École Temps-Réel 2017

Lundi 28 août 2017
Paris, France

Timed automata and parametric timed automata

Étienne André

LIPN, Université Paris 13, CNRS, France

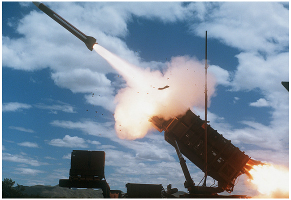
Feat. joint work with Giuseppe Lipari and Sun Youcheng

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

Context: Verifying complex timed systems

- Need for early bug detection
  - Bugs discovered when final testing: expensive

  $\leadsto$ Need for a thorough specification and verification phase



The Therac-25 radiation therapy machine (1/2)

- Radiation therapy machine used in the 1980s
- Involved in accidents between 1985 and 1987, in which patients were given massive overdoses of radiation
  - Approximately 100 times the intended dose!
  - Numerous causes, including race condition

The Therac-25 radiation therapy machine (1/2)

- Radiation therapy machine used in the 1980s
- Involved in accidents between 1985 and 1987, in which patients were given massive overdoses of radiation
  - Approximately 100 times the intended dose!
  - Numerous causes, including race condition

"The failure only occurred when a particular nonstandard sequence of keystrokes was entered on the VT-100 terminal which controlled the PDP-11 computer: an X *to (erroneously) select 25MV photon mode followed by* ↑, E *to (correctly) select 25 MeV Electron mode, then* Enter, *all within eight seconds."*

The Therac-25 radiation therapy machine (2/2)

The testing engineers could obviously not detect this strange (and quick!) sequence leading to the failure.

The Therac-25 radiation therapy machine (2/2)

The testing engineers could obviously not detect this strange (and quick!) sequence leading to the failure.

Limits of testing

This case illustrates the difficulty of bug detection without formal methods.

Bugs can be difficult to find

. . . and can have dramatic consequences for critical systems:

- health-related devices
- aeronautics and aerospace transportation
- smart homes and smart cities
- military devices
- etc.

Bugs can be difficult to find

. . . and can have dramatic consequences for critical systems:

- health-related devices
- aeronautics and aerospace transportation
- smart homes and smart cities
- military devices
- etc.

Hence, high need for formal verification

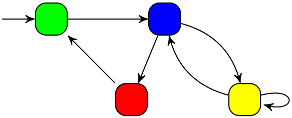
Outline

Outline: Finite-state automata

Outline: Finite-state automata

Model checking concurrent systems

- Use formal methods [Baier and Katoen, 2008]



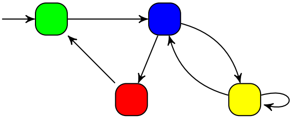
 is unreachable

A model of the system

A property to be satisfied

Model checking concurrent systems

- Use formal methods [Baier and Katoen, 2008]



$$\overset{?}{\models}$$

 is unreachable

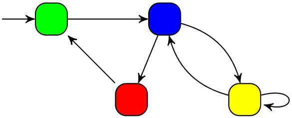
A model of the system

A property to be satisfied

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

Model checking concurrent systems

- Use formal methods [Baier and Katoen, 2008]



$\overset{?}{\models}$

 is unreachable

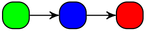
A model of the system

A property to be satisfied

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

Yes

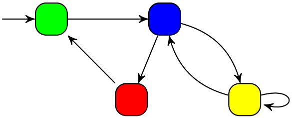
No



Counterexample

Model checking concurrent systems

- Use formal methods [Baier and Katoen, 2008]



$\overset{?}{\models}$

 is unreachable

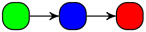
A model of the system

A property to be satisfied

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

Yes

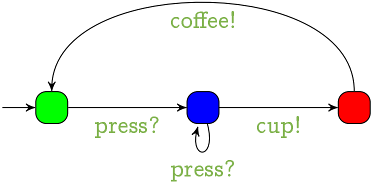
No

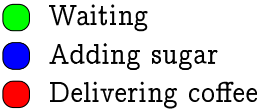


Counterexample

Turing award (2007) to Edmund M. Clarke, Allen Emerson and Joseph Sifakis

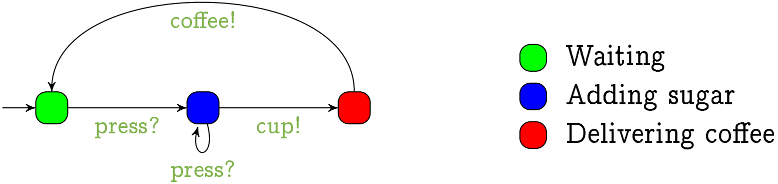
Finite state automata: A coffee machine $\mathcal{A}_C$





- Example of runs
  - Coffee with no sugar

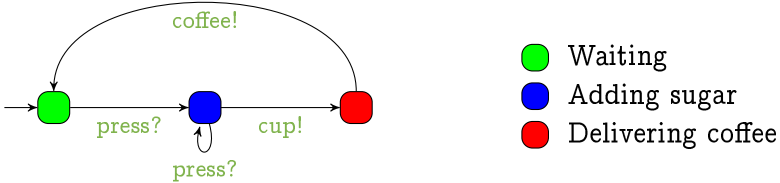
Finite state automata: A coffee machine $\mathcal{A}_C$



- Example of runs
  - Coffee with no sugar

  - Coffee with 2 doses of sugar

Finite state automata: A coffee machine $\mathcal{A}_C$



- Example of runs
  - Coffee with no sugar

  - Coffee with 2 doses of sugar

  - And so on

Outline: Timed automata

Beyond finite state automata

Finite State Automata give a powerful syntax and semantics to model qualitative aspects of systems

- Executions, sequence of actions
- Modular definitions (parallelism)
- Powerful checking (reachability, safety, liveness...)

Beyond finite state automata

Finite State Automata give a powerful syntax and semantics to model qualitative aspects of systems

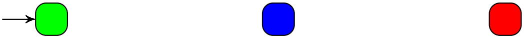
- Executions, sequence of actions
- Modular definitions (parallelism)
- Powerful checking (reachability, safety, liveness. . .)

But what about quantitative aspects:

- Time ("the airbag always eventually inflates, but maybe 10 seconds after the crash")
- Temperature ("the alarm always eventually ring, but maybe when the temperature is above 75 degrees")

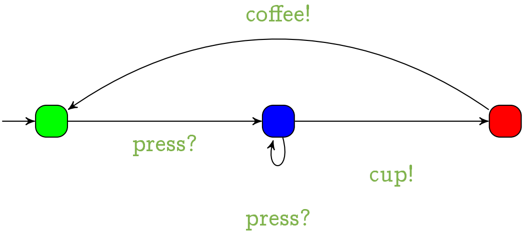
Timed automaton (TA)

- Finite state automaton (sets of locations)



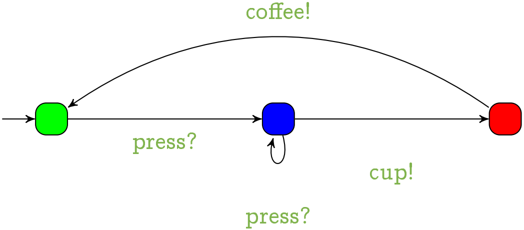
Timed automaton (TA)

- Finite state automaton (sets of locations and actions)



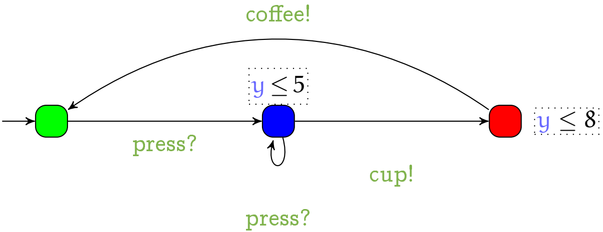
Timed automaton (TA)

- 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



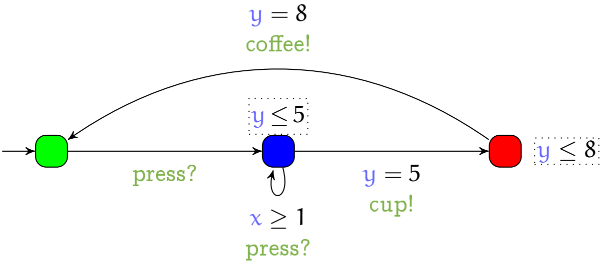
Timed automaton (TA)

- 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
  - Can be compared to integer constants in invariants
- Features
  - Location invariant: property to be verified to stay at a location



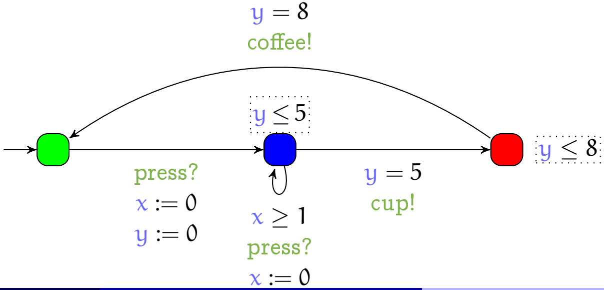
Timed automaton (TA)

- 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
  - 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



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
  - 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 at each transition



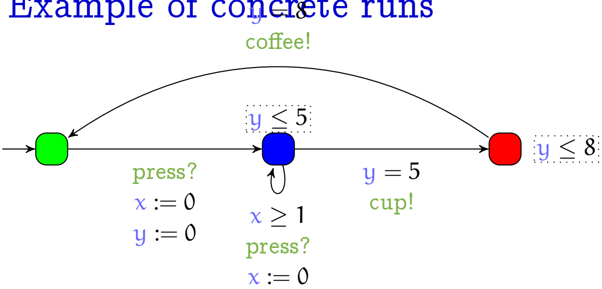
Concrete semantics of timed automata

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

  Example: $\left(\quad , \binom{x=1.2}{y=3.7} \right)$

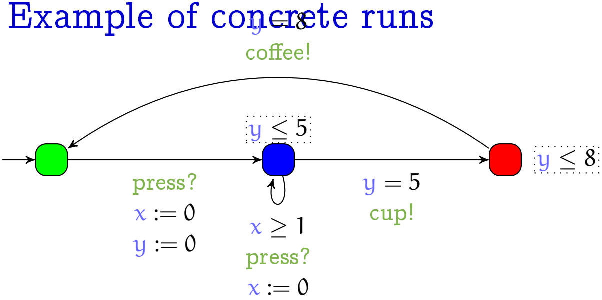
- Concrete run: alternating sequence of concrete states and actions or time elapse

Example of concrete runs



- Possible concrete runs for the coffee machine

Example of concrete runs

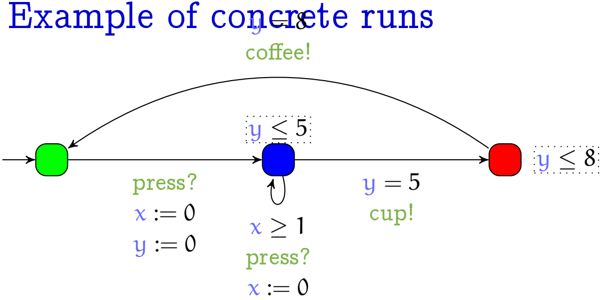


- Possible concrete runs for the coffee machine
  - Coffee with no sugar

$x =$ 0

$y =$ 0

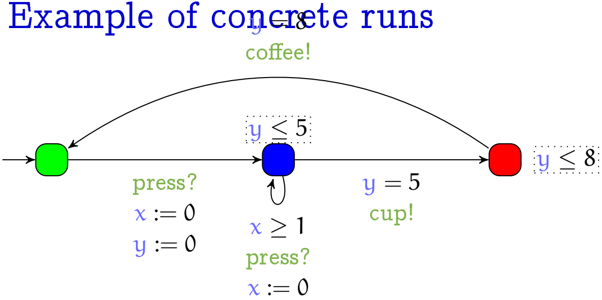
Example of concrete runs



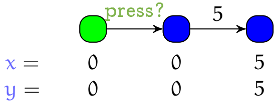
- Possible concrete runs for the coffee machine
  - Coffee with no sugar

$x =$	0	0
$y =$	0	0

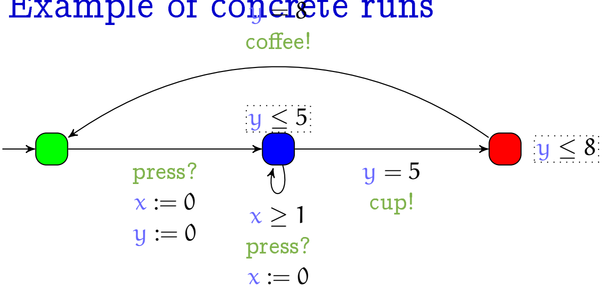
Example of concrete runs



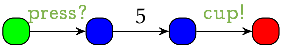
- Possible concrete runs for the coffee machine
  - Coffee with no sugar



Example of concrete runs

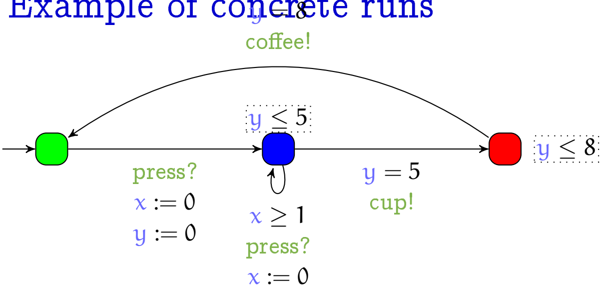


- Possible concrete runs for the coffee machine
  - Coffee with no sugar

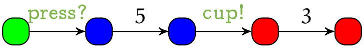


$x =$	0	0	5	5
$y =$	0	0	5	5

Example of concrete runs

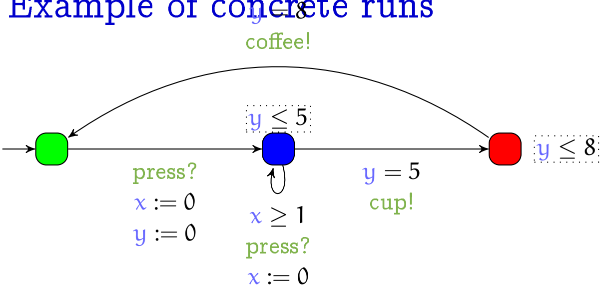


- Possible concrete runs for the coffee machine
  - Coffee with no sugar

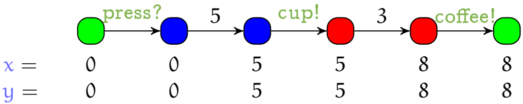


$x =$	0	0	5	5	8
$y =$	0	0	5	5	8

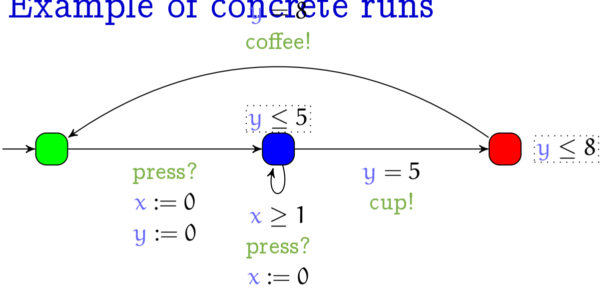
Example of concrete runs



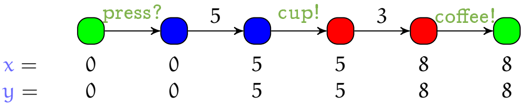
- Possible concrete runs for the coffee machine
  - Coffee with no sugar



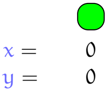
Example of concrete runs



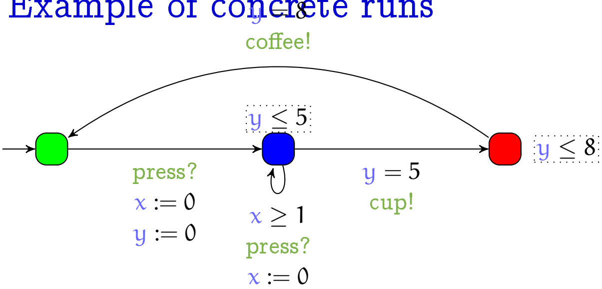
- Possible concrete runs for the coffee machine
  - Coffee with no sugar



  - Coffee with 2 doses of sugar



Example of concrete runs



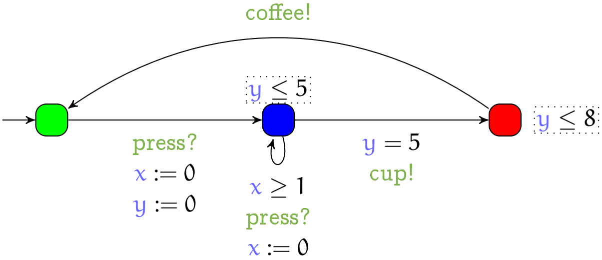
- Possible concrete runs for the coffee machine
  - Coffee with no sugar

$x =$	0	0	5	5	8	8
$y =$	0	0	5	5	8	8

  - Coffee with 2 doses of sugar

$x =$	0	0
$y =$	0	0

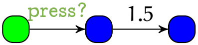
Example of concrete runs



- Possible concrete runs for the coffee machine
  - Coffee with no sugar

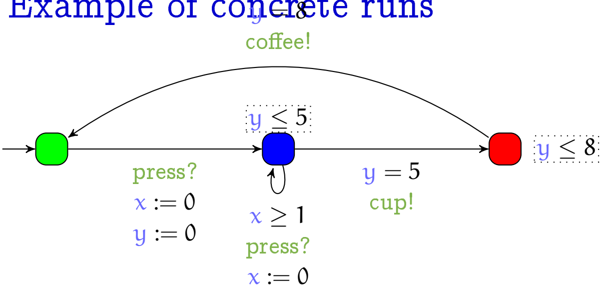
$x =$	0	0	5	5	8	8
$y =$	0	0	5	5	8	8

  - Coffee with 2 doses of sugar



$x =$	0	0	1.5
$y =$	0	0	1.5

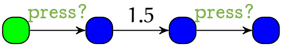
Example of concrete runs



- Possible concrete runs for the coffee machine
  - Coffee with no sugar

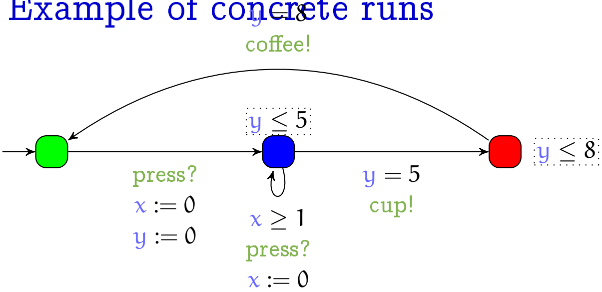
$x =$	0	0	5	5	8	8
$y =$	0	0	5	5	8	8

  - Coffee with 2 doses of sugar



$x =$	0	0	1.5	0
$y =$	0	0	1.5	1.5

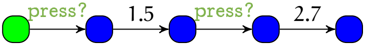
Example of concrete runs



- Possible concrete runs for the coffee machine
  - Coffee with no sugar

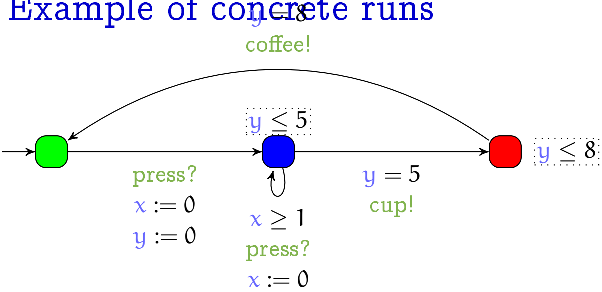
$x =$	0	0	5	5	8	8
$y =$	0	0	5	5	8	8

  - Coffee with 2 doses of sugar



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

Example of concrete runs



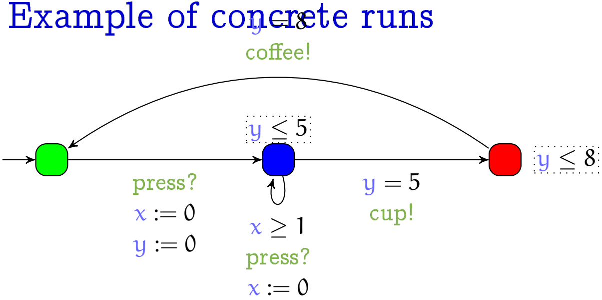
- Possible concrete runs for the coffee machine
  - Coffee with no sugar

$x =$	0	0	5	5	8	8
$y =$	0	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

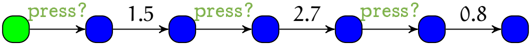
Example of concrete runs



- Possible concrete runs for the coffee machine
  - Coffee with no sugar

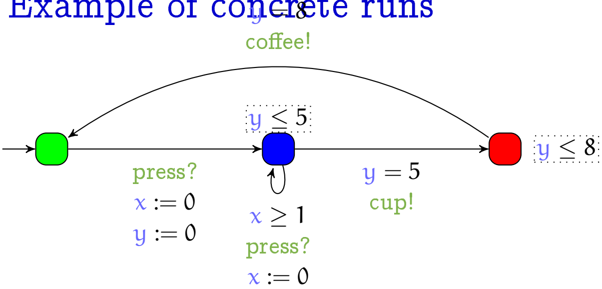
$x =$	0	0	5	5	8	8
$y =$	0	0	5	5	8	8

  - Coffee with 2 doses of sugar



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

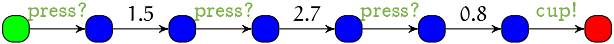
Example of concrete runs



- Possible concrete runs for the coffee machine
  - Coffee with no sugar

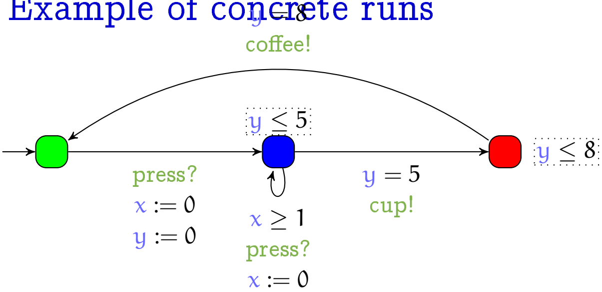
$x =$	0	0	5	5	8	8
$y =$	0	0	5	5	8	8

  - Coffee with 2 doses of sugar



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

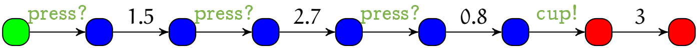
Example of concrete runs



- Possible concrete runs for the coffee machine
  - Coffee with no sugar

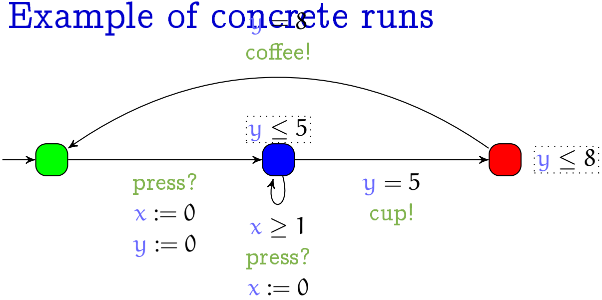
$x =$	0	0	5	5	8	8
$y =$	0	0	5	5	8	8

  - 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

Example of concrete runs



- Possible concrete runs for the coffee machine
  - Coffee with no sugar

$x =$	0	0	5	5	8	8
$y =$	0	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	3.8
$y =$	0	0	1.5	1.5	4.2	4.2	5	5	8	8

Dense time

- Time is dense: transitions can be taken anytime
  - Infinite number of timed runs
  - Model checking needs a finite structure!
- Some runs are equivalent
  - Taking the press? action at $t = 1.5$ or $t = 1.57$ is equivalent w.r.t. the possible actions
- Idea: reason with abstractions
  - Region automaton [Alur and Dill, 1994], and zone automaton
  - Example: in location , all clock values in the following zone are equivalent

$$y \leq 5 \wedge y - x \geq 4$$

  - This abstraction is finite

Symbolic states for timed automata

- Objective: group all concrete states reachable by the same sequence of discrete actions
- Symbolic state: a location l and a (infinite) set of states Z
- For timed automata, Z can be represented by a convex polyhedron with a special form called zone, with constraints

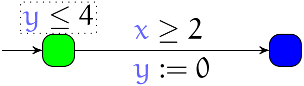
$$-d_{0i} \leq x_i \leq d_{i0} \text{ and } x_i - x_j \leq d_{ij}$$

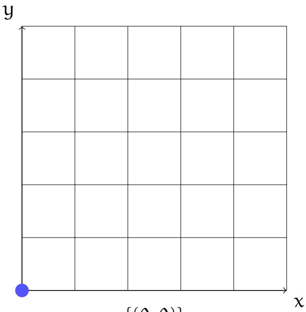
- Computation of successive reachable symbolic states can be performed symbolically with polyhedral operations: for edge $e = (l, a, g, R, l')$:

$$\mathsf{Succ}((l, Z), e) = (l', [(Z \cap g)]_R \cap I(l'))^{\nearrow} \cap I(l'))$$

- With an additional technicality, there is a finite number of reachable zones in a TA.

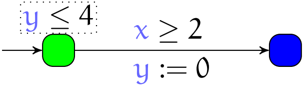
Symbolic states for timed automata: Example

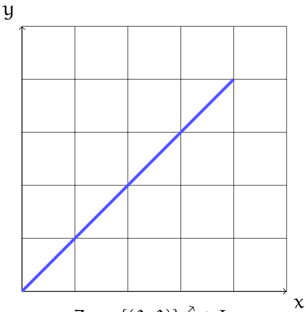




$\{(0, 0)\}$

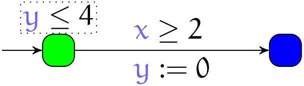
Symbolic states for timed automata: Example

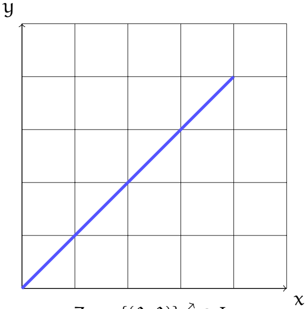




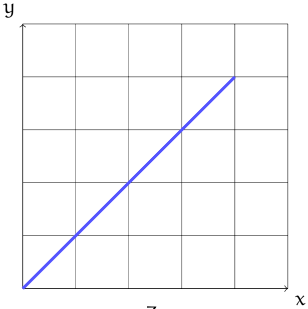
$Z_0 = \{(0,0)\}^{\nearrow} \cap I$●

Symbolic states for timed automata: Example



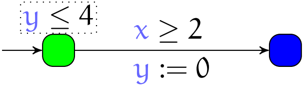


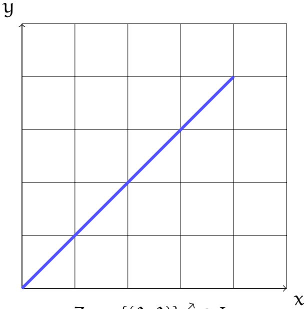
$Z_0 = \{(0,0)\}^{\nearrow} \cap I$



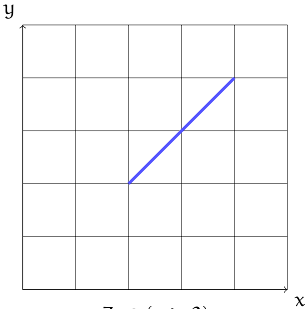
Z_0

Symbolic states for timed automata: Example



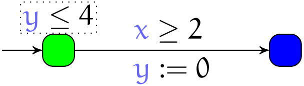


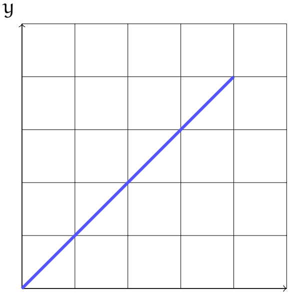
$Z_0 = \{(0,0)\}^{\nearrow} \cap I$



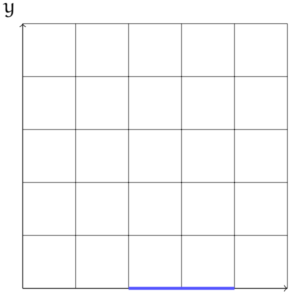
$Z_0 \cap (x \geq 2)$

Symbolic states for timed automata: Example



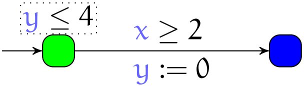


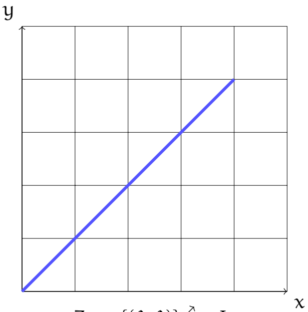
$Z_0 = \{(0,0)\}^{\nearrow} \cap I$



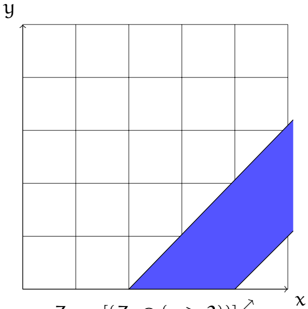
$[(Z_0 \cap (x \geq 2))]_{\{y\}}$

Symbolic states for timed automata: Example





$Z_0 = \{(0,0)\}^{\nearrow} \cap I$



$Z_1 = [(Z_0 \cap (x \geq 2))]^{\nearrow}_{\{y\}}$

Abstract semantics of timed automata

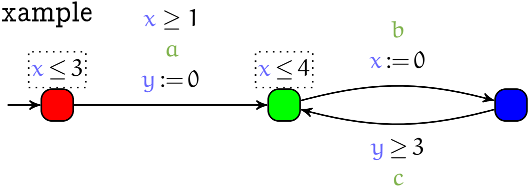
- Abstract state of a TA: pair (l, C), where
  - l is a location, and C is a constraint on the clocks ("zone")

Abstract semantics of timed automata

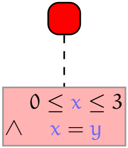
- Abstract state of a TA: pair (l, C), where
  - l is a location, and C is a constraint on the clocks ("zone")
- Abstract run: alternating sequence of abstract states and actions

Abstract semantics of timed automata

- Abstract state of a TA: pair (l, C), where
  - l is a location, and C is a constraint on the clocks ("zone")
- Abstract run: alternating sequence of abstract states and actions
- Example

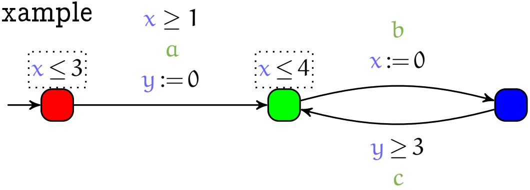


  - Possible abstract run for this TA

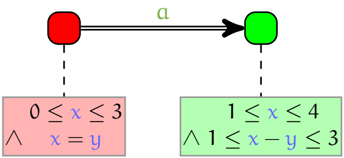


Abstract semantics of timed automata

- Abstract state of a TA: pair (l, C), where
  - l is a location, and C is a constraint on the clocks ("zone")
- Abstract run: alternating sequence of abstract states and actions
- Example

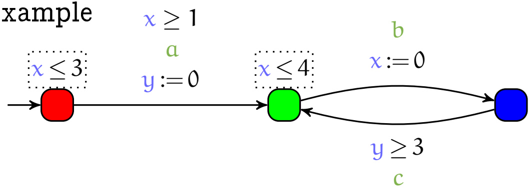


  - Possible abstract run for this TA

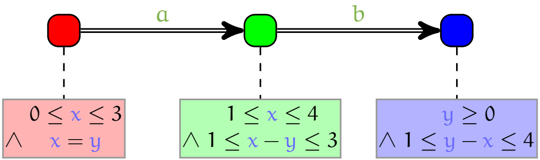


Abstract semantics of timed automata

- Abstract state of a TA: pair (l, C), where
  - l is a location, and C is a constraint on the clocks ("zone")
- Abstract run: alternating sequence of abstract states and actions
- Example

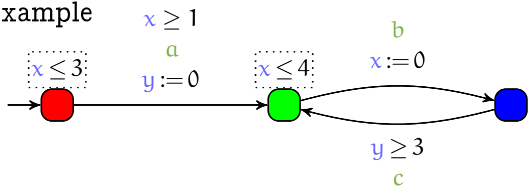


  - Possible abstract run for this TA

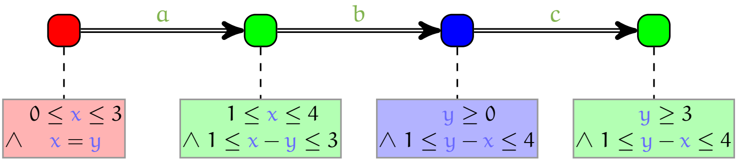


Abstract semantics of timed automata

- Abstract state of a TA: pair (l, C), where
  - l is a location, and C is a constraint on the clocks ("zone")
- Abstract run: alternating sequence of abstract states and actions
- Example



  - Possible abstract run for this TA



What is decidability?

Definition

A decision problem is decidable if one can design an algorithm that, for any input of the problem, can answer yes or no (in a finite time, with a finite memory).

What is decidability?

Definition

A decision problem is decidable if one can design an algorithm that, for any input of the problem, can answer yes or no (in a finite time, with a finite memory).

- "given three integers, is one of them the product of the other two?"
- "given a timed automaton, does there exist a run from the initial state to a given location l?"
- "given a context-free grammar, does it generate all strings?"
- "given a Turing machine, will it eventually halt?"

What is decidability?

Definition

A decision problem is decidable if one can design an algorithm that, for any input of the problem, can answer yes or no (in a finite time, with a finite memory).

- $\surd$ "given three integers, is one of them the product of the other two?"
- "given a timed automaton, does there exist a run from the initial state to a given location l?"
- "given a context-free grammar, does it generate all strings?"
- "given a Turing machine, will it eventually halt?"

What is decidability?

Definition

A decision problem is decidable if one can design an algorithm that, for any input of the problem, can answer yes or no (in a finite time, with a finite memory).

- √ "given three integers, is one of them the product of the other two?"
- √ "given a timed automaton, does there exist a run from the initial state to a given location l?"
- "given a context-free grammar, does it generate all strings?"
- "given a Turing machine, will it eventually halt?"

What is decidability?

Definition

A decision problem is decidable if one can design an algorithm that, for any input of the problem, can answer yes or no (in a finite time, with a finite memory).

- $\surd$ "given three integers, is one of them the product of the other two?"
- $\surd$ "given a timed automaton, does there exist a run from the initial state to a given location l?"
- $\times$ "given a context-free grammar, does it generate all strings?"
- "given a Turing machine, will it eventually halt?"

What is decidability?

Definition

A decision problem is decidable if one can design an algorithm that, for any input of the problem, can answer yes or no (in a finite time, with a finite memory).

- $\surd$ "given three integers, is one of them the product of the other two?"
- $\surd$ "given a timed automaton, does there exist a run from the initial state to a given location l?"
- $\times$ "given a context-free grammar, does it generate all strings?"
- $\times$ "given a Turing machine, will it eventually halt?"

Why studying decidability?

If a decision problem is undecidable, it is hopeless to look for algorithms yielding exact solutions (because that is impossible)

Why studying decidability?

If a decision problem is undecidable, it is hopeless to look for algorithms yielding exact solutions (because that is impossible)

However, one can:

- design semi-algorithms: if the algorithm halts, then its result is correct
- design algorithms yielding over- or under-approximations

Decision problems for timed automata

The finiteness of the region automaton allows us to check properties

- ☺ Reachability of a location [Alur and Dill, 1994]
- ☺ Liveness (Büchi conditions) [Alur and Dill, 1994]

Some problems impossible to check using the region automaton (but still decidable)

- ☺ non-Zenoness emptiness check [Wang et al., 2015]

Some undecidable problems (and hence impossible to check in general)

- ☹ universality of the timed language [Alur and Dill, 1994]
- ☹ timed language inclusion [Alur and Dill, 1994]

Software supporting timed automata

Timed automata have been successfully used since the 1990s

Tools for modeling and verifying models specified using TA

- HyTech (also hybrid, parametric timed automata) [Henzinger et al., 1997]
- Kronos [Yovine, 1997]
- TReX (also parametric timed automata) [Annichini et al., 2001]
- Uppaal [Larsen et al., 1997]
- Roméo (parametric time Petri nets) [Lime et al., 2009]
- PAT (also other formalisms) [Sun et al., 2009]
- IMITATOR (also parametric timed automata) [André et al., 2012]

Outline: Parametric timed automata

Beyond timed model checking: parameter synthesis

- 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?
  - System incompletely specified: Can I verify my system even if I don't know the period value with full certainty?

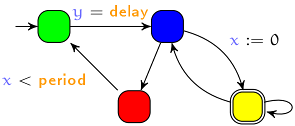
Beyond timed model checking: parameter synthesis

- 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?
  - System incompletely specified: Can I verify my system even if I don't know the period value with full certainty?

- Parameter synthesis
  - Consider that timing constants are unknown constants (parameters)

timed model checking



$\models^{?}$

 is unreachable

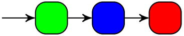
A model of the system

A property to be satisfied

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

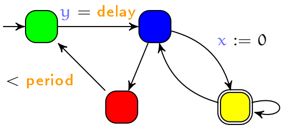
Yes

No



Counterexample

Parametric timed model checking



$\overset{?}{\models}$

 is unreachable

A model of the system

A property to be satisfied

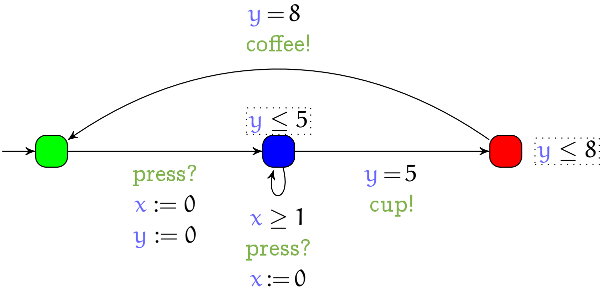
- Question: for what values of the parameters does the model of the system satisfy the property?

Yes if...

$$2\mathsf{delay} > \mathsf{period} \wedge \mathsf{period} < 20.46$$

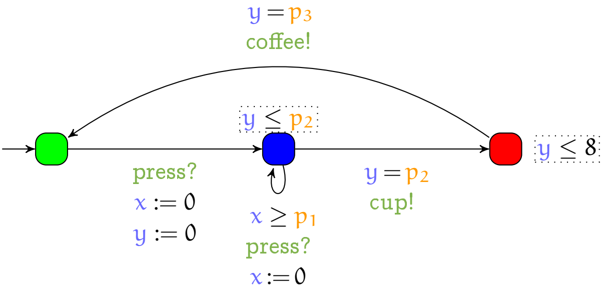
Parametric Timed Automaton (PTA)

- 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., 1993b]
  - Unknown constants compared to a clock in guards and invariants

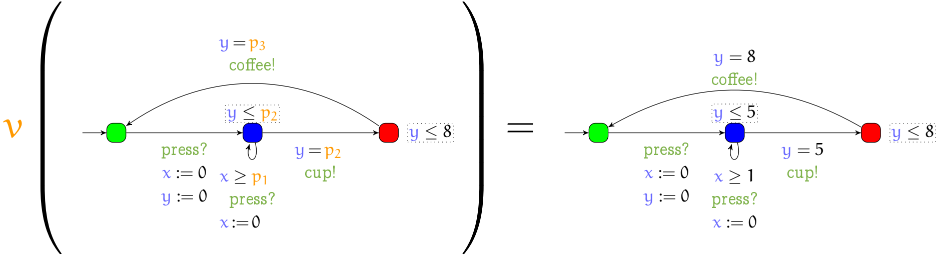


Valuation of a PTA

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

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 each parameter p is valuated by $v(p)$



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

Symbolic semantics of parametric timed automata

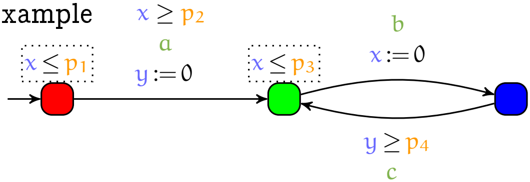
- Symbolic state of a PTA: pair (l, C), where
  - l is a location,
  - C is a convex polyhedron over X and P with a special form, called parametric zone [Hune et al., 2002]

Symbolic semantics of parametric timed automata

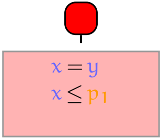
- Symbolic state of a PTA: pair (l, C), where
  - l is a location,
  - C is a convex polyhedron over X and P with a special form, called parametric zone [Hune et al., 2002]
- Symbolic run: alternating sequence of symbolic states and actions

Symbolic semantics of parametric timed automata

- Symbolic state of a PTA: pair (l, C), where
  - l is a location,
  - C is a convex polyhedron over X and P with a special form, called parametric zone [Hune et al., 2002]
- Symbolic run: alternating sequence of symbolic states and actions
- Example

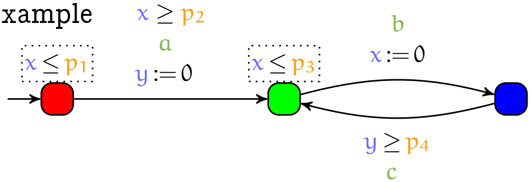


  - Possible symbolic run for this PTA

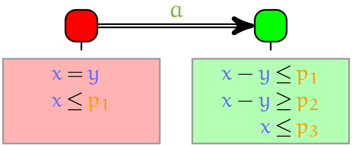


Symbolic semantics of parametric timed automata

- Symbolic state of a PTA: pair (l, C), where
  - l is a location,
  - C is a convex polyhedron over X and P with a special form, called parametric zone [Hune et al., 2002]
- Symbolic run: alternating sequence of symbolic states and actions
- Example

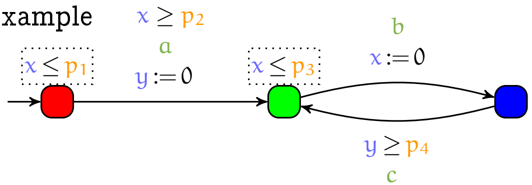


  - Possible symbolic run for this PTA

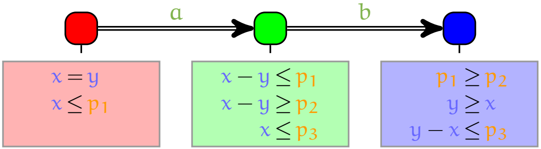


Symbolic semantics of parametric timed automata

- Symbolic state of a PTA: pair (l, C), where
  - l is a location,
  - C is a convex polyhedron over X and P with a special form, called parametric zone [Hune et al., 2002]
- Symbolic run: alternating sequence of symbolic states and actions
- Example

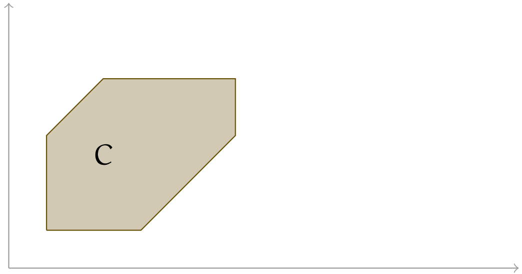


  - Possible symbolic run for this PTA



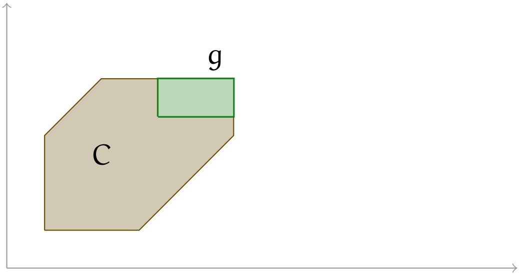
Symbolic semantics of PTA: Illustration

$$C' = [(C \cap g)]_R \cap I(l'))^{\nearrow} \cap I(l'))$$



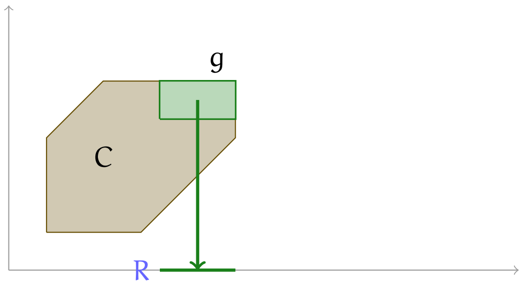
Symbolic semantics of PTA: Illustration

$$C' = [(C \cap g)]_R \cap I(l'))^{\nearrow} \cap I(l'))$$



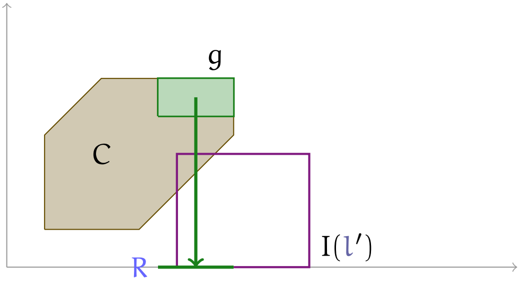
Symbolic semantics of PTA: Illustration

$$C' = [(C \cap g)]_R \cap I(l'))^{\nearrow} \cap I(l'))$$



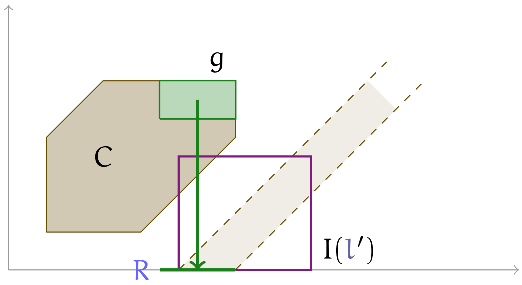
Symbolic semantics of PTA: Illustration

$$C' = [(C \cap g)]_R \cap I(l'))^{\nearrow} \cap I(l'))$$



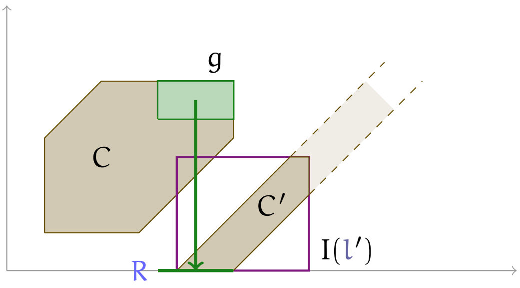
Symbolic semantics of PTA: Illustration

$$C' = [(C \cap g)]_R \cap I(l'))^{\nearrow} \cap I(l'))$$

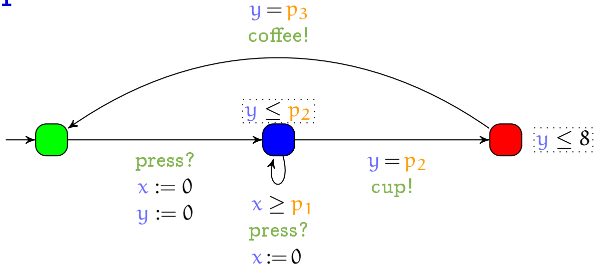


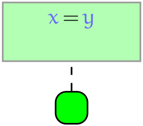
Symbolic semantics of PTA: Illustration

$$C' = [(C \cap g)]_R \cap I(l'))^{\nearrow} \cap I(l'))$$

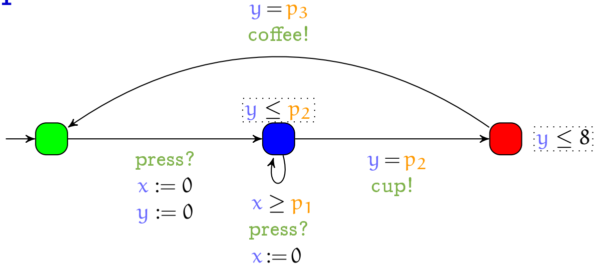


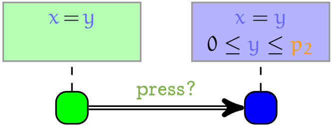
Symbolic exploration: Coffee machine



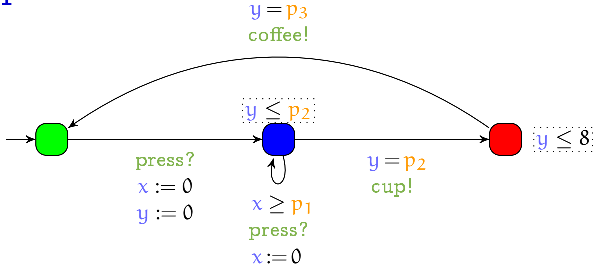


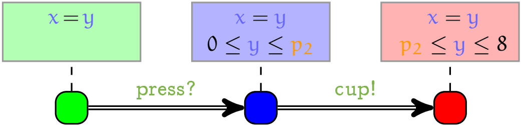
Symbolic exploration: Coffee machine



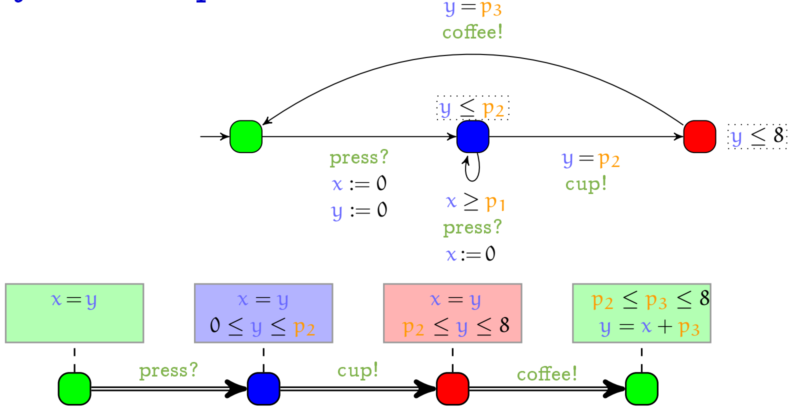


Symbolic exploration: Coffee machine

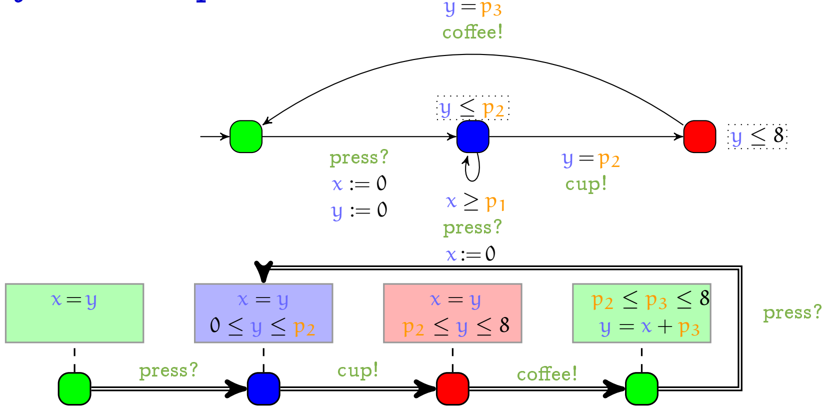




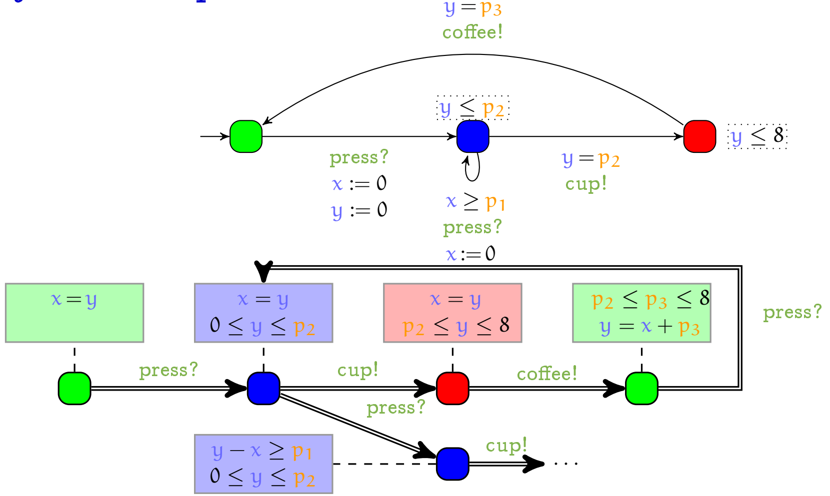
Symbolic exploration: Coffee machine



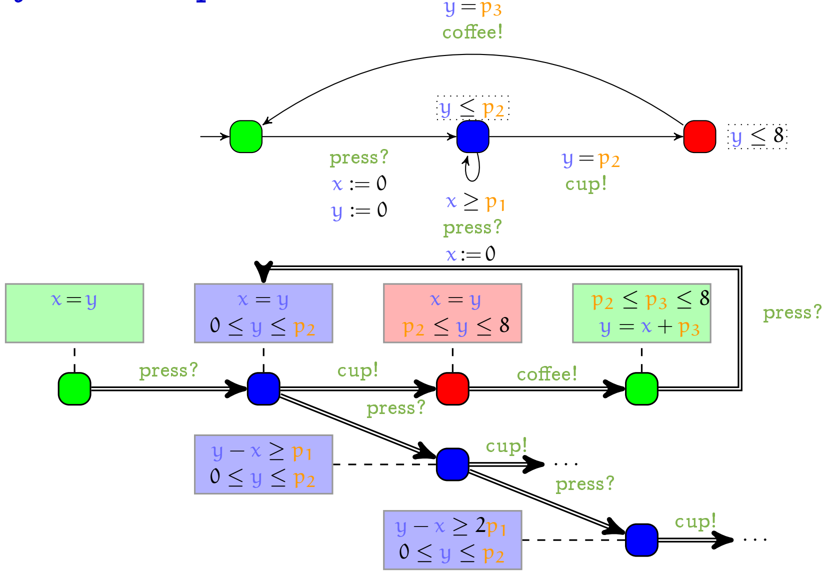
Symbolic exploration: Coffee machine



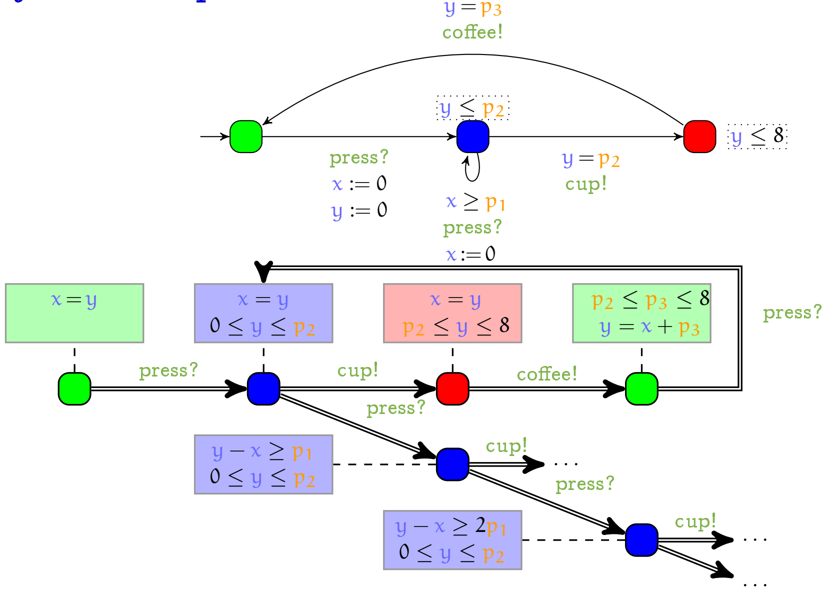
Symbolic exploration: Coffee machine



Symbolic exploration: Coffee machine



Symbolic exploration: Coffee machine



Decision and computation problems for PTA

- EF-Emptiness "Is the set of parameter valuations for which a given location l is reachable empty?"

  Example: "Does there exist at least one parameter valuation for which I can get a coffee with 2 sugars?"

- EF-Universality "Do all parameter valuations allow to reach a given location l?"

  Example: "Are all parameter valuations such that I may eventually get a coffee?"

- Preservation of the untimed language "Given a parameter valuation, does there exist another valuation with the same untimed language?"

  Example: "Given the valuation $p_1 = 1, p_2 = 5, p_3 = 8$, do there exist other valuations with the same possible untimed behaviours?"

Decision and computation problems for PTA

- EF-Emptiness "Is the set of parameter valuations for which a given location l is reachable empty?"

  Example: "Does there exist at least one parameter valuation for which I can get a coffee with 2 sugars?" $\surd$, e. g., $p_1 = 1, p_2 = 5, p_3 = 8$

- EF-Universality "Do all parameter valuations allow to reach a given location l?"

  Example: "Are all parameter valuations such that I may eventually get a coffee?"

- Preservation of the untimed language "Given a parameter valuation, does there exist another valuation with the same untimed language?"

  Example: "Given the valuation $p_1 = 1, p_2 = 5, p_3 = 8$, do there exist other valuations with the same possible untimed behaviours?"

Decision and computation problems for PTA

- EF-Emptiness "Is the set of parameter valuations for which a given location l is reachable empty?"

  Example: "Does there exist at least one parameter valuation for which I can get a coffee with 2 sugars?" $\surd$, e. g., $p_1 = 1, p_2 = 5, p_3 = 8$

- EF-Universality "Do all parameter valuations allow to reach a given location l?"

  Example: "Are all parameter valuations such that I may eventually get a coffee?" $\times$, e. g., $p_1 = 1, p_2 = 5, p_3 = 2$

- Preservation of the untimed language "Given a parameter valuation, does there exist another valuation with the same untimed language?"

  Example: "Given the valuation $p_1 = 1, p_2 = 5, p_3 = 8$, do there exist other valuations with the same possible untimed behaviours?"

Decision and computation problems for PTA

- EF-Emptiness "Is the set of parameter valuations for which a given location l is reachable empty?"

  Example: "Does there exist at least one parameter valuation for which I can get a coffee with 2 sugars?" $\surd$, e. g., $p_1 = 1, p_2 = 5, p_3 = 8$

- EF-Universality "Do all parameter valuations allow to reach a given location l?"

  Example: "Are all parameter valuations such that I may eventually get a coffee?" $\times$, e. g., $p_1 = 1, p_2 = 5, p_3 = 2$

- Preservation of the untimed language "Given a parameter valuation, does there exist another valuation with the same untimed language?"

  Example: "Given the valuation $p_1 = 1, p_2 = 5, p_3 = 8$, do there exist other valuations with the same possible untimed behaviours?" $\surd$

Computation problems for PTA

- EF-Synthesis "Find all parameter valuations for which a given location l is reachable"

  Example: "What are all parameter valuations such that one may eventually get a coffee?"

- and so on

Computation problems for PTA

- EF-Synthesis "Find all parameter valuations for which a given location l is reachable"

  Example: "What are all parameter valuations such that one may eventually get a coffee?" $0 \leq p_2 \leq p_3 \leq 8$

- and so on

Undecidability

- The symbolic state space is infinite in general
- No finite abstraction exists like for timed automata

Undecidability

- The symbolic state space is infinite in general
- No finite abstraction exists like for timed automata

Bad news

All interesting problems are undecidable for (general) parametric timed automata.

Decidability results for PTA (1/2)

- EF-emptiness problem
  "Is the set of parameter valuations for which a given location l is reachable empty?"
  undecidable [Alur et al., 1993b]

Decidability results for PTA (1/2)

- EF-emptiness problem
  "Is the set of parameter valuations for which a given location l is reachable empty?"
  undecidable [Alur et al., 1993b]

- EF-universality problem
  "Do all parameter valuations allow to reach a given location l?"
  undecidable [André et al., 2016]

Decidability results for PTA (1/2)

- EF-emptiness problem
  "Is the set of parameter valuations for which a given location l is reachable empty?"
  undecidable [Alur et al., 1993b]

- EF-universality problem
  "Do all parameter valuations allow to reach a given location l?"
  undecidable [André et al., 2016]

- AF-emptiness problem
  "Is the set of parameter valuations for which all runs eventually reach a given location l empty?"
  undecidable [Jovanović et al., 2015]

Decidability results for PTA (2/2)

- AF-universality problem
  "Do all parameter valuations allow to reach a given location l for all runs?"
  undecidable [André et al., 2016]

Decidability results for PTA (2/2)

- AF-universality problem
  "Do all parameter valuations allow to reach a given location l for all runs?"
  undecidable [André et al., 2016]

- Preservation of the untimed language
  "Given a parameter valuation, does there exist another valuations with the same untimed language?"
  undecidable [André and Markey, 2015]

Decidability results for PTA (2/2)

- AF-universality problem
  "Do all parameter valuations allow to reach a given location l for all runs?"
  undecidable [André et al., 2016]

- Preservation of the untimed language
  "Given a parameter valuation, does there exist another valuations with the same untimed language?"
  undecidable [André and Markey, 2015]

In fact most interesting problems for PTAs are undecidable

[André, 2017]

Limiting the number of clocks

Undecidability of EF-emptiness is achieved for a single parameter

[Miller, 2000, Beneš et al., 2015]

However, reducing the number of clocks yields decidability of the EF-emptiness problem:

Limiting the number of clocks

Undecidability of EF-emptiness is achieved for a single parameter
[Miller, 2000, Beneš et al., 2015]

However, reducing the number of clocks yields decidability of the EF-emptiness problem:

- √ 1 parametric clock and arbitrarily many non-parametric clocks and integer-valued parameters [Beneš et al., 2015]

Limiting the number of clocks

Undecidability of EF-emptiness is achieved for a single parameter

[Miller, 2000, Beneš et al., 2015]

However, reducing the number of clocks yields decidability of the EF-emptiness problem:

- √ 1 parametric clock and arbitrarily many non-parametric clocks and integer-valued parameters [Beneš et al., 2015]
- √ 1 parametric clock and arbitrarily many rational-valued parameters [Miller, 2000]

Limiting the number of clocks

Undecidability of EF-emptiness is achieved for a single parameter

[Miller, 2000, Beneš et al., 2015]

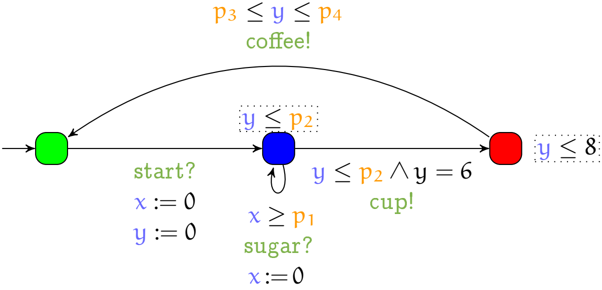
However, reducing the number of clocks yields decidability of the EF-emptiness problem:

- √ 1 parametric clock and arbitrarily many non-parametric clocks and integer-valued parameters [Beneš et al., 2015]
- √ 1 parametric clock and arbitrarily many rational-valued parameters [Miller, 2000]
- √ 2 parametric clocks and 1 integer-valued parameter [Bundala and Ouaknine, 2014]

L/U-PTAs

Definition

A lower/upper bound PTA (L/U-PTA) is a PTA in which each parameter p is always compared with clocks as an upper bound or always as a lower bound.



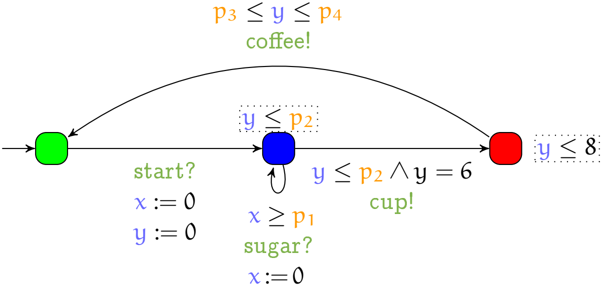
Lower-bound parameters:

Upped-bound parameters:

L/U-PTAs

Definition

A lower/upper bound PTA (L/U-PTA) is a PTA in which each parameter p is always compared with clocks as an upper bound or always as a lower bound.



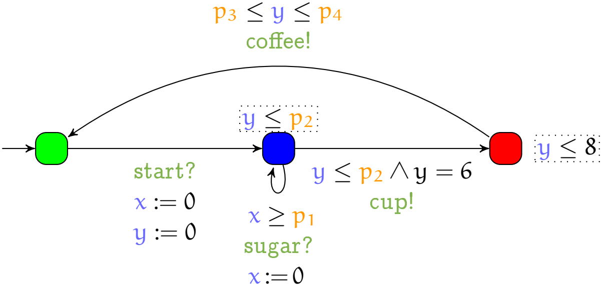
Lower-bound parameters: p_1, p_3

Upped-bound parameters:

L/U-PTAs

Definition

A lower/upper bound PTA (L/U-PTA) is a PTA in which each parameter p is always compared with clocks as an upper bound or always as a lower bound.



Lower-bound parameters: p_1, p_3

Upped-bound parameters: p_2, p_4

Decidable problems for L/U-PTA

- EF-emptiness problem
  "Does there exist a parameter valuation for which a given location l is reachable?"
  decidable [Hune et al., 2002]

Decidable problems for L/U-PTA

- EF-emptiness problem
  "Does there exist a parameter valuation for which a given location l is reachable?"
  decidable [Hune et al., 2002]

- EF-universality problem
  "Do all parameter valuations allow to reach a given location l?"
  decidable [Bozzelli and La Torre, 2009]

Decidable problems for L/U-PTA

- EF-emptiness problem
  "Does there exist a parameter valuation for which a given location l is reachable?"
  decidable [Hune et al., 2002]

- EF-universality problem
  "Do all parameter valuations allow to reach a given location l?"
  decidable [Bozzelli and La Torre, 2009]

- EF-finiteness problem
  "Is the set of parameter valuations allowing to reach a given location l finite?"
  decidable (for integer valuations) [Bozzelli and La Torre, 2009]

Undecidable problems for L/U-PTA

- AF-emptiness problem
  "Does there exist a parameter valuation for which a given location l is always eventually reachable?"
  undecidable [Jovanović et al., 2015]

Undecidable problems for L/U-PTA

- AF-emptiness problem
  "Does there exist a parameter valuation for which a given location l is always eventually reachable?"
  undecidable [Jovanović et al., 2015]

- AF-universality problem
  "Are all valuations such that a given location l is always eventually reachable?"
  undecidable (but...) [André and Lime, 2017]

Undecidable problems for L/U-PTA

- AF-emptiness problem
  "Does there exist a parameter valuation for which a given location l is always eventually reachable?"
  undecidable [Jovanović et al., 2015]

- AF-universality problem
  "Are all valuations such that a given location l is always eventually reachable?"
  undecidable (but. . .) [André and Lime, 2017]

- language preservation emptiness problem
  "Given a parameter valuation v, can we find another valuation with the same untimed language?"
  undecidable [André and Markey, 2015]

Outline: Modeling and verifying real-time systems with parameters

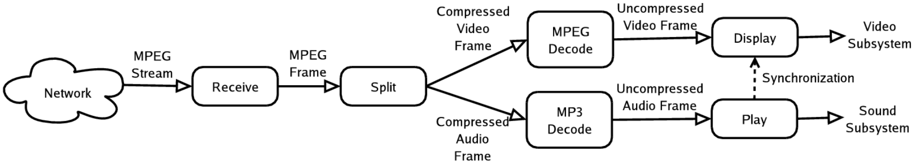
Context: Hard real-time embedded systems

- Modern hard real-time embedded systems are distributed in nature
- Many of them have critical timing requirements:
  - automotive systems (modern cars have 10-20 embedded boards connected by one or more CAN bus)
  - avionics systems (several distributed control boards connected by one or more dedicated networks)

- To analyze the schedulability of such systems, it is very important to estimate the (worst-case) computation times of the tasks
- Estimating WCET is very difficult in modern architectures

Real-time pipelines

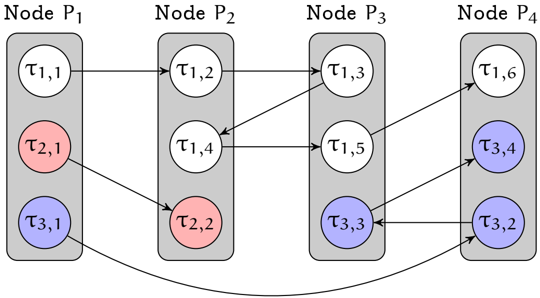
- Many real-time applications can be modeled as pipelines (also called transactions) of tasks



- Executed on a distributed (or multicore) system
- Activated cyclically (periodic or sporadic)
- Using preemption
  - Lower-priority tasks can be temporarily interrupted by a higher-priority task

Model

- A set of pipelines $\{\mathcal{P}^{(1)}, \dots, \mathcal{P}^{(p)}\}$ distributed over m nodes
- Each pipeline $\mathcal{P}^{(i)}$ is a chain of n_i tasks $\{\tau_{i,1}, \dots, \tau_{i,n_i}\}$
- Pipeline $\mathcal{P}^{(i)}$ has an end-to-end (E2E) deadline D_i and period T_i

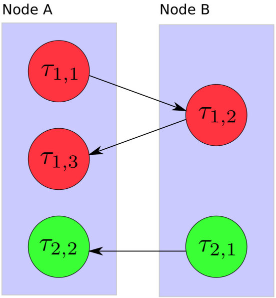


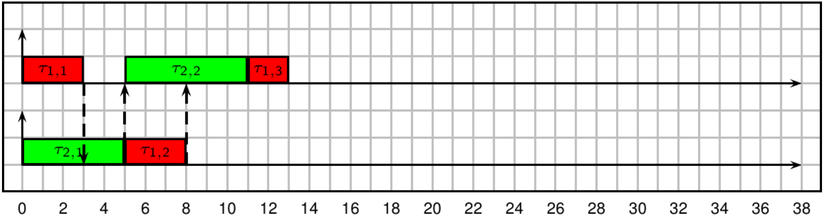
- Scheduling Problem: Guarantee that all pipelines complete before their E2E deadlines

Activations, jitter, deadline

- An example

Task	Per.	E2E	Comp.	Resp.	Jitter	prio
$\tau_{1,1}$	15		3	3	0	HI
$\tau_{1,2}$	-		3	8	0-2	LO
$\tau_{1,3}$	-	15	2	13	3	LO
$\tau_{2,1}$	12		5	6	0	HI
$\tau_{2,2}$	-	12	6	?	0-1	ME

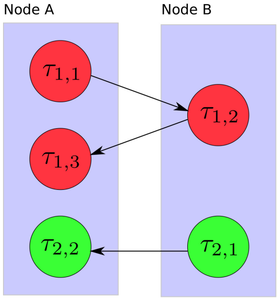


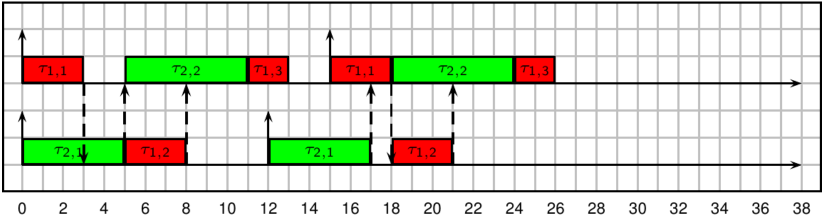


Activations, jitter, deadline

- An example

Task	Per.	E2E	Comp.	Resp.	Jitter	prio
$\tau_{1,1}$	15		3	3	0	HI
$\tau_{1,2}$	-		3	8	0-2	LO
$\tau_{1,3}$	-	15	2	13	3	LO
$\tau_{2,1}$	12		5	6	0	HI
$\tau_{2,2}$	-	12	6	?	0-1	ME

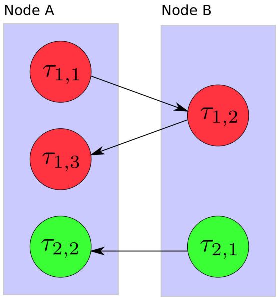


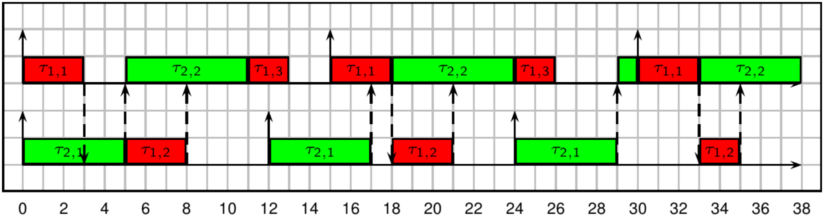


Activations, jitter, deadline

- An example

Task	Per.	E2E	Comp.	Resp.	Jitter	prio
$\tau_{1,1}$	15		3	3	0	HI
$\tau_{1,2}$	-		3	8	0-2	LO
$\tau_{1,3}$	-	15	2	13	3	LO
$\tau_{2,1}$	12		5	6	0	HI
$\tau_{2,2}$	-	12	6	**14**	0-1	ME



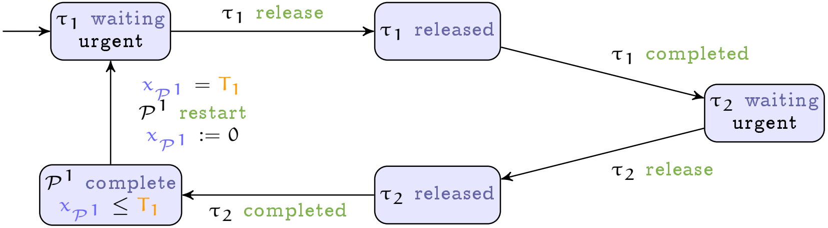


Enriching the model with parameters

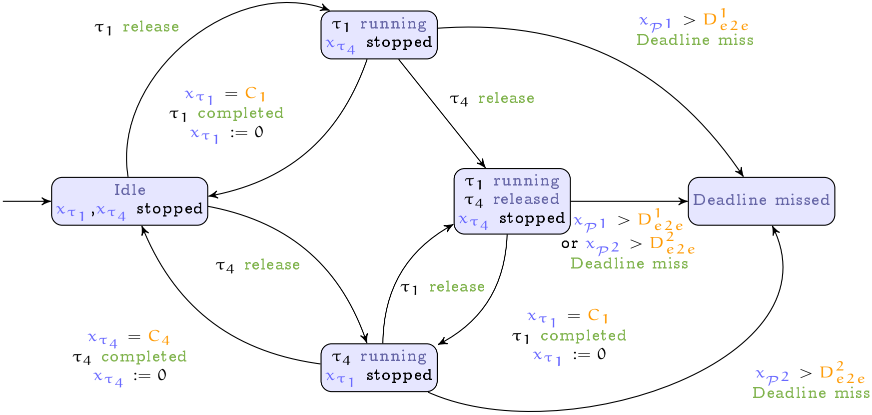
- A task is identified by three parameters:
  - C_i is the worst-case computation time (or worst-case transmission time, in case it models a message)
  - R_i is the task worst-case response time, i. e., the worst case finishing time of any task instance relative to the activation of its pipeline.
  - J_i is the task worst-case activation jitter, i. e., the greatest time since its activation that a task must wait for all preceding tasks to complete their execution

- A parameter of major interest is the computation time

Modeling a task / pipeline



Modeling the fixed priority scheduler (preemptive)



Actually a PTA extended with stopwatches [Sun et al., 2013]

an extension of stopwatch automata [Adbeddaïm and Maler, 2002]

Parametric verification of real-time systems

Many problems can be reduced to parametric reachability (EFsynth):
find parameter valuations for which a given state is (un)reachable

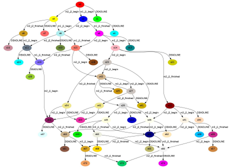
- ☹ This problem is undecidable for PTAs and many subclasses [A., STTT 2017]
- ☺ But we can still compute part (and often all) of the solution

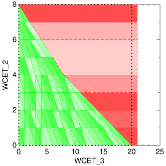
Interesting problems:

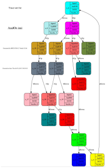
- Find parameter valuations for which no deadline violation occurs (i. e., for which the system is schedulable)
- Compute the worst-case computation time
- Find the parametric WCET when the jitter is unknown

IMITATOR

- A tool for modeling and verifying real-time systems with unknown constants modeled with parametric timed automata
  - Communication through (strong) broadcast synchronization
  - Rational-valued shared discrete variables
  - Stopwatches, to model schedulability problems with preemption
- Verification
  - Computation of the symbolic state space
  - (non-Zeno) parametric model checking (using a subset of TCTL)
  - Language and trace preservation, and robustness analysis
  - Parametric deadlock-freeness checking
  - Behavioral cartography







IMITATOR

Under continuous development since 2008 [André et al., 2012]

A library of benchmarks

- Communication protocols
- Schedulability problems
- Asynchronous circuits
- . . . and more

Free and open source software: Available under the GNU-GPL license

GPL Free Software

Free as in Freedom

IMITATOR

Under continuous development since 2008 [André et al., 2012]

A library of benchmarks

- Communication protocols
- Schedulability problems
- Asynchronous circuits
- . . . and more

Free and open source software: Available under the GNU-GPL license

Try it!

www.imitator.fr

Some success stories

- Modeled and verified an asynchronous memory circuit by ST-Microelectronics
  - Project ANR Valmem
- Parametric schedulability analysis of a prospective architecture for the flight control system of the next generation of spacecrafts designed at ASTRIUM Space Transportation [Fribourg et al., 2012]
- Formal timing analysis of music scores [Fanchon and Jacquemard, 2013]
- Solution to a challenge related to a distributed video processing system by Thales

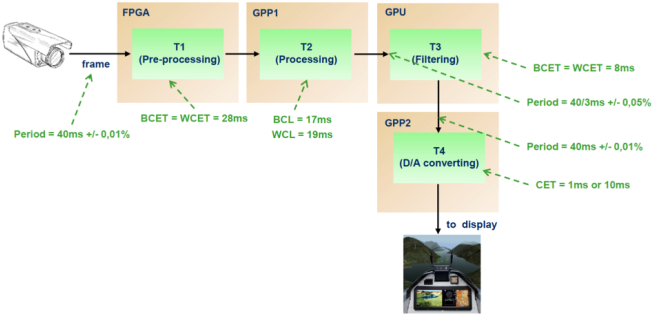
Outline

The FMTV 2015 Challenge (1/2)

Challenge by Thales proposed during the WATERS 2014 workshop
Solutions presented at WATERS 2015

System: an unmanned aerial video system with uncertain periods

- Period constant but with a small uncertainty (typically 0.01 %)
- Not a jitter!



The FMTV 2015 Challenge (2/2)

Goal

Compute the end-to-end BCET and WCET times for a buffer size of $n = 1$ and $n = 3$

The FMTV 2015 Challenge (2/2)

Goal

Compute the end-to-end BCET and WCET times for a buffer size of $n = 1$ and $n = 3$

- ☹ Not a typical parameter synthesis problem?
  - No parameters in the specification

The FMTV 2015 Challenge (2/2)

Goal

Compute the end-to-end BCET and WCET times for a buffer size of $n = 1$ and $n = 3$

☹ Not a typical parameter synthesis problem?

- No parameters in the specification

☺ A typical parameter synthesis problem

- The end-to-end time can be set as a parameter... to be synthesized
- The uncertain period is typically a parameter (with some constraint, e. g., $P1 \in [40 - 0.004, 40 + 0.004]$)

Methodology

1. Propose a PTA model with parameters for uncertain periods and the end-to-end time

Methodology

1. Propose a PTA model with parameters for uncertain periods and the end-to-end time
2. Add a specific location corresponding to the correct transmission of the frame

Methodology

1. Propose a PTA model with parameters for uncertain periods and the end-to-end time
2. Add a specific location corresponding to the correct transmission of the frame
3. Run the reachability synthesis algorithm EFsynth (implemented in IMITATOR) w.r.t. that location

Methodology

1. Propose a PTA model with parameters for uncertain periods and the end-to-end time
2. Add a specific location corresponding to the correct transmission of the frame
3. Run the reachability synthesis algorithm EFsynth (implemented in IMITATOR) w.r.t. that location
4. Gather all constraints (in as many dimensions as uncertain periods + the end-to-end time)

Methodology

1. Propose a PTA model with parameters for uncertain periods and the end-to-end time
2. Add a specific location corresponding to the correct transmission of the frame
3. Run the reachability synthesis algorithm EFsynth (implemented in IMITATOR) w.r.t. that location
4. Gather all constraints (in as many dimensions as uncertain periods + the end-to-end time)
5. Eliminate all parameters but the end-to-end time

Note: not eliminating parameters allows one to know for which values of the periods the best / worst case execution times are obtained.

Methodology

1. Propose a PTA model with parameters for uncertain periods and the end-to-end time
2. Add a specific location corresponding to the correct transmission of the frame
3. Run the reachability synthesis algorithm EFsynth (implemented in IMITATOR) w.r.t. that location
4. Gather all constraints (in as many dimensions as uncertain periods + the end-to-end time)
5. Eliminate all parameters but the end-to-end time
6. Exhibit the minimum and the maximum

Note: not eliminating parameters allows one to know for which values of the periods the best / worst case execution times are obtained.

To build the PTA model

- Uncertainties in the system:
  - $P1 \in [40 - 0.004, 40 + 0.004]$
  - $P3 \in [\frac{40}{3} - \frac{1}{150}, \frac{40}{3} + \frac{1}{150}]$
  - $P4 \in [40 - 0.004, 40 + 0.004]$

To build the PTA model

- Uncertainties in the system:
  - $P1 \in [40 - 0.004, 40 + 0.004]$
  - $P3 \in [\frac{40}{3} - \frac{1}{150}, \frac{40}{3} + \frac{1}{150}]$
  - $P4 \in [40 - 0.004, 40 + 0.004]$
- Parameters:
  - P1_uncertain
  - P3_uncertain
  - P4_uncertain

To build the PTA model

- Uncertainties in the system:
  - $P1 \in [40 - 0.004, 40 + 0.004]$
  - $P3 \in [\frac{40}{3} - \frac{1}{150}, \frac{40}{3} + \frac{1}{150}]$
  - $P4 \in [40 - 0.004, 40 + 0.004]$
- Parameters:
  - P1_uncertain
  - P3_uncertain
  - P4_uncertain
- The end-to-end latency (another parameter): E2E

To build the PTA model

- Uncertainties in the system:
  - $P1 \in [40 - 0.004, 40 + 0.004]$
  - $P3 \in [\frac{40}{3} - \frac{1}{150}, \frac{40}{3} + \frac{1}{150}]$
  - $P4 \in [40 - 0.004, 40 + 0.004]$
- Parameters:
  - P1_uncertain
  - P3_uncertain
  - P4_uncertain
- The end-to-end latency (another parameter): E2E
- Others:
  - the register between task 2 and task 3: discrete variable $reg_{2,3}$
  - the buffer between task 3 and task 4: $n = 1$ or $n = 3$

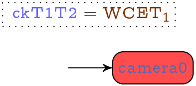
Simplification

- T1 and T2 are synchronised; T1, T3 and T4 are asynchronised
  - (exact modeling of the system behaviour is too heavy)

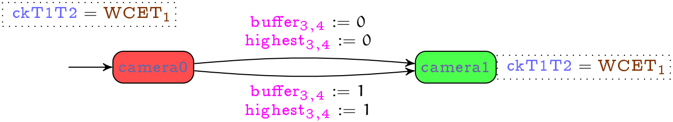
Simplification

- T1 and T2 are synchronised; T1, T3 and T4 are asynchronised
  - (exact modeling of the system behaviour is too heavy)
- We choose a single arbitrary frame, called the target one
- We assume the system is initially in an arbitrary status
  - This is our only uncertain assumption (in other words, can the periods deviate from each other so as to yield any arbitrary deviation?)

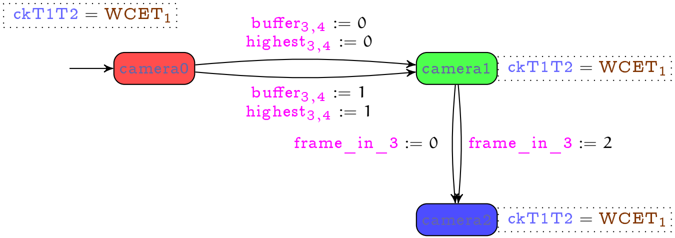
The initialization automaton



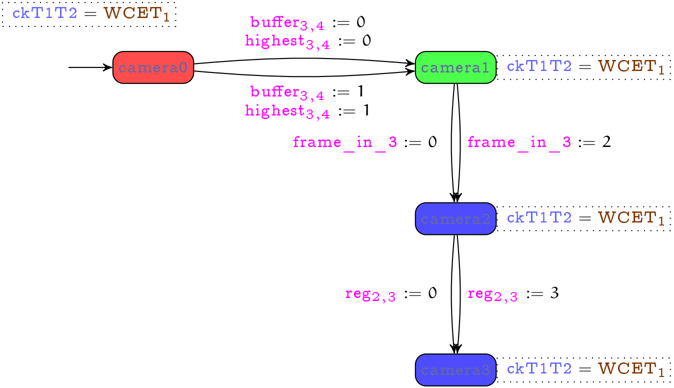
The initialization automaton



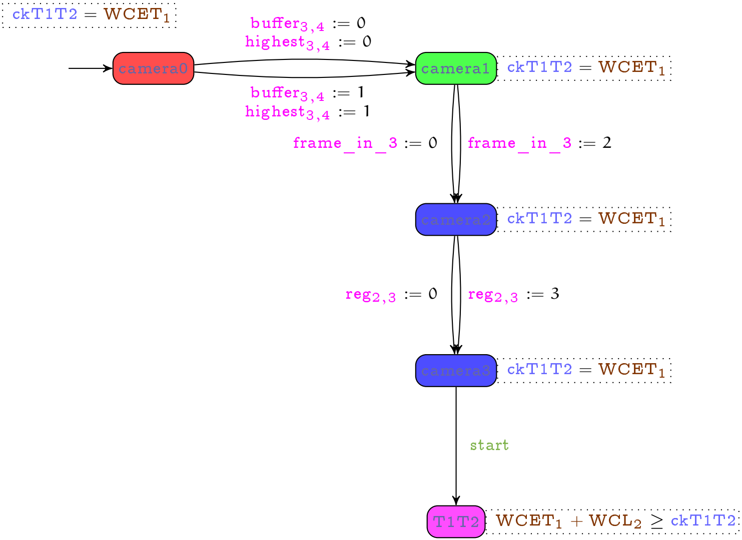
The initialization automaton



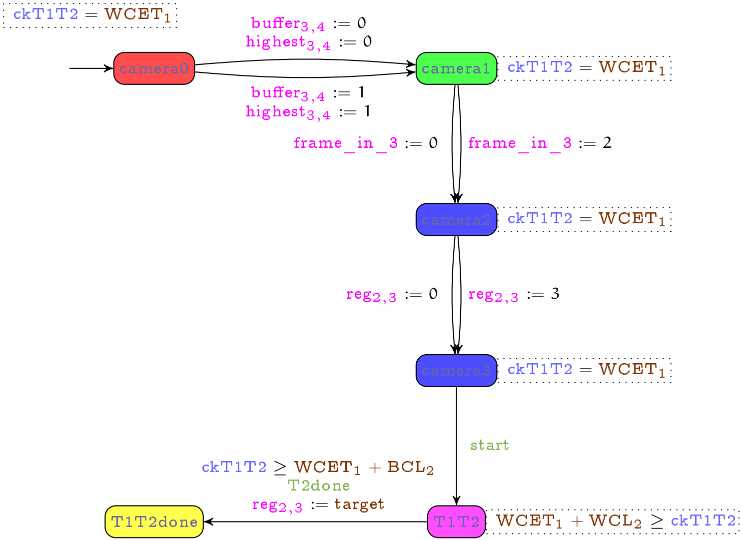
The initialization automaton



The initialization automaton

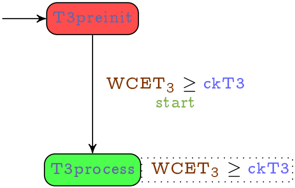


The initialization automaton

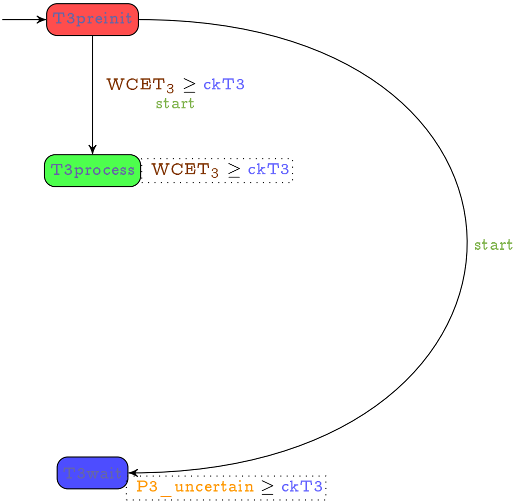


Task T3

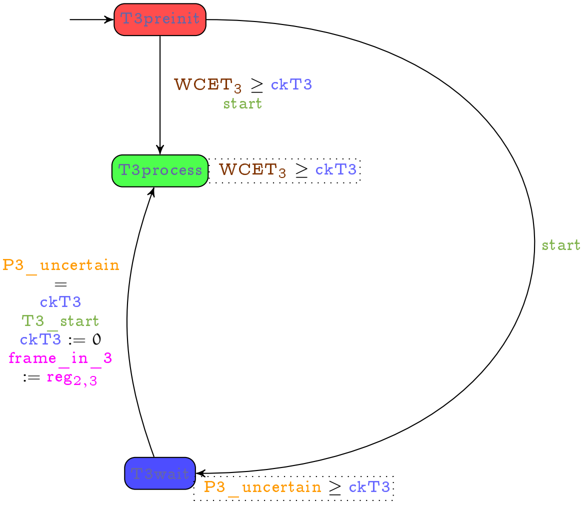
Task T3


T3preinit
$\mathrm{WCET}_3 \geq \mathrm{ckT3}$
start
T3process
$\mathrm{WCET}_3 \geq \mathrm{ckT3}$

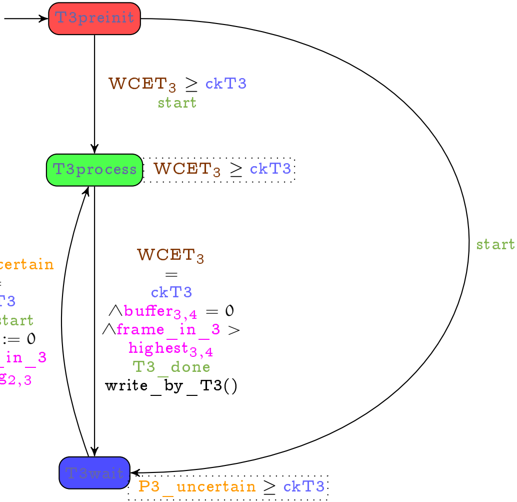
Task T3


T3preinit
$WCET_3 \geq ckT3$
start
T3process
$WCET_3 \geq ckT3$
start
T3wait
$P3\_uncertain \geq ckT3$

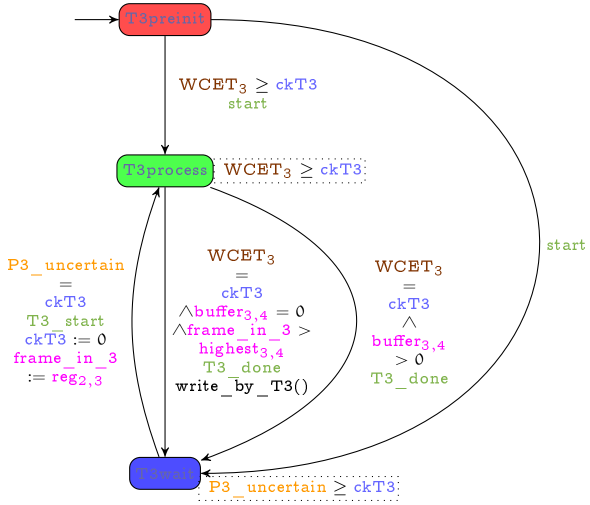
Task T3


T3preinit
$WCET_3 \geq ckT3$
start
T3process
$WCET_3 \geq ckT3$
start
P3_uncertain
=
ckT3
T3_start
ckT3 := 0
frame_in_3
:= $reg_{2,3}$
T3wait
P3_uncertain ≥ ckT3

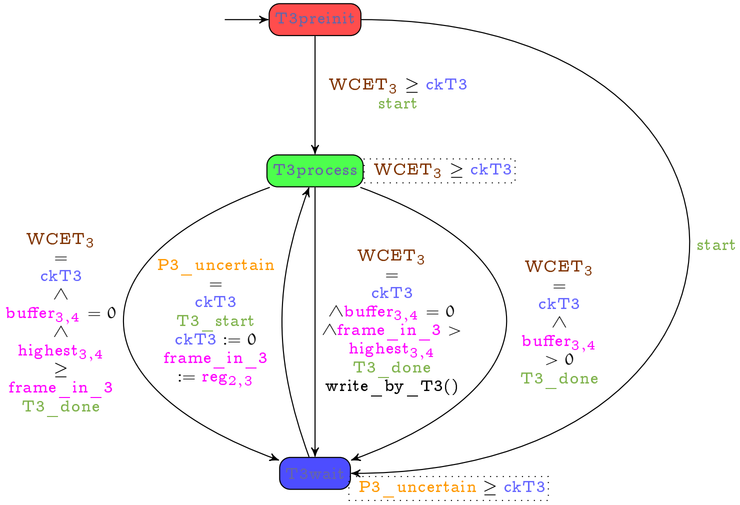
Task T3


T3preinit
$\mathrm{WCET}_3 \geq \mathrm{ckT3}$
start
T3process
$\mathrm{WCET}_3 \geq \mathrm{ckT3}$
start
P3_uncertain
=
ckT3
T3_start
ckT3 := 0
frame_in_3
:= $\mathrm{reg}_{2,3}$
WCET_3
=
ckT3
$\wedge \mathrm{buffer}_{3,4} = 0$
$\wedge$frame_in_3 >
$\mathrm{highest}_{3,4}$
T3_done
write_by_T3()
T3wait
P3_uncertain ≥ ckT3

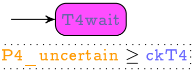
Task T3


T3preinit
WCET$_3$ ≥ ckT3
start
T3process
WCET$_3$ ≥ ckT3
start
P3_uncertain
=
ckT3
T3_start
ckT3 := 0
frame_in_3
:= reg$_{2,3}$
WCET$_3$
=
ckT3
∧buffer$_{3,4}$ = 0
∧frame_in_3 >
highest$_{3,4}$
T3_done
write_by_T3()
WCET$_3$
=
ckT3
∧
buffer$_{3,4}$
> 0
T3_done
T3wait
P3_uncertain ≥ ckT3

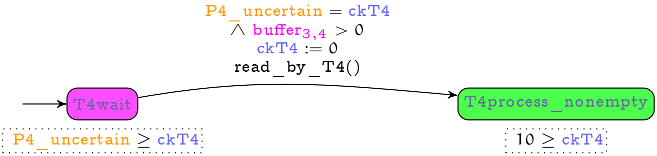
Task T3


T3preinit
$WCET_3 \geq ckT3$
start
T3process
$WCET_3 \geq ckT3$
start
$WCET_3 = ckT3 \wedge buffer_{3,4} = 0 \wedge highest_{3,4} \geq frame\_in\_3$
T3_done
$P3\_uncertain = ckT3$
T3_start
$ckT3 := 0$
$frame\_in\_3 := reg_{2,3}$
$WCET_3 = ckT3 \wedge buffer_{3,4} = 0 \wedge frame\_in\_3 > highest_{3,4}$
T3_done
write_by_T3()
$WCET_3 = ckT3 \wedge buffer_{3,4} > 0$
T3_done
T3wait
$P3\_uncertain \geq ckT3$

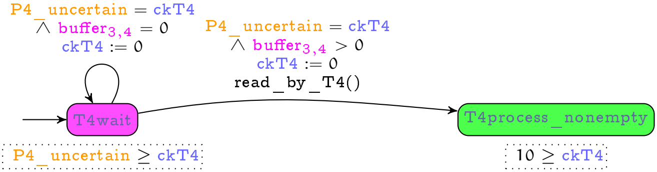
Task T4



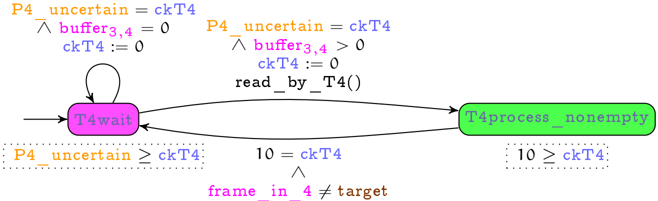
Task T4


P4_uncertain = ckT4
∧ buffer_{3,4} > 0
ckT4 := 0
read_by_T4()
T4wait
T4process_nonempty
P4_uncertain ≥ ckT4
10 ≥ ckT4

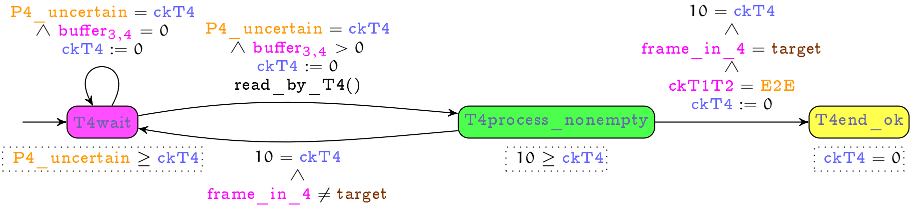
Task T4


P4_uncertain = ckT4
∧ buffer3,4 = 0
ckT4 := 0
P4_uncertain = ckT4
∧ buffer3,4 > 0
ckT4 := 0
read_by_T4()
T4wait
T4process_nonempty
P4_uncertain ≥ ckT4
10 ≥ ckT4

Task T4


P4_uncertain = ckT4
∧ buffer_{3,4} = 0
ckT4 := 0
P4_uncertain = ckT4
∧ buffer_{3,4} > 0
ckT4 := 0
read_by_T4()
T4wait
T4process_nonempty
P4_uncertain ≥ ckT4
10 = ckT4
∧
frame_in_4 ≠ target
10 ≥ ckT4

Task T4


P4_uncertain = ckT4
∧ buffer3,4 = 0
ckT4 := 0
P4_uncertain = ckT4
∧ buffer3,4 > 0
ckT4 := 0
read_by_T4()
10 = ckT4
∧
frame_in_4 = target
∧
ckT1T2 = E2E
ckT4 := 0
T4wait
T4process_nonempty
T4end_ok
P4_uncertain ≥ ckT4
10 = ckT4
∧
frame_in_4 ≠ target
10 ≥ ckT4
ckT4 = 0

Results

E2E latency results for $n = 1$ and $n = 3$

	$n = 1$	$n = 3$
min E2E	63 ms	63 ms
max E2E	145.008 ms	225.016 ms

Results obtained using IMITATOR in a few seconds

Outline

Summary

- Finite-state automata
  - ☺ Mostly decidable results
  - ☺ Efficient model checking algorithms
  - ☹ Miss the quantitative aspects
  - ☺ Many powerful tools
- Timed automata
  - ☺ Finite abstract semantics
  - ☺ Some decidable results
  - ☹ Some undecidable results
  - ☺ Several powerful tools
- Parametric timed automata
  - ☺ Very expressive
  - ☹ No finite abstract semantics
  - ☹ Almost only undecidability results
  - ☺ Some powerful tools

Perspectives

Address harder problems

- Thales challenge: what is the minimum time between two lost frames? (due to the uncertain periods)
  - Requires to model check thousands of frame processings

Improve the efficiency of parameter synthesis techniques

- Promising heuristics: approximations using the integer hull
  [Jovanović et al., 2015, André et al., 2015]
- Distributed parameter synthesis
  - Multi-core synthesis [Laarman et al., 2013]
  - Distributed synthesis based on locations [Zhang et al., 2016]

Beyond (parametric) timed automata

Beyond time...

- Cost, temperature, energy
  - Hybrid automata [Alur et al., 1993a, Alur et al., 1995]
    - Very expressive, but often undecidable
    - Some interesting software (including SpaceEx [Frehse et al., 2011])

Probabilities

- Useful when a property cannot be proved with full certainty
  - Security
- Another way to model systems known with limited precision

Bibliography

References I

Adbeddaïm, Y. and Maler, O. (2002).
Preemptive job-shop scheduling using stopwatch automata.
In *TACAS*, volume 2280 of *LNCS*, pages 113–126. Springer-Verlag.

Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T. A., Ho, P.-H., Nicollin, X., Olivero, A., Sifakis, J., and Yovine, S. (1995).
The algorithmic analysis of hybrid systems.
Theoretical Computer Science, 138(1):3–34.

Alur, R., Courcoubetis, C., Henzinger, T. A., and Ho, P.-H. (1993a).
Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems.
In Grossman, R. L., Nerode, A., Ravn, A. P., and Rischel, H., editors, *Hybrid Systems 1992*, volume 736 of *LNCS*, pages 209–229. Springer.

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. (1993b).
Parametric real-time reasoning.
In *STOC*, pages 592–601. ACM.

References II

André, É. (2017).
What's decidable about parametric timed automata?
International Journal on Software Tools for Technology Transfer.
To appear.

André, É., Fribourg, L., Kühne, U., and Soulat, R. (2012).
IMITATOR 2.5: A tool for analyzing robustness in scheduling problems.
In *FM*, volume 7436 of *LNCS*, pages 33–36. Springer.

André, É. and Lime, D. (2017).
Liveness in L/U-parametric timed automata.
In Legay, A. and Schneider, K., editors, *ACSD*, pages 9–18. IEEE.
To appear.

André, É., Lime, D., and Roux, O. H. (2015).
Integer-complete synthesis for bounded parametric timed automata.
In *RP*, volume 9058 of *LNCS*. Springer.

André, É., Lime, D., and Roux, O. H. (2016).
Decision problems for parametric timed automata.
In Ogata, K., Lawford, M., and Liu, S., editors, *ICFEM*, volume 10009 of *LNCS*, pages 400–416. Springer.

References III

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

Annichini, A., Bouajjani, A., and Sighireanu, M. (2001).
TReX: A tool for reachability analysis of complex systems.
In Berry, G., Comon, H., and Finkel, A., editors, *CAV*, volume 2102 of *LNCS*, pages 368–372. Springer.

Baier, C. and Katoen, J.-P. (2008).
Principles of Model Checking.
MIT Press.

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 *LNCS*, pages 69–81. Springer.

Bozzelli, L. and La Torre, S. (2009).
Decision problems for lower/upper bound parametric timed automata.
Formal Methods in System Design, 35(2):121–151.

References IV

Bundala, D. and Ouaknine, J. (2014).
Advances in parametric real-time reasoning.
In *MFCS*, volume 8634 of *LNCS*, pages 123–134. Springer.

Fanchon, L. and Jacquemard, F. (2013).
Formal timing analysis of mixed music scores.
In *ICMC (International Computer Music Conference)*.

Frehse, G., Le Guernic, C., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., and Maler, O. (2011).
SpaceEx: Scalable verification of hybrid systems.
In Gopalakrishnan, G. and Qadeer, S., editors, *CAV*, volume 6806 of *LNCS*, pages 379–395. Springer.

Fribourg, L., Lesens, D., Moro, P., and Soulat, R. (2012).
Robustness analysis for scheduling problems using the inverse method.
In *TIME*, pages 73–80. IEEE Computer Society Press.

Henzinger, T. A., Ho, P.-H., and Wong-Toi, H. (1997).
HyTech: A model checker for hybrid systems.
International Journal on Software Tools for Technology Transfer, 1:110–122.

References V

Hune, T., Romijn, J., Stoelinga, M., and Vaandrager, F. W. (2002).
Linear parametric model checking of timed automata.
Journal of Logic and Algebraic Programming, 52-53:183–220.

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

Laarman, A., Olesen, M. C., Dalsgaard, A. E., Larsen, K. G., and Van De Pol, J. (2013).
Multi-core emptiness checking of timed Büchi automata using inclusion abstraction.
In *CAV*, volume 8044 of *LNCS*, pages 968–983, Heidelberg, Germany. Springer.

Larsen, K. G., Pettersson, P., and Yi, W. (1997).
UPPAAL in a nutshell.
International Journal on Software Tools for Technology Transfer, 1(1-2):134–152.

Lime, D., Roux, O. H., Seidner, C., and Traonouez, L.-M. (2009).
Romeo: A parametric model-checker for Petri nets with stopwatches.
In Kowalewski, S. and Philippou, A., editors, *TACAS*, volume 5505 of *LNCS*, pages 54–57. Springer.

References VI

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

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

Sun, J., Liu, Y., Dong, J. S., and Pang, J. (2009).
PAT: Towards flexible verification under fairness.
In *CAV*, volume 5643 of *LNCS*, pages 709–714. Springer.

Sun, Y., Soulat, R., Lipari, G., André, É., and Fribourg, L. (2013).
Parametric schedulability analysis of fixed priority real-time distributed systems.
In *FTSCS*, volume 419 of *CCIS*, pages 212–228. Springer.

Wang, T., Sun, J., Wang, X., Liu, Y., Si, Y., Dong, J. S., Yang, X., and Li, X. (2015).
A systematic study on explicit-state non-zenoness checking for timed automata.
IEEE Transactions on Software Engineering, 41(1):3–18.

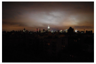
References VII

Yovine, S. (1997).
Kronos: A verification tool for real-time systems.
International Journal on Software Tools for Technology Transfer, 1(1-2):123–133.

Zhang, Z., Nielsen, B., and Larsen, K. G. (2016).
Distributed algorithms for time optimal reachability analysis.
In *FORMATS 2016*, volume 9884 of *LNCS*, pages 157–173. Springer.

Additional explanation

Explanation for the 4 pictures in the beginning



Allusion to the Northeast blackout (USA, 2003)
Computer bug
Consequences: 11 fatalities, huge cost
(Picture actually from the Sandy Hurricane, 2012)

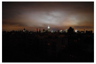
Error screen on the earliest versions of Macintosh

Allusion to the sinking of the Sleipner A offshore platform (Norway, 1991)
No fatalities
Computer bug: inaccurate finite element analysis modeling
(Picture actually from the Deepwater Horizon Offshore Drilling Platform)

Allusion to the MIM-104 Patriot Missile Failure (Iraq, 1991)
28 fatalities, hundreds of injured
Computer bug: software error (clock drift)
(Picture of an actual MIM-104 Patriot Missile, though not the one of 1991)

Licensing

Source of the graphics used I



Title: Hurricane Sandy Blackout New York Skyline
Author: David Shankbone
Source: https://commons.wikimedia.org/wiki/File:Hurricane_Sandy_Blackout_New_York_Skyline.JPG
License: CC BY 3.0

Title: Sad mac
Author: Przemub
Source: https://commons.wikimedia.org/wiki/File:Sad_mac.png
License: Public domain

Title: Deepwater Horizon Offshore Drilling Platform on Fire
Author: ideum
Source: https://secure.flickr.com/photos/ideum/4711481781/
License: CC BY-SA 2.0

Title: DA-SC-88-01663
Author: imcomkorea
Source: https://secure.flickr.com/photos/imcomkorea/3017886760/
License: CC BY-NC-ND 2.0

Source of the graphics used II

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

Title: Renault Twizy
Author: Citron
Source: https://commons.wikimedia.org/wiki/File:Renault_Twizy.jpg
License: CC BY-SA 3.0

Title: Airbus A380
Author: Axwel
Source: https://commons.wikimedia.org/wiki/File:Airbus_A380_blue_sky.jpg
License: CC BY 2.0

Source of the graphics used III

License of this document