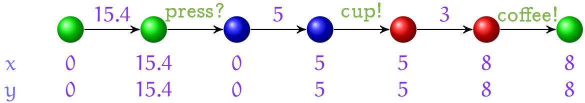
Concrete semantics of timed automata

- Concrete state of a TA: pair (l, w), where
  - l is a location,
  - w is a valuation of each clock
- Concrete run: alternating sequence of concrete states and actions or elapsing of time
  - Possible concrete runs for the coffee machine
    - Coffee with no sugar



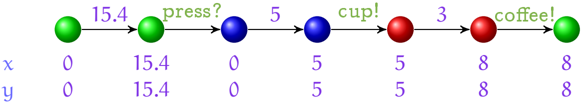
Concrete semantics of timed automata

- Concrete state of a TA: pair (l, w), where
  - l is a location,
  - w is a valuation of each clock
- Concrete run: alternating sequence of concrete states and actions or elapsing of time
  - Possible concrete runs for the coffee machine
    - Coffee with no sugar

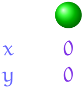


Concrete semantics of timed automata

- Concrete state of a TA: pair (l, w), where
  - l is a location,
  - w is a valuation of each clock
- Concrete run: alternating sequence of concrete states and actions or elapsing of time
  - Possible concrete runs for the coffee machine
    - Coffee with no sugar



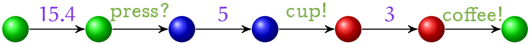
    - Coffee with 2 doses of sugar



Concrete semantics of timed automata

- Concrete state of a TA: pair (l, w), where
  - l is a location,
  - w is a valuation of each clock

- Concrete run: alternating sequence of concrete states and actions or elapsing of time
  - Possible concrete runs for the coffee machine
    - Coffee with no sugar



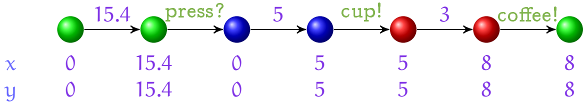
x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

    - Coffee with 2 doses of sugar

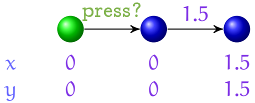
x	0	0
y	0	0

Concrete semantics of timed automata

- Concrete state of a TA: pair (l, w), where
  - l is a location,
  - w is a valuation of each clock
- Concrete run: alternating sequence of concrete states and actions or elapsing of time
  - Possible concrete runs for the coffee machine
    - Coffee with no sugar

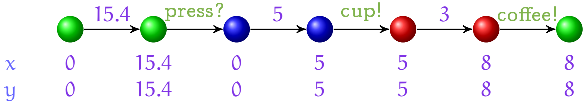


    - Coffee with 2 doses of sugar

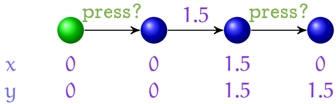


Concrete semantics of timed automata

- Concrete state of a TA: pair (l, w), where
  - l is a location,
  - w is a valuation of each clock
- Concrete run: alternating sequence of concrete states and actions or elapsing of time
  - Possible concrete runs for the coffee machine
    - Coffee with no sugar

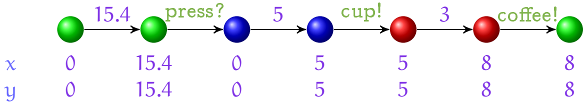


    - Coffee with 2 doses of sugar

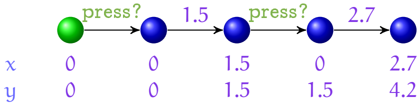


Concrete semantics of timed automata

- Concrete state of a TA: pair (l, w), where
  - l is a location,
  - w is a valuation of each clock
- Concrete run: alternating sequence of concrete states and actions or elapsing of time
  - Possible concrete runs for the coffee machine
    - Coffee with no sugar

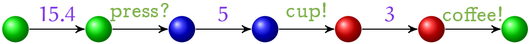


    - Coffee with 2 doses of sugar



Concrete semantics of timed automata

- Concrete state of a TA: pair (l, w), where
  - l is a location,
  - w is a valuation of each clock
- Concrete run: alternating sequence of concrete states and actions or elapsing of time
  - Possible concrete runs for the coffee machine
    - Coffee with no sugar



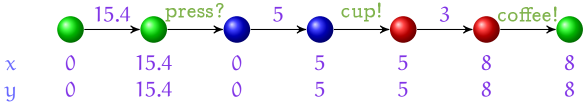
x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

    - Coffee with 2 doses of sugar

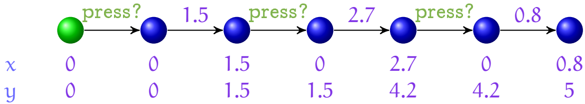
x	0	0	1.5	0	2.7	0
y	0	0	1.5	1.5	4.2	4.2

Concrete semantics of timed automata

- Concrete state of a TA: pair (l, w), where
  - l is a location,
  - w is a valuation of each clock
- Concrete run: alternating sequence of concrete states and actions or elapsing of time
  - Possible concrete runs for the coffee machine
    - Coffee with no sugar



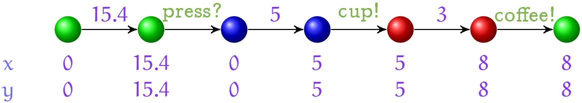
    - Coffee with 2 doses of sugar



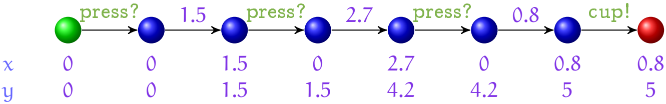
Concrete semantics of timed automata

- Concrete state of a TA: pair (l, w), where
  - l is a location,
  - w is a valuation of each clock

- Concrete run: alternating sequence of concrete states and actions or elapsing of time
  - Possible concrete runs for the coffee machine
    - Coffee with no sugar



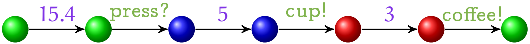
    - Coffee with 2 doses of sugar



Concrete semantics of timed automata

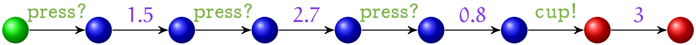
- Concrete state of a TA: pair (l, w), where
  - l is a location,
  - w is a valuation of each clock

- Concrete run: alternating sequence of concrete states and actions or elapsing of time
  - Possible concrete runs for the coffee machine
    - Coffee with no sugar



x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

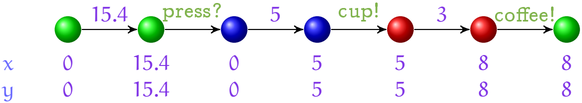
    - Coffee with 2 doses of sugar



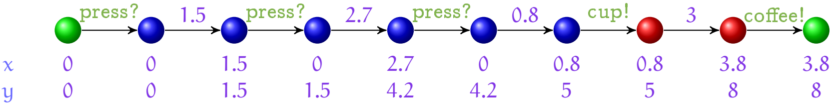
x	0	0	1.5	0	2.7	0	0.8	0.8	3.8
y	0	0	1.5	1.5	4.2	4.2	5	5	8

Concrete semantics of timed automata

- Concrete state of a TA: pair (l, w), where
  - l is a location,
  - w is a valuation of each clock
- Concrete run: alternating sequence of concrete states and actions or elapsing of time
  - Possible concrete runs for the coffee machine
    - Coffee with no sugar

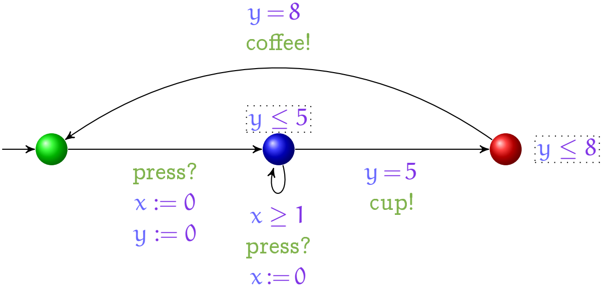


    - Coffee with 2 doses of sugar



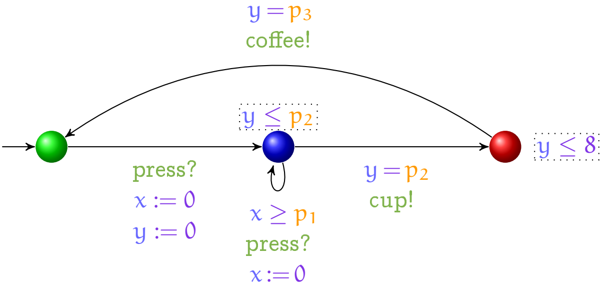
Parametric Timed Automaton (PTA)

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



Parametric Timed Automaton (PTA)

- Timed automaton (sets of locations, actions and clocks) augmented with a set P of parameters [Alur et al., 1993]
  - Unknown constants used in guards and invariants



Symbolic semantics of a PTA

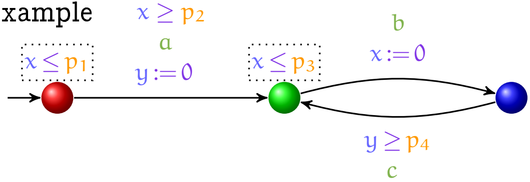
- Symbolic state of a PTA: pair (l, C), where
  - l is a location,
  - C is a polyhedron (conjunction of inequalities) over X and P

Symbolic semantics of a PTA

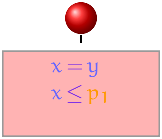
- Symbolic state of a PTA: pair (l, C), where
  - l is a location,
  - C is a polyhedron (conjunction of inequalities) over X and P
- Symbolic run: alternating sequence of symbolic states and actions

Symbolic semantics of a PTA

- Symbolic state of a PTA: pair (l, C), where
  - l is a location,
  - C is a polyhedron (conjunction of inequalities) over X and P
- Symbolic run: alternating sequence of symbolic states and actions
- Example

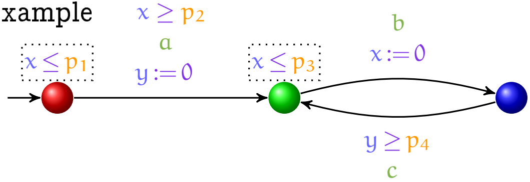


  - Possible symbolic run for this PTA

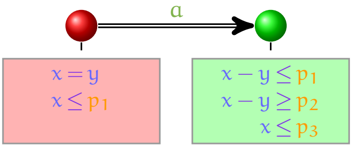


Symbolic semantics of a PTA

- Symbolic state of a PTA: pair (l, C), where
  - l is a location,
  - C is a polyhedron (conjunction of inequalities) over X and P
- Symbolic run: alternating sequence of symbolic states and actions
- Example

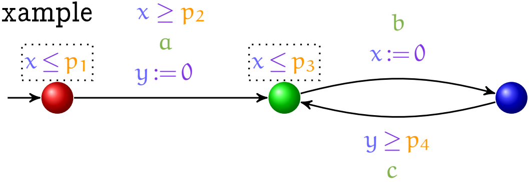


  - Possible symbolic run for this PTA

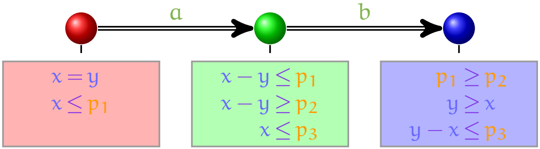


Symbolic semantics of a PTA

- Symbolic state of a PTA: pair (l, C), where
  - l is a location,
  - C is a polyhedron (conjunction of inequalities) over X and P
- Symbolic run: alternating sequence of symbolic states and actions
- Example



  - Possible symbolic run for this PTA



Valuation of a PTA

- Given a PTA $\mathcal{A}$ and a parameter valuation v, we denote by $v(\mathcal{A})$ the (non-parametric) timed automaton where all parameters are valuated by v

Objective: Computation problems

Definition (reachability synthesis (EF))

Input: a PTA $\mathcal{A}$ and a set of locations G

Problem: Synthesize all parameter valuations v such that there exists a run of $v(\mathcal{A})$ reaching a location $l \in G$

Objective: Computation problems

Definition (reachability synthesis (EF))

Input: a PTA $\mathcal{A}$ and a set of locations G
Problem: Synthesize all parameter valuations v such that there exists a run of $v(\mathcal{A})$ reaching a location $l \in G$

Definition (unavoidability synthesis (AF))

Input: a PTA $\mathcal{A}$ and a set of locations G
Problem: Synthesize all parameter valuations v such that all runs of $v(\mathcal{A})$ eventually reach a location $l \in G$

Outline

Decidability results: reachability

Reachability emptiness

Reachability emptiness ("does there exist at least one parameter valuation reaching a given location l?") is undecidable for PTA [Alur et al., 1993]

- even with a single parametric clock [Miller, 2000]
- even with only strict constraints [Doyen, 2007]
- even with a single integer-valued parameter [Beneš et al., 2015]

Decidability results: unavoidability

Reachability emptiness

Unavoidability emptiness ("does there exist at least one parameter valuation such that all runs reach a given location l?") is undecidable for PTA, even with a single bounded parameter

[Jovanović et al., 2015]

Synthesis of bounded integers

What if parameters are bounded integers. . . ?

Synthesis of bounded integers

What if parameters are bounded integers. . . ?

Bounded integers

Reachability and unavoidability emptiness are decidable (and PSPACE-complete) for PTA with bounded integers [Jovanović et al., 2015]

Synthesis of bounded integers

What if parameters are bounded integers. . . ?

Bounded integers

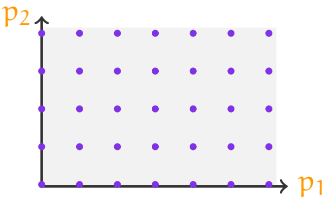
Reachability and unavoidability emptiness are decidable (and PSPACE-complete) for PTA with bounded integers [Jovanović et al., 2015]

Two algorithms:

- IEF: reachability synthesis
- IAF: unavoidability synthesis

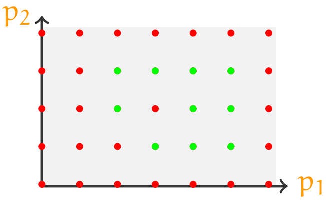
Synthesis of bounded integers: How?

Naive idea: enumerate all integers, and check the TA (which is PSPACE-complete [Alur and Dill, 1994])



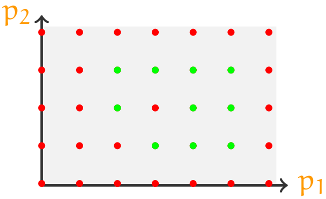
Synthesis of bounded integers: How?

Naive idea: enumerate all integers, and check the TA (which is PSPACE-complete [Alur and Dill, 1994])



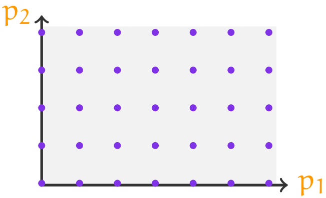
Synthesis of bounded integers: How?

Naive idea: enumerate all integers, and check the TA (which is PSPACE-complete [Alur and Dill, 1994])



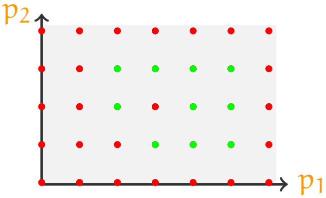
Smarter: symbolic algorithm [Jovanović et al., 2015]

- More efficient than exhaustive enumeration with Uppaal



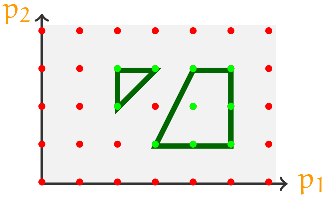
Synthesis of bounded integers: How?

Naive idea: enumerate all integers, and check the TA (which is PSPACE-complete [Alur and Dill, 1994])



Smarter: symbolic algorithm [Jovanović et al., 2015]

- More efficient than exhaustive enumeration with UPPAAL



Integer hull of a polyhedron

Definition (integer hull)

Let C be a polyhedron.

The integer hull of C is

$$\mathsf{IH}(C) = \mathsf{Conv}(\mathsf{IV}(C))$$

(Conv: convex hull; IV set of vectors with integer coordinates)

Integer hull of a polyhedron

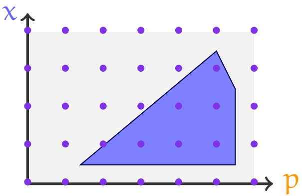
Definition (integer hull)

Let C be a polyhedron.

The integer hull of C is

$$\mathsf{IH}(C) = \mathsf{Conv}(\mathsf{IV}(C))$$

(Conv: convex hull; IV set of vectors with integer coordinates)



Integer hull of a polyhedron

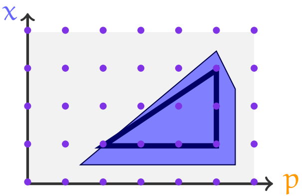
Definition (integer hull)

Let C be a polyhedron.

The integer hull of C is

$$\mathsf{IH}(C) = \mathsf{Conv}(\mathsf{IV}(C))$$

(Conv: convex hull; IV set of vectors with integer coordinates)



Reachability synthesis

Algorithm $\mathsf{EF}(\mathcal{A}, G)$

$K \leftarrow \bot$
Add the initial state to the waiting list

while the waiting list is not empty
 Pick a symbolic state (l, C) from the waiting list
 if $l \in G$ **then** $K \leftarrow K \vee \quad C \downarrow_P$
 else if $(l, \quad C \ ) = (l', \quad C' \ )$, for some (l', C') met before
  then do not explore further this branch
 else store $(l, \quad C \ )$ and add its successors to the waiting list

return K

Reachability synthesis of bounded integers using IH

Algorithm $\mathsf{IEF}(\mathcal{A}, G)$ [Jovanović et al., 2015]

$K \leftarrow \bot$

Add the initial state to the waiting list

while the waiting list is not empty

Pick a symbolic state (l, C) from the waiting list

if $l \in G$ **then** $K \leftarrow K \vee \mathsf{IH}(C)\downarrow_P$

else if $(l, \mathsf{IH}(C)) = (l', \mathsf{IH}(C'))$, for some (l', C') met before

then do not explore further this branch

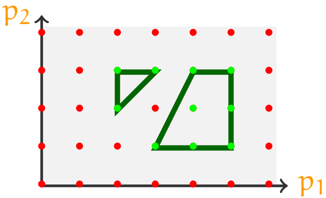
else store $(l, \mathsf{IH}(C))$ and add its successors to the waiting list

return K

Outline

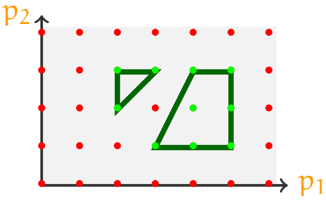
What about the dense result?

IEF and IAF return symbolic sets of integer valuations



What about the dense result?

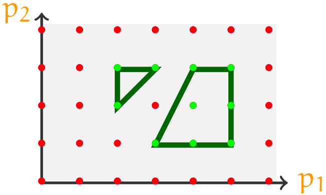
IEF and IAF return symbolic sets of integer valuations



Can we interpret the result of IEF and IAF over dense parameter valuations?

What about the dense result?

IEF and IAF return symbolic sets of integer valuations



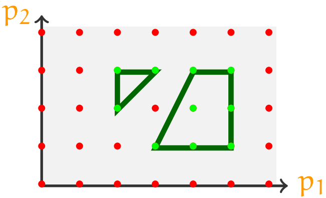
Can we interpret the result of IEF and IAF over dense parameter valuations?

- ☹ For IEF: yes! ... but it may not terminate

(example in paper)

What about the dense result?

IEF and IAF return symbolic sets of integer valuations



Can we interpret the result of IEF and IAF over dense parameter valuations?

- ☹ For IEF: yes! ...but it may not terminate

  (example in paper)

- ☹ For IAF: no! May yield incorrect valuations

  (counter-example in paper)

A parametric extrapolation for PTA

Definition (M-extrapolation)

Let M be the largest constant in $\mathcal{A}$ (including the bounds on the parameters), let x be a clock.
The (M, x)-extrapolation is

$$\mathsf{Ext}_x^M(C) = \big(C \cap (x \leq M)\big) \cup \mathsf{Cyl}_x\big(C \cap (x > M)\big) \cap (x > M).$$

A parametric extrapolation for PTA

Definition (M-extrapolation)

Let M be the largest constant in $\mathcal{A}$ (including the bounds on the parameters), let x be a clock.
The (M, x)-extrapolation is

$$\mathsf{Ext}_x^M(C) = \big(C \cap (x \leq M)\big) \cup \mathsf{Cyl}_x\big(C \cap (x > M)\big) \cap (x > M).$$

Generalized to (M, X)-extrapolation by applying to all clocks.

Integer reachability synthesis

Algorithm $\mathsf{IEF}(\mathcal{A}, G)$

$K \leftarrow \bot$
Add the initial state to the waiting list

while the waiting list is not empty
    Pick a symbolic state (l, C) from the waiting list
    if $l \in G$ **then** $K \leftarrow K \vee \mathsf{IH}(C)\downarrow_P$
    else if $(l, \mathsf{IH}(C)) = (l', \mathsf{IH}(C'))$,
        for some (l', C') met before
        then do not explore further this branch
    else store $(l, \mathsf{IH}(C))$ and add its successors to the waiting list

return K

Integer complete reachability synthesis RIEF

Algorithm $\mathsf{RIEF}(\mathcal{A}, G)$

$K \leftarrow \bot$

Add the initial state to the waiting list

while the waiting list is not empty

Pick a symbolic state (l, C) from the waiting list

if $l \in G$ **then** $K \leftarrow K \vee \quad C \downarrow_P$

else if $\big(l, \mathsf{IH}\big(\mathsf{Ext}_X^M(C)\big)\big) = \big(l', \mathsf{IH}\big(\mathsf{Ext}_X^M(C')\big)\big)$,

for some (l', C') met before

then do not explore further this branch

else store $\big(l, \mathsf{IH}(\mathsf{Ext}_X^M(C))\big)$ and add its successors to the waiting list

return K

Termination of RIEF

Theorem

For any PTA $\mathcal{A}$ with bounded parameters, the computation of RIEF$(\mathcal{A}, G)$ *terminates.*

Proof (hint).

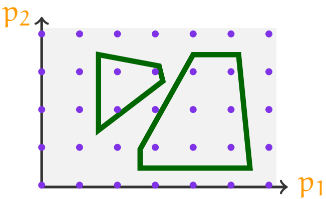
From the finiteness of the number of integer hulls of (M, X)-extrapolations of possible states. □

Characterization of RIEF

Theorem

Given a PTA $\mathcal{A}$ *with bounded parameters,* $\mathsf{RIEF}(\mathcal{A}, G)$ *contains*

1. *no valuation that is not a solution of* $\mathsf{EF}(\mathcal{A}, G)$ *[correctness]*

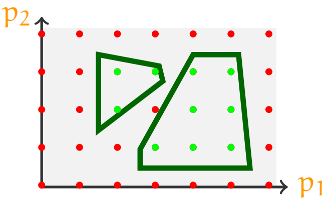


Characterization of RIEF

Theorem

Given a PTA $\mathcal{A}$ *with bounded parameters,* $\mathsf{RIEF}(\mathcal{A}, G)$ *contains*

1. *no valuation that is not a solution of* $\mathsf{EF}(\mathcal{A}, G)$ *[correctness]*
2. *all the integer parameter valuations solution of EF [integer-completeness]*

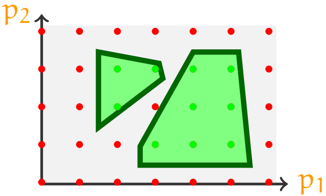


Characterization of RIEF

Theorem

Given a PTA $\mathcal{A}$ with bounded parameters, RIEF$(\mathcal{A}, G)$ *contains*

1. *no valuation that is not a solution of* EF$(\mathcal{A}, G)$ *[correctness]*
2. *all the integer parameter valuations solution of EF* *[integer-completeness]*
3. *all the rational valuations in the parametric zones computed by the symbolic exploration* *[?]*



Unavoidability

Algorithm RIAF computing parameter valuations such that a set of locations is unavoidable

Similar principle and similar results

(see paper)

Outline

Roméo

Model checker for parametric time Petri nets and PTA [Lime et al., 2009]

Uses the Parma Polyhedra Library (PPL) for operations on polyhedra [Bagnara et al., 2008]

Available in the open source CeCILL license

www.ROMEO.xxx

Case study: scheduling example

Three tasks τ_1, τ_2, τ_3 scheduled using static priorities ($\tau_1 > \tau_2 > \tau_3$) in a non-preemptive manner [Jovanović et al., 2015]

Task τ_1: periodic with period a and a non-deterministic duration in $[10, b]$

Task τ_2: minimal activation time of $2a$ and a non-deterministic duration in $[18, 28]$

Task τ_3: periodic with period $3a$ and a non-deterministic duration in $[20, 28]$.

Each task: deadline equal to its period

Case study: scheduling example

Three tasks τ_1, τ_2, τ_3 scheduled using static priorities ($\tau_1 > \tau_2 > \tau_3$) in a non-preemptive manner [Jovanović et al., 2015]

Task τ_1: periodic with period a and a non-deterministic duration in $[10, b]$

Task τ_2: minimal activation time of $2a$ and a non-deterministic duration in $[18, 28]$

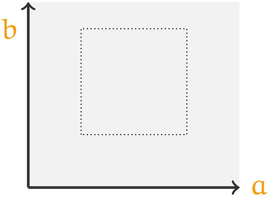
Task τ_3: periodic with period $3a$ and a non-deterministic duration in $[20, 28]$.

Each task: deadline equal to its period

Goal: synthesize parameter valuations ensuring that the system does not reach a deadline violation.

Experiments

Bounded parameters: $a \in [0, 50]$ and $b \in [0, 50]$

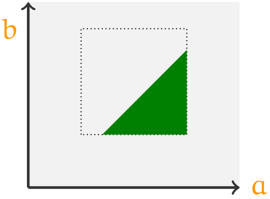


Algorithm	Result	Time
IEF	discrete	7.4 s
RIEF	dense	12.7 s

Experiments

Bounded parameters: $a \in [0, 50]$ and $b \in [0, 50]$

Result obtained by IEF: $a \geq 34, b \geq 10, a - b \geq 24$



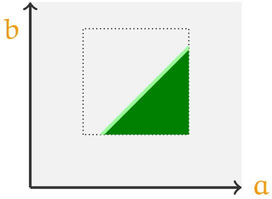
Algorithm	Result	Time
IEF	discrete	7.4 s
RIEF	dense	12.7 s

Experiments

Bounded parameters: $a \in [0, 50]$ and $b \in [0, 50]$

Result obtained by IEF: $a \geq 34, b \geq 10, a - b \geq 24$

Result obtained by RIEF: $a > \frac{562}{17}, b \geq 10, a - b > \frac{392}{17}$



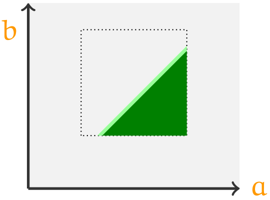
Algorithm	Result	Time
IEF	discrete	7.4 s
RIEF	dense	12.7 s

Experiments

Bounded parameters: $a \in [0, 50]$ and $b \in [0, 50]$

Result obtained by IEF: $a \geq 34, b \geq 10, a - b \geq 24$

Result obtained by RIEF: $a > \frac{562}{17}, b \geq 10, a - b > \frac{392}{17}$



Algorithm	Result	Time
IEF	discrete	7.4 s
RIEF	dense	12.7 s

- ☺ Slightly better result by RIEF
- ☹ Longer computation time (IH is expensive)
- ☺ Most important: RIEF is dense

Outline

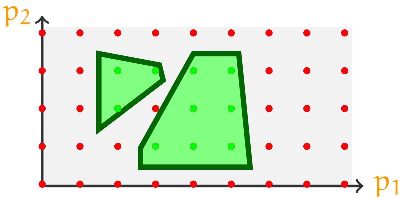
Summary

- Two synthesis algorithms for PTA with guaranteed termination and dense result
  - Dense valuations are important for robustness

- First terminating algorithms over dense valuations with guarantee on the results

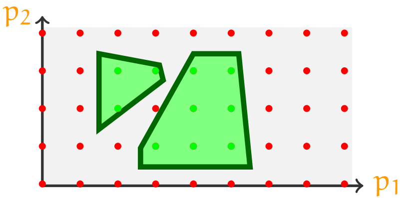
Perspectives

- Exact characterization of the result of RIEF and RIAF
  - What part of the result may be missing?



Perspectives

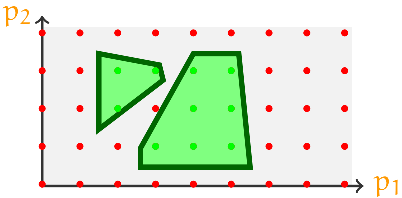
- Exact characterization of the result of RIEF and RIAF
  - What part of the result may be missing?



- Extension of this principle to further algorithms
  - Inverse method (trace or language preservation) [A., Chatain, Encrenaz, Fribourg, 2009] and implementation in IMITATOR

Perspectives

- Exact characterization of the result of RIEF and RIAF
  - What part of the result may be missing?



- Extension of this principle to further algorithms
  - Inverse method (trace or language preservation) [A., Chatain, Encrenaz, Fribourg, 2009] and implementation in IMITATOR
- Use multi-core processors
  - E.g., some cores to compute successor states, and some to check the equality of integer hulls

Bibliography

References I

Alur, R. and Dill, D. L. (1994).
A theory of timed automata.
Theoretical Computer Science, 126(2):183–235.

Alur, R., Henzinger, T. A., and Vardi, M. Y. (1993).
Parametric real-time reasoning.
In *STOC*, pages 592–601. ACM.

André, É., Chatain, T., Encrenaz, E., and Fribourg, L. (2009).
An inverse method for parametric timed automata.
International Journal of Foundations of Computer Science, 20(5):819–836.

André, É. and Markey, N. (2015).
Language preservation problems in parametric timed automata.
In *FORMATS*, volume 9268 of *Lecture Notes in Computer Science*, pages 27–43. Springer.

Bagnara, R., Hill, P. M., and Zaffanella, E. (2008).
The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems.
Science of Computer Programming, 72(1–2):3–21.

References II

Beneš, N., Bezděk, P., Larsen, K. G., and Srba, J. (2015).
Language emptiness of continuous-time parametric timed automata.
In *ICALP, Part II*, volume 9135 of *Lecture Notes in Computer Science*, pages 69–81. Springer.

Doyen, L. (2007).
Robust parametric reachability for timed automata.
Information Processing Letters, 102(5):208–213.

Jovanović, A., Lime, D., and Roux, O. H. (2015).
Integer parameter synthesis for real-time systems.
IEEE Transactions on Software Engineering, 41(5):445–461.

Lime, D., Roux, O. H., Seidner, C., and Traonouez, L.-M. (2009).
Romeo: A parametric model-checker for Petri nets with stopwatches.
In *TACAS*, volume 5505 of *Lecture Notes in Computer Science*, pages 54–57. Springer.

Markey, N. (2011).
Robustness in real-time systems.
In *SIES*, pages 28–34. IEEE Computer Society Press.

References III

Miller, J. S. (2000).
Decidability and complexity results for timed automata and semi-linear hybrid automata.
In *HSCC*, volume 1790 of *Lecture Notes in Computer Science*, pages 296–309. Springer.

Licensing

Source of the graphics used I

Title: Smiley green alien big eyes (aaah)
Author: LadyofHats
Source: https://commons.wikimedia.org/wiki/File:Smiley_green_alien_big_eyes.svg
License: public domain

Title: Smiley green alien big eyes (cry)
Author: LadyofHats
Source: https://commons.wikimedia.org/wiki/File:Smiley_green_alien_big_eyes.svg
License: public domain

License of this document