TIPS'18

2018年9月8日星期六
北京，中國

Parametric timed automata: modeling and verifying real-time systems under uncertainty

Étienne André

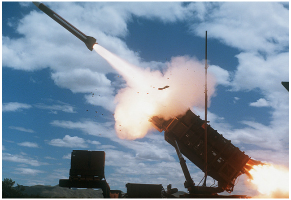
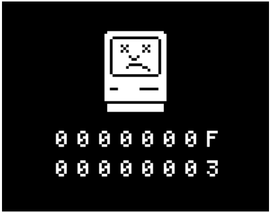
LIPN, Université Paris 13, CNRS, France

Feat. joint works with Giuseppe Lipari, and Sun Youcheng

Context: Verifying complex timed systems

- Real-time systems are everywhere
  - Hard timing constraints and concurrency
  - Criticality: risk for huge damages in case of unexpected behavior (bug)
  - Bugs discovered when final testing: expensive
  - ⇝ Need for a thorough specification and verification phase



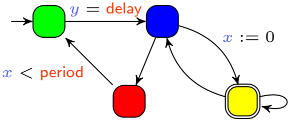
Outline

1. Parametric timed automata
2. Modeling and verifying real-time systems with parameters
3. A case study: Verifying a real-time system under uncertainty
4. Conclusion and perspectives

Outline

Model checking timed concurrent systems

- Use formal methods [Baier and Katoen, 2008]



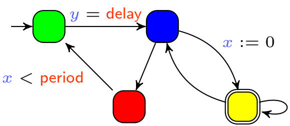
A model of the system

 is unreachable

A property to be satisfied

Model checking timed concurrent systems

- Use formal methods [Baier and Katoen, 2008]



A model of the system

$\overset{?}{\models}$

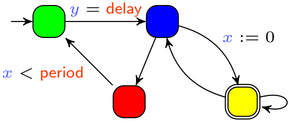
 is unreachable

A property to be satisfied

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

Model checking timed concurrent systems

- Use formal methods [Baier and Katoen, 2008]



A model of the system

$\models^{?}$

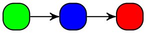
 is unreachable

A property to be satisfied

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

Yes

No



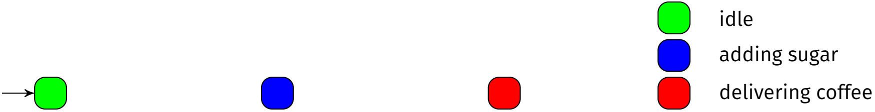
Counterexample

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

Outline

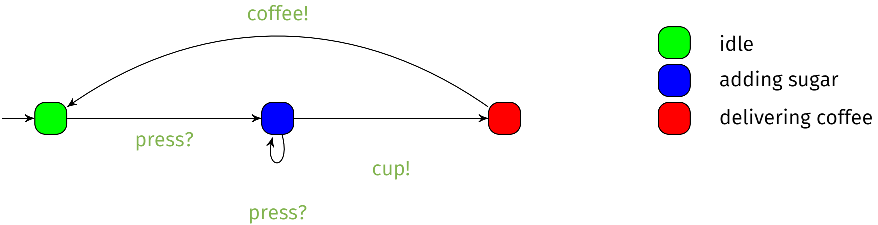
Timed automaton (TA)

- Finite state automaton (sets of locations)



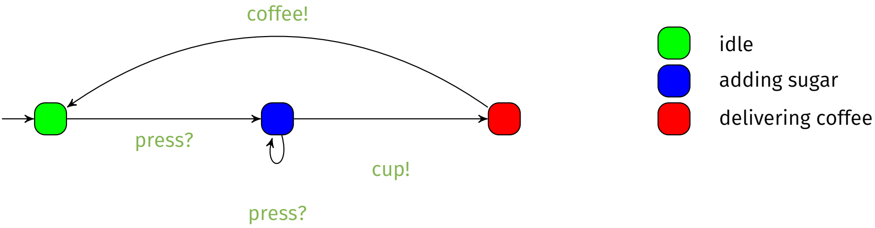
Timed automaton (TA)

- Finite state automaton (sets of locations and actions)



Timed automaton (TA)

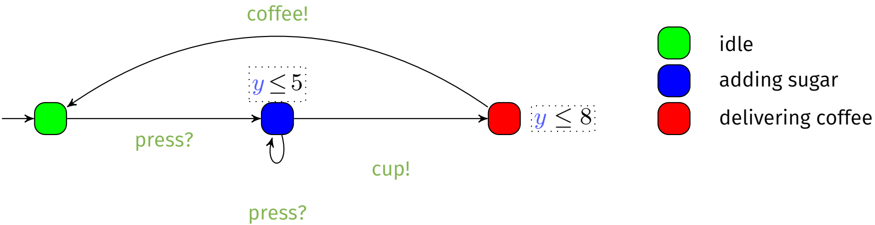
- Finite state automaton (sets of locations and actions) augmented with a set X of clocks [Alur and Dill, 1994]
  - Real-valued variables evolving linearly at the same rate



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
  - Can be compared to integer constants in invariants

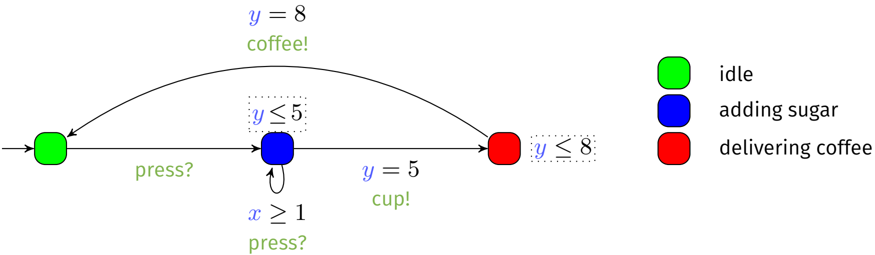
- Features
  - Location invariant: property to be verified to stay at a location



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

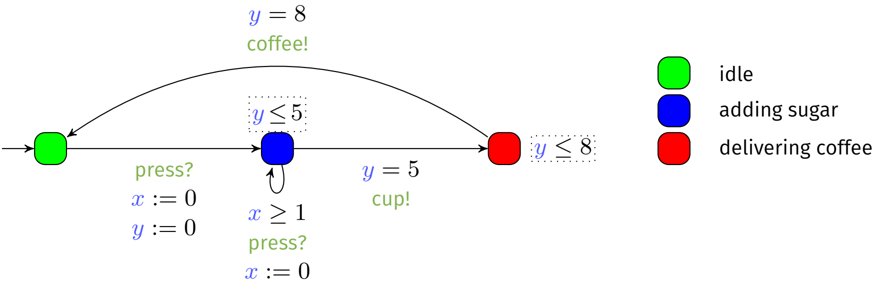
- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property to be verified to enable a transition



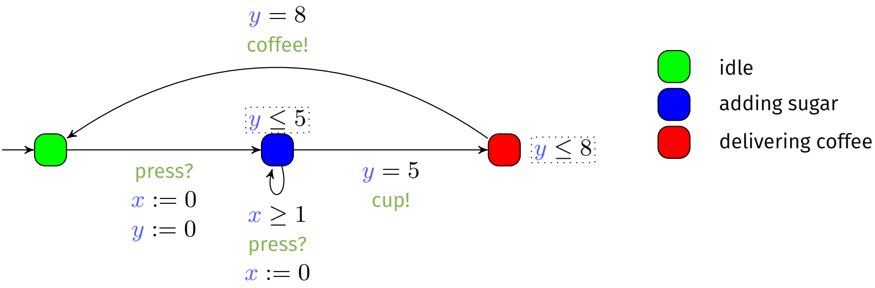
Timed automaton (TA)

- 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

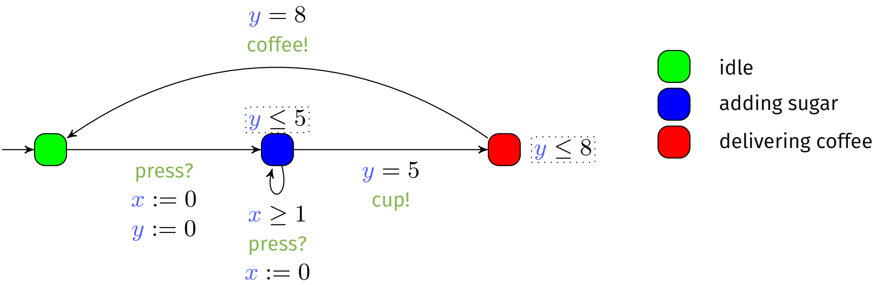
- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property to be verified to enable a transition
  - Clock reset: some of the clocks can be set to 0 along transitions



The most critical system: The coffee machine



The most critical system: The coffee machine

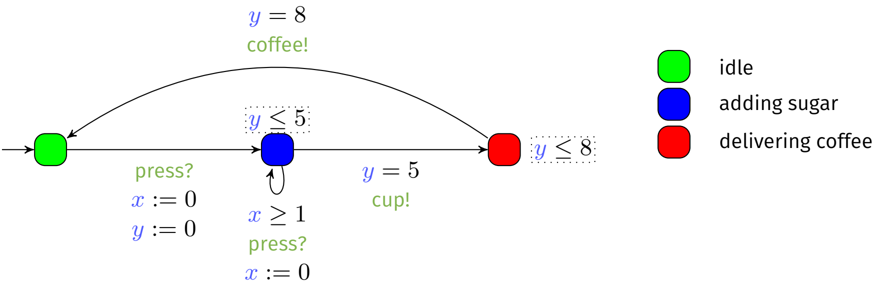


- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar

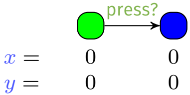
$x =$ 0

$y =$ 0

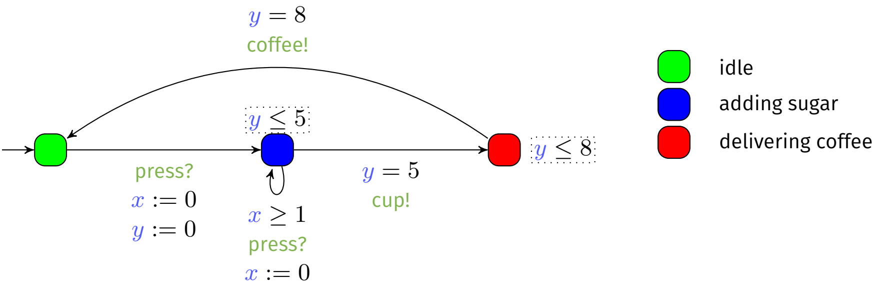
The most critical system: The coffee machine



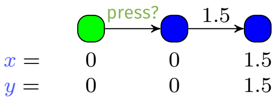
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



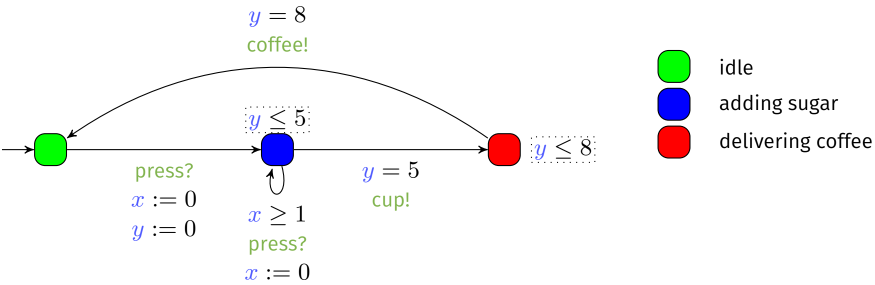
The most critical system: The coffee machine



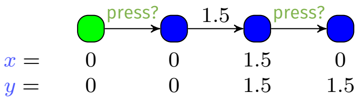
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



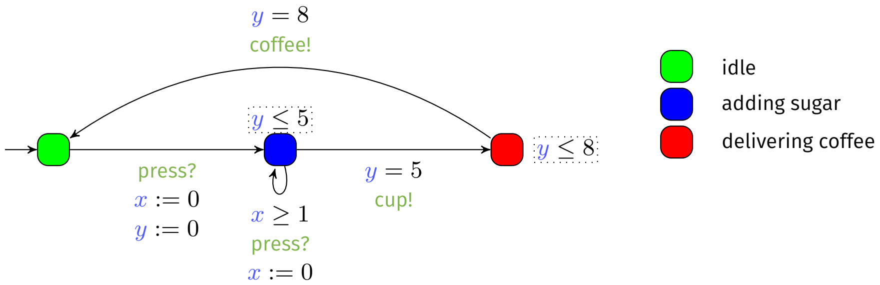
The most critical system: The coffee machine



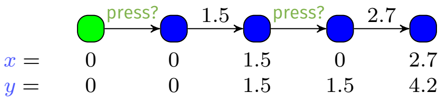
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



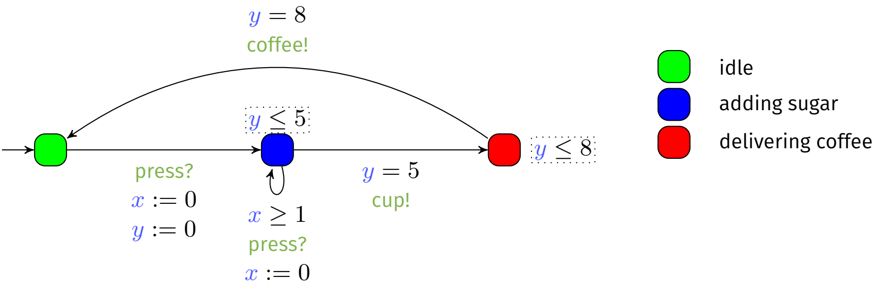
The most critical system: The coffee machine



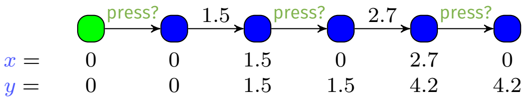
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



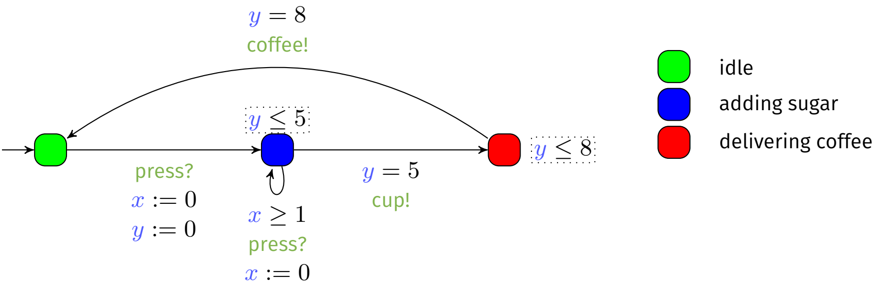
The most critical system: The coffee machine



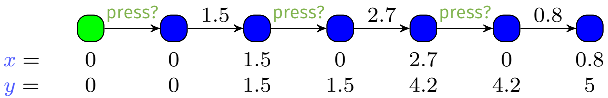
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



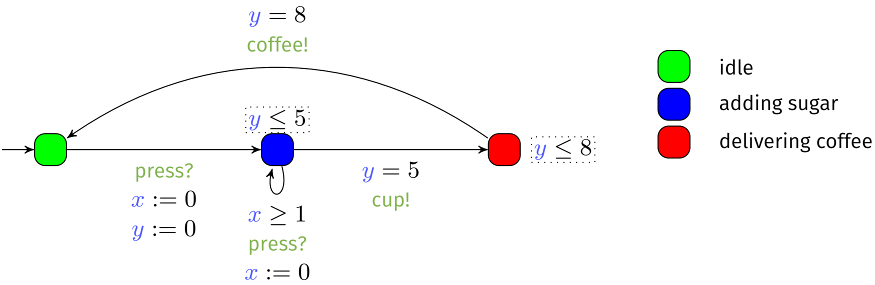
The most critical system: The coffee machine



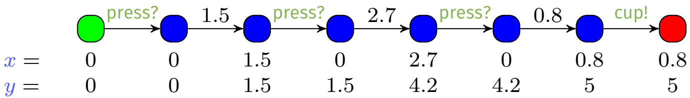
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



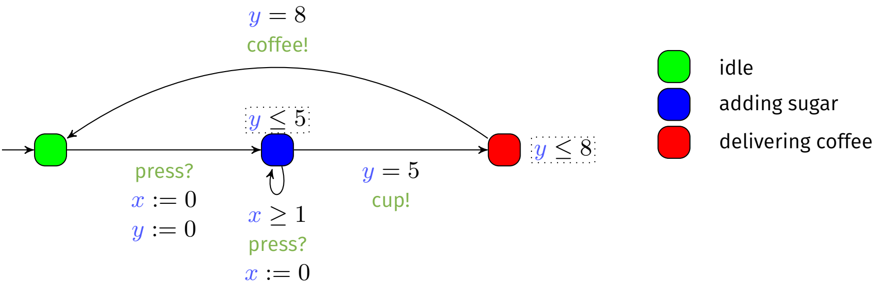
The most critical system: The coffee machine



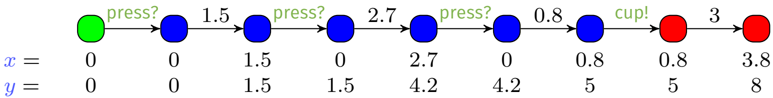
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



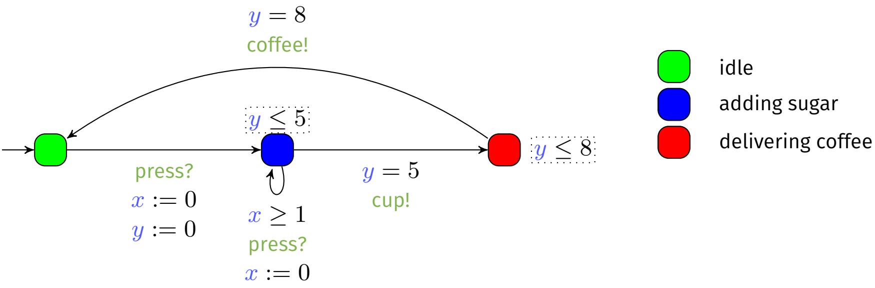
The most critical system: The coffee machine



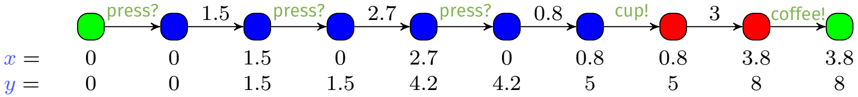
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



The most critical system: The coffee machine



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



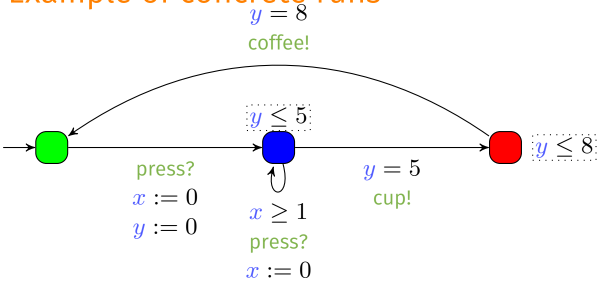
Concrete semantics of timed automata

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

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

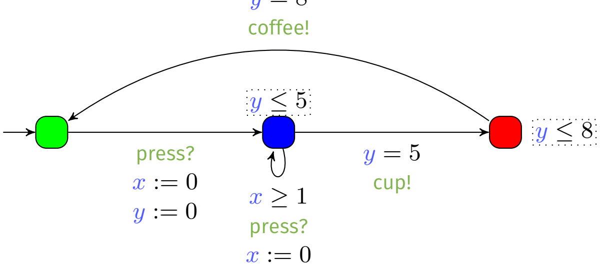
- Concrete run: alternating sequence of concrete states and actions or time elapse

Example of concrete runs

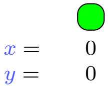


- Possible concrete runs for the coffee machine

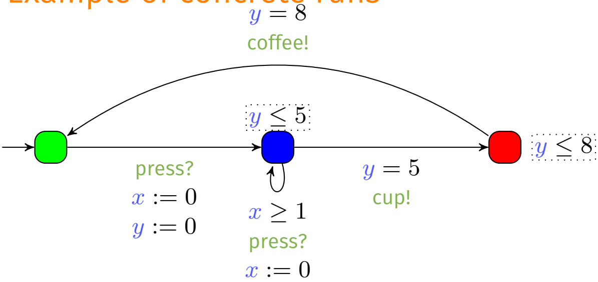
Example of concrete runs



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



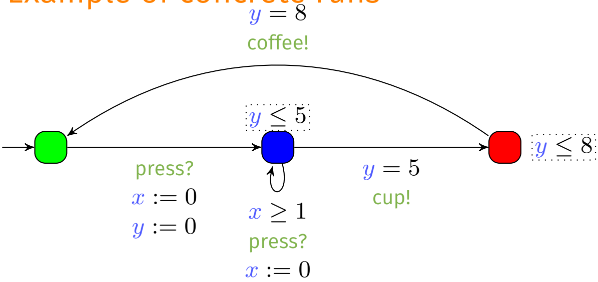
Example of concrete runs



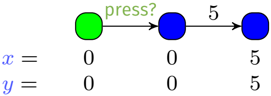
- Possible concrete runs for the coffee machine
  - Coffee with no sugar

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

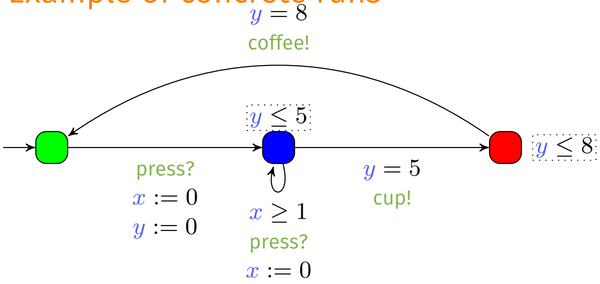
Example of concrete runs



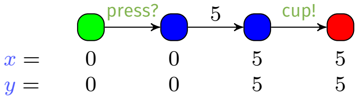
- Possible concrete runs for the coffee machine
  - Coffee with no sugar



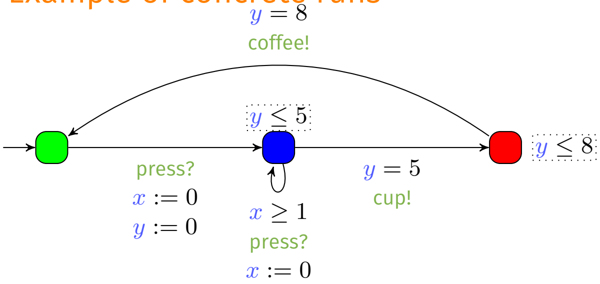
Example of concrete runs



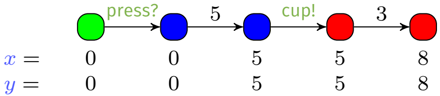
- Possible concrete runs for the coffee machine
  - Coffee with no sugar



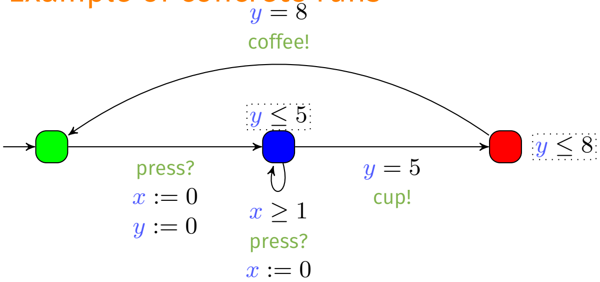
Example of concrete runs



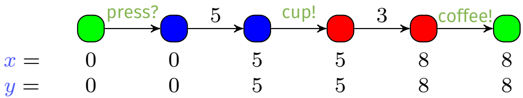
- Possible concrete runs for the coffee machine
  - Coffee with no sugar



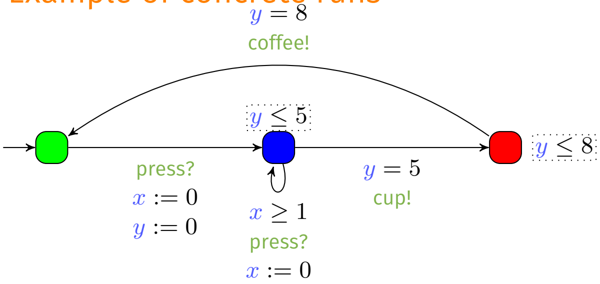
Example of concrete runs



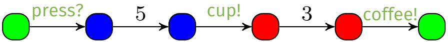
- Possible concrete runs for the coffee machine
  - Coffee with no sugar



Example of concrete runs



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

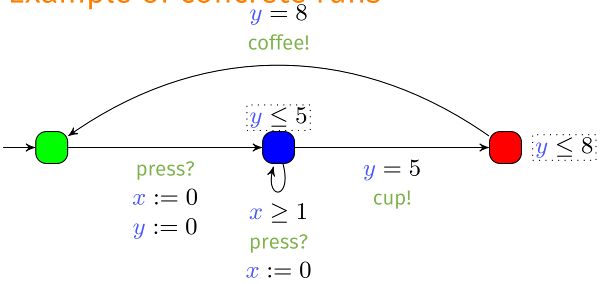


$x =$	0	0	5	5	8	8
$y =$	0	0	5	5	8	8

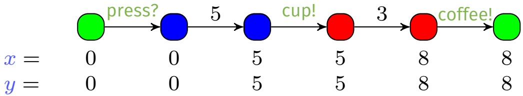
  - Coffee with 2 doses of sugar

$x =$	0
$y =$	0

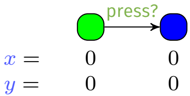
Example of concrete runs



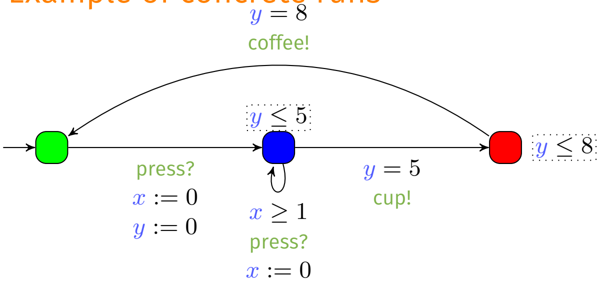
- Possible concrete runs for the coffee machine
  - Coffee with no sugar



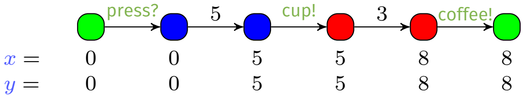
  - Coffee with 2 doses of sugar



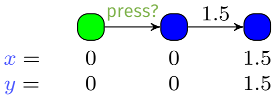
Example of concrete runs



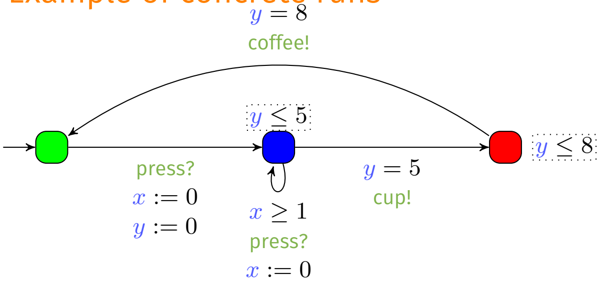
- Possible concrete runs for the coffee machine
  - Coffee with no sugar



  - Coffee with 2 doses of sugar



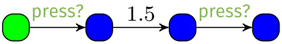
Example of concrete runs



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

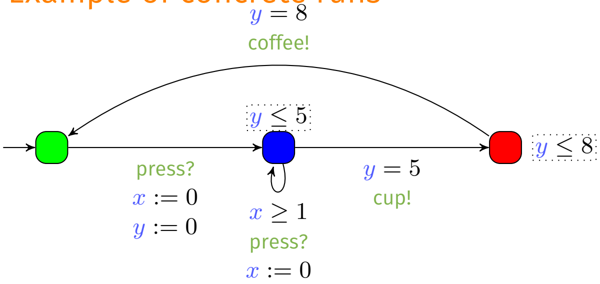
$x =$	0	0	5	5	8	8
$y =$	0	0	5	5	8	8

  - Coffee with 2 doses of sugar



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

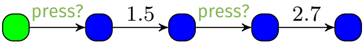
Example of concrete runs



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

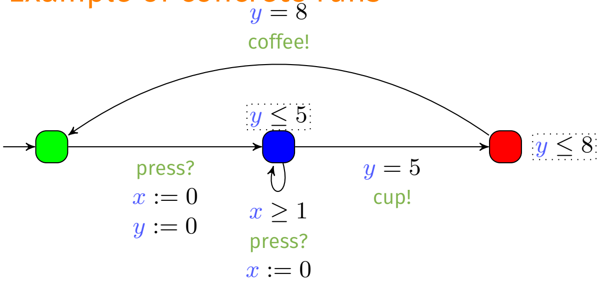
$x =$	0	0	5	5	8	8
$y =$	0	0	5	5	8	8

  - 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



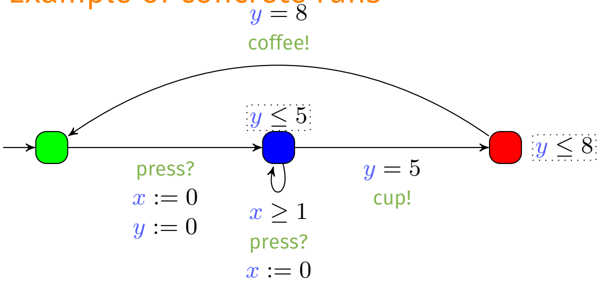
- Possible concrete runs for the coffee machine
  - Coffee with no sugar

$x =$	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

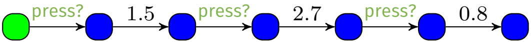
Example of concrete runs



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

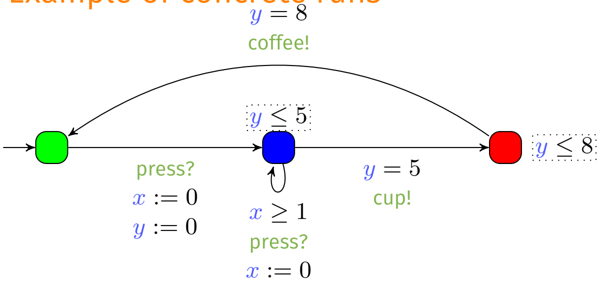
$x =$	0	0	5	5	8	8
$y =$	0	0	5	5	8	8

  - 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

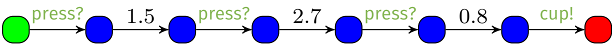
Example of concrete runs



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

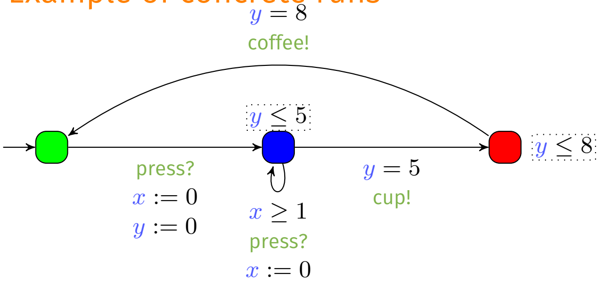
$x =$	0	0	5	5	8	8
$y =$	0	0	5	5	8	8

  - Coffee with 2 doses of sugar

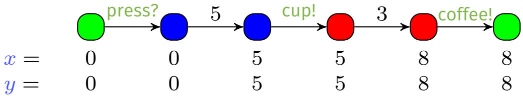


$x =$	0	0	1.5	0	2.7	0	0.8	0.8
$y =$	0	0	1.5	1.5	4.2	4.2	5	5

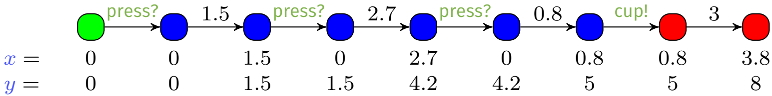
Example of concrete runs



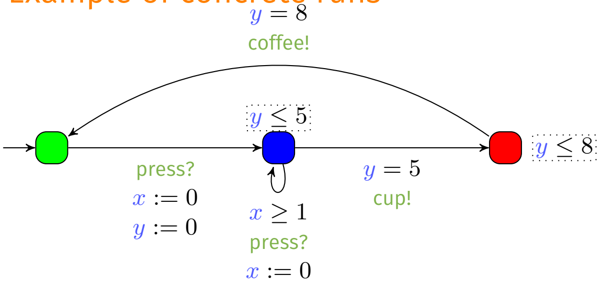
- Possible concrete runs for the coffee machine
  - Coffee with no sugar



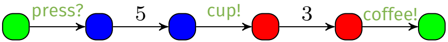
  - Coffee with 2 doses of sugar



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	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

Timed automata: A success story

- An expressive formalism
  - Dense time
  - Concurrency
- A tractable verification in theory
  - Reachability is PSPACE-complete [Alur and Dill, 1994]
- A very efficient verification in practice
  - Symbolic verification: relatively insensitive to constants
  - Several model checkers, notably UPPAAL [Larsen et al., 1997]
  - Long list of successful case studies

Outline

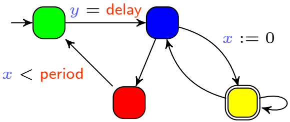
Beyond timed model checking: parameter synthesis

- 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



A model of the system

$\stackrel{?}{\models}$

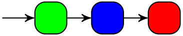
 is unreachable

A property to be satisfied

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

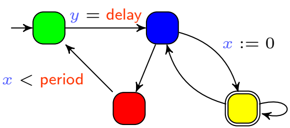
Yes

No



Counterexample

Parametric timed model checking



A model of the system

$\overset{?}{\models}$

 is unreachable

A property to be satisfied

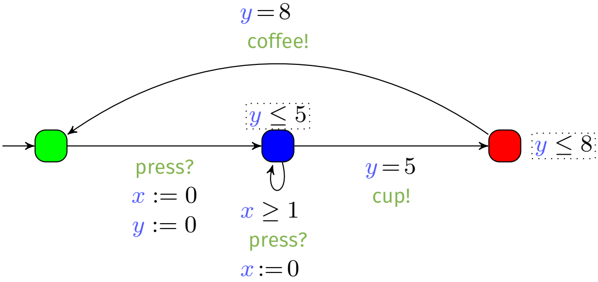
- Question: for what values of the parameters does the model of the system satisfy the property?

Yes if...

$$2\text{delay} > \text{period} \\ \land \text{period} < 20.46$$

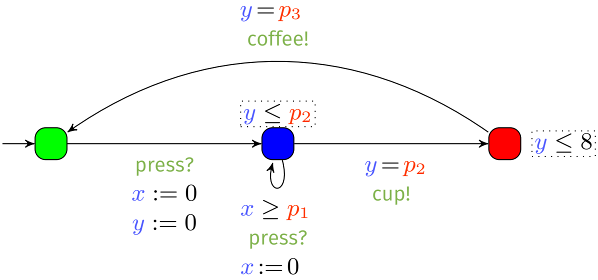
Parametric Timed Automaton (PTA)

- 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

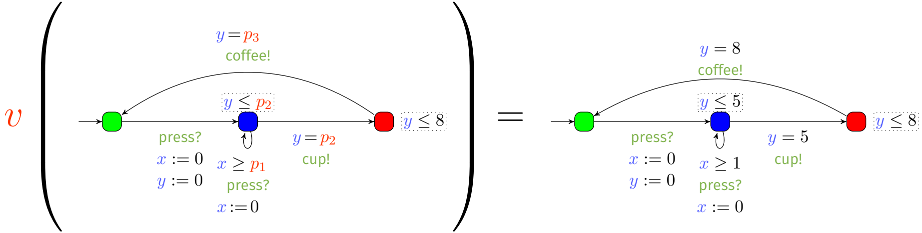


Notation: Valuation of a PTA

- 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 $v(p)$

Notation: 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 $v(p)$



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

Symbolic semantics of parametric timed automata

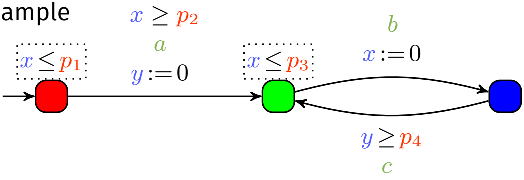
- Symbolic state of a PTA: pair (l, C), where
  - l is a location,
  - C is a convex polyhedron over X and P with a special form, called parametric zone [Hune et al., 2002]

Symbolic semantics of parametric timed automata

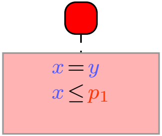
- Symbolic state of a PTA: pair (l, C), where
  - l is a location,
  - C is a convex polyhedron over X and P with a special form, called parametric zone [Hune et al., 2002]
- Symbolic run: alternating sequence of symbolic states and actions

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

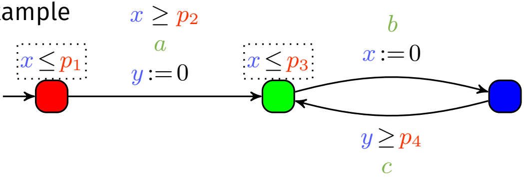


  - Possible symbolic run for this PTA

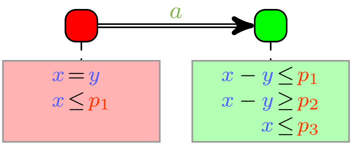


Symbolic semantics of parametric timed automata

- 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

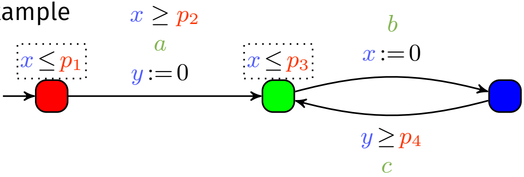


  - Possible symbolic run for this PTA

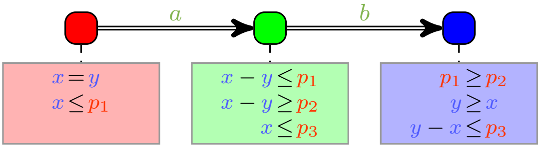


Symbolic semantics of parametric timed automata

- 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

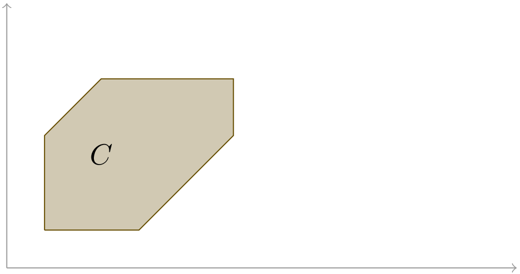


  - Possible symbolic run for this PTA



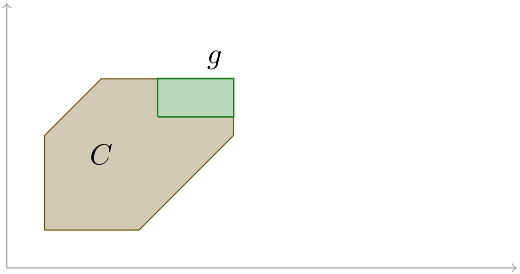
Symbolic semantics of PTA: Illustration

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



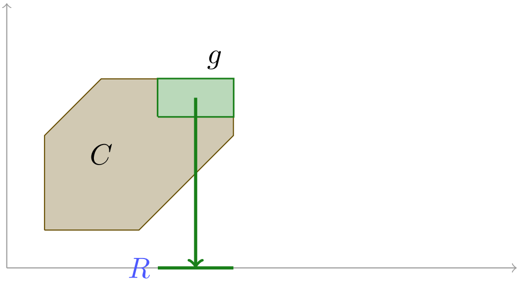
Symbolic semantics of PTA: Illustration

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



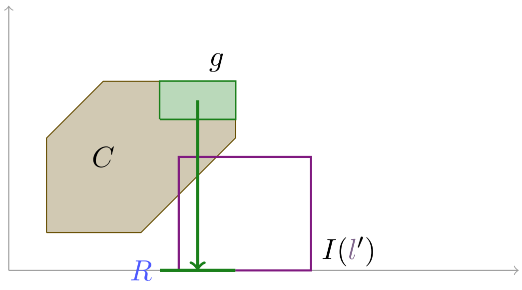
Symbolic semantics of PTA: Illustration

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



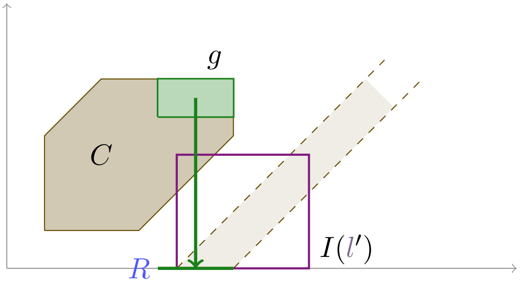
Symbolic semantics of PTA: Illustration

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



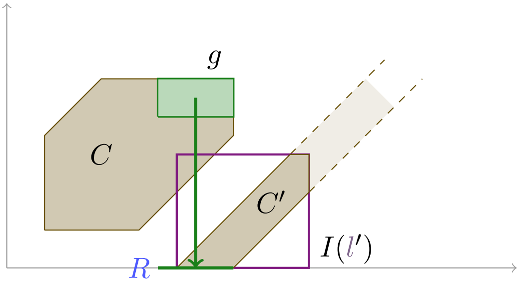
Symbolic semantics of PTA: Illustration

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

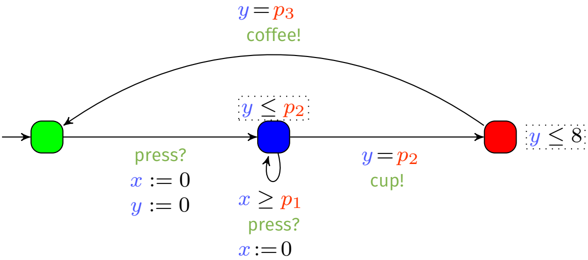


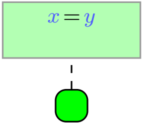
Symbolic semantics of PTA: Illustration

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

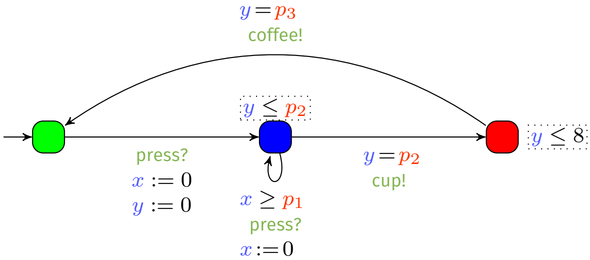


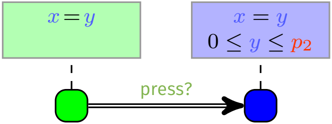
Symbolic exploration: Coffee machine



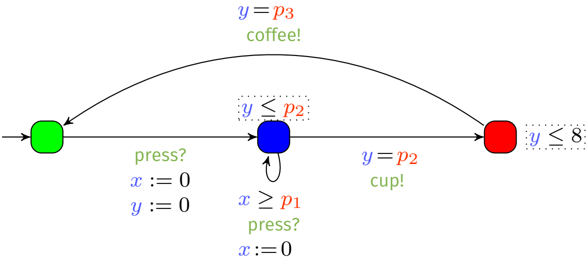


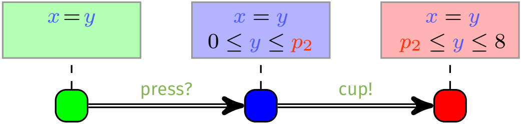
Symbolic exploration: Coffee machine



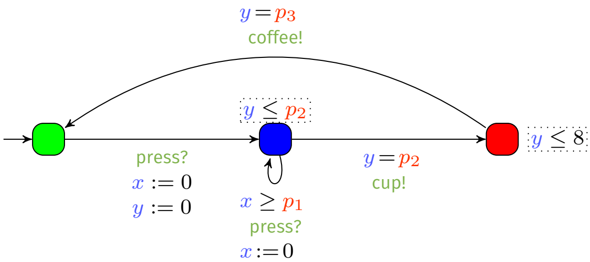


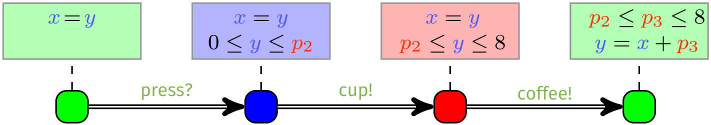
Symbolic exploration: Coffee machine



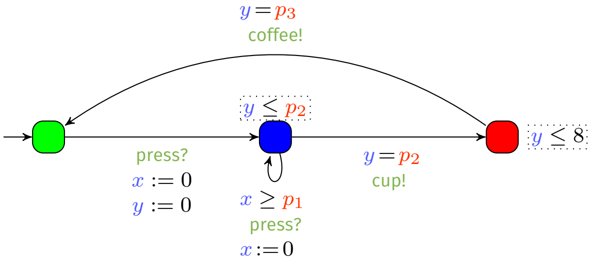


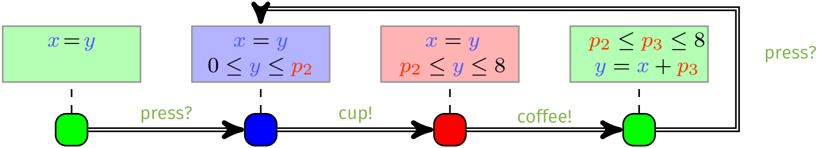
Symbolic exploration: Coffee machine



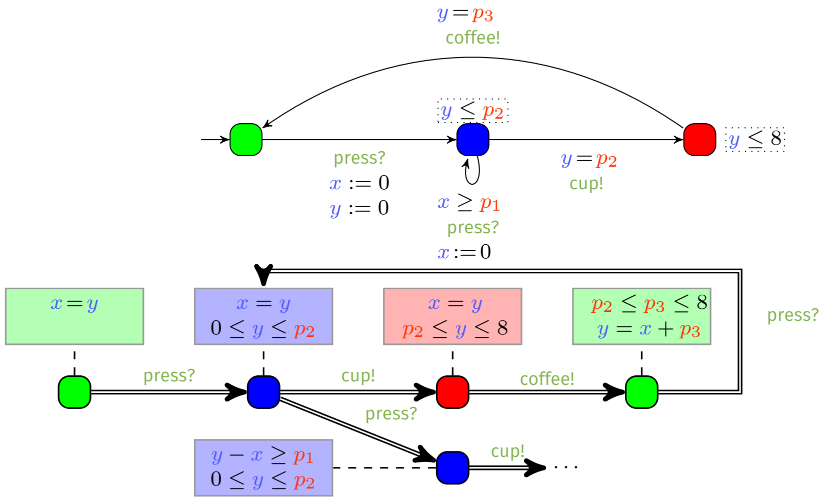


Symbolic exploration: Coffee machine

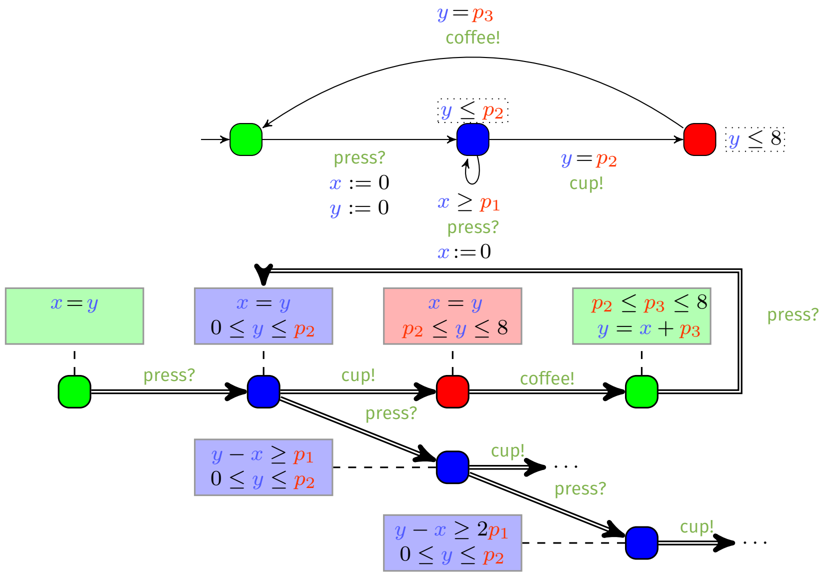




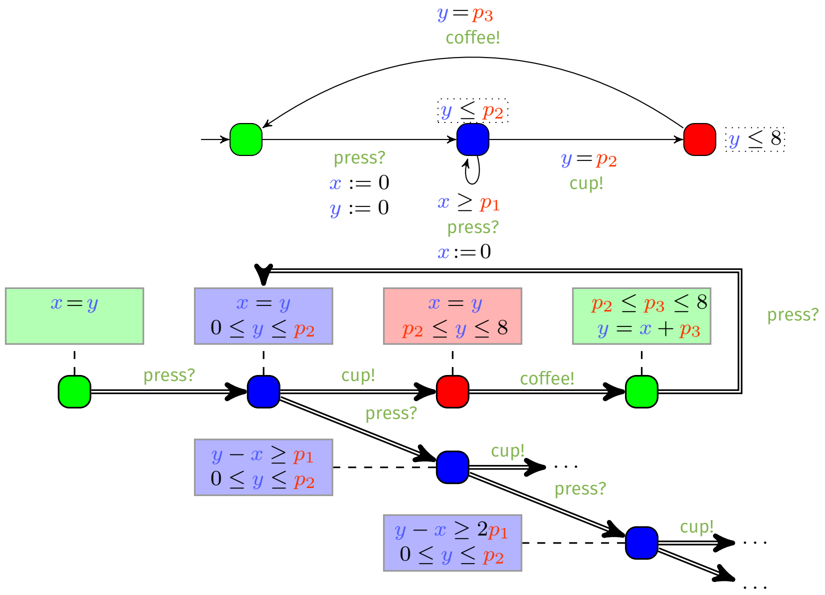
Symbolic exploration: Coffee machine



Symbolic exploration: Coffee machine



Symbolic exploration: Coffee machine



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 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?"

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

  Example: "Does there exist at least one parameter valuation for which I can always eventually get a coffee?"

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?"

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

  Example: "Does there exist at least one parameter valuation for which I can always eventually get a coffee?"

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., $p_1 = 1, p_2 = 5, p_3 = 2$

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

  Example: "Does there exist at least one parameter valuation for which I can always eventually get a coffee?"

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?" √, 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?" ×, e. g., $p_1 = 1, p_2 = 5, p_3 = 2$

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

  Example: "Does there exist at least one parameter valuation for which I can always eventually get a coffee?" √, e. g., $p_1 = 1, p_2 = 5, p_3 = 8$

Undecidability

- The symbolic state space is infinite in general
- No finite abstraction exists (unlike timed automata)

Undecidability

- The symbolic state space is infinite in general
- No finite abstraction exists (unlike timed automata)

Bad news

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

[ÉA, STTT 2017]

Outline

Outline

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)

- It is very important to check that all tasks meet their deadlines

Schedulability analysis

Real-time system:

- Set of tasks (with a period, a WCET and a deadline)
- One processor (uniprocessor) or more (multiprocessor)
- Scheduling policies: fixed priority (FPS), earliest deadline first (EDF)...

Definition (Schedulability analysis)

Given a real-time system and a scheduling policy, certify that no deadline miss will ever occur

Schedulability analysis

Real-time system:

- Set of tasks (with a period, a WCET and a deadline)
- One processor (uniprocessor) or more (multiprocessor)
- Scheduling policies: fixed priority (FPS), earliest deadline first (EDF)...

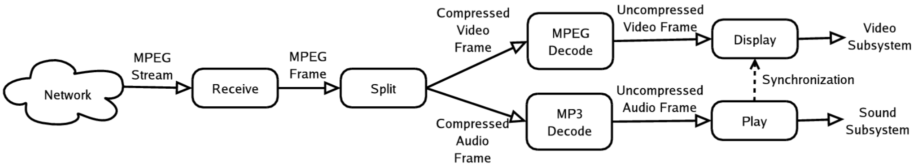
Definition (Schedulability analysis)

Given a real-time system and a scheduling policy, certify that no deadline miss will ever occur

In general, schedulability analysis is hard

Real-time pipelines

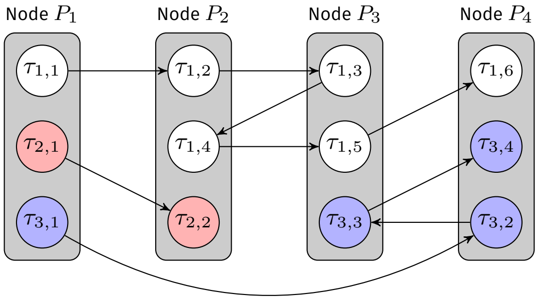
- Many real-time applications can be modeled as pipelines (also called transactions) of tasks



- 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

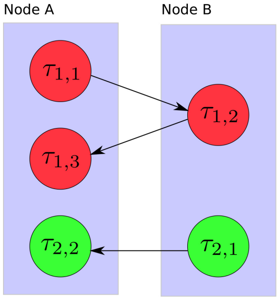


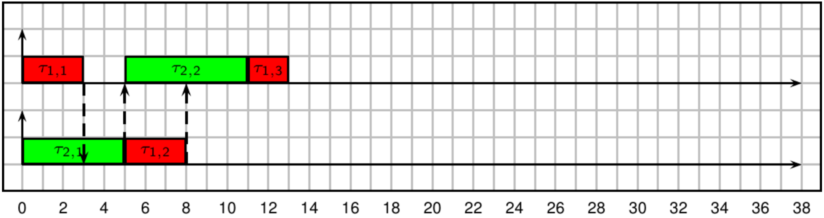
- Scheduling Problem: Guarantee that all pipelines complete before their E2E deadlines

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

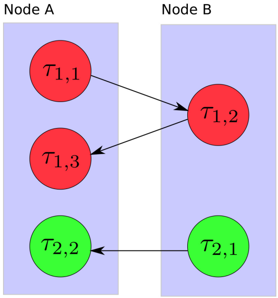


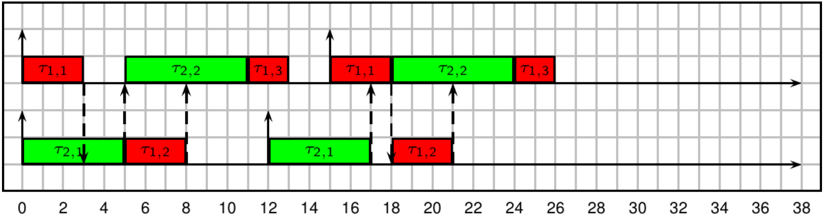


Activations, jitter, deadline

- 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

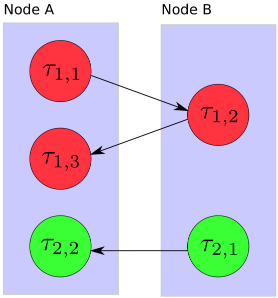


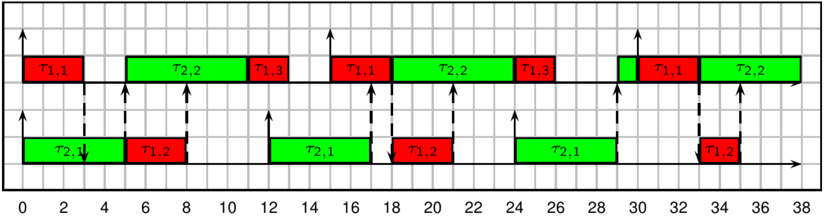


Activations, jitter, deadline

- 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





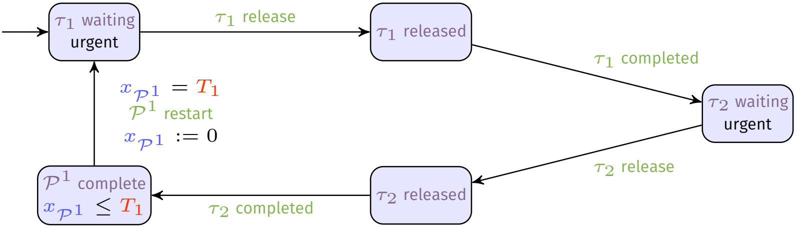
Outline

Enriching the model with parameters

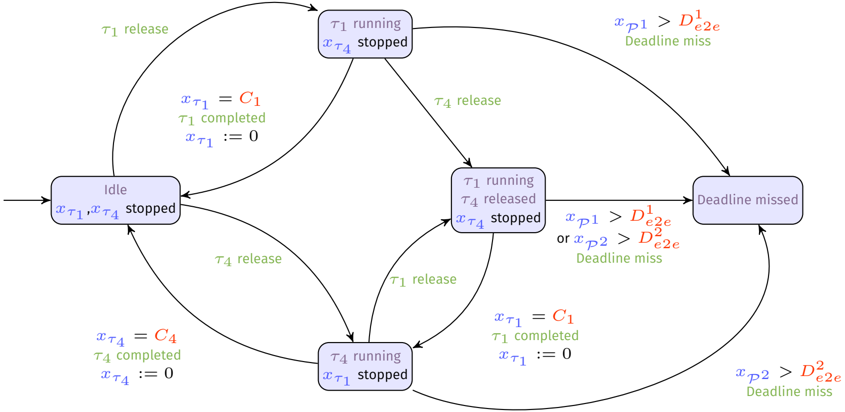
- A task is identified by three parameters:
  - C_i is the worst-case computation time (or worst-case transmission time, in case it models a message)
  - R_i is the task worst-case response time, i. e., the worst case finishing time of any task instance relative to the activation of its pipeline.
  - 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]

Schedulability analysis with parametric model checking

Goal: parametric schedulability analysis

Given a real-time system and a scheduling policy, synthesize valuations (deadlines, periods...) such that the system is schedulable.

Modeling a real-time system with PTAs

- Each task or chain of task: one PTA
- Each scheduler: one PTA
- Use stopwatches to model preemption

Schedulability analysis with parametric model checking

Goal: parametric schedulability analysis

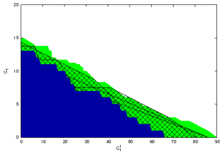
Given a real-time system and a scheduling policy, synthesize valuations (deadlines, periods...) such that the system is schedulable.

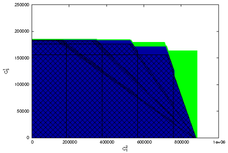
Modeling a real-time system with PTAs

- Each task or chain of task: one PTA
- Each scheduler: one PTA
- Use stopwatches to model preemption

Comparison with analytical methods [Sun, Soulat, Lipari, André, Fribourg, FTSCS'13]

- Slower but much better in terms of completeness
- And can evaluate robustness



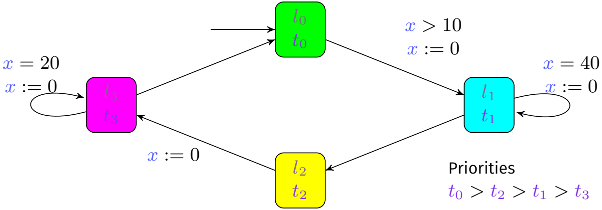


Outline

A unified formalism: Parametric task automata

Extension of task automata [Norström et al., 1999, Fersman et al., 2007] with parameters

[André, FMICS'17]

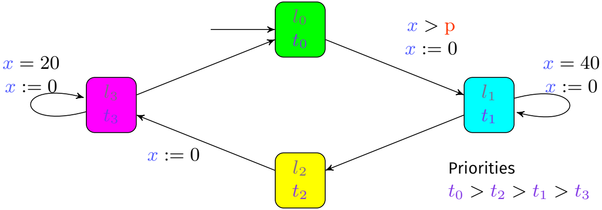


Task	B	W	D
t_0	0	1	2
t_1	4	4	20
t_2	0	1	4
t_3	2	2	10

A unified formalism: Parametric task automata

Extension of task automata [Norström et al., 1999, Fersman et al., 2007] with parameters

[André, FMICS'17]

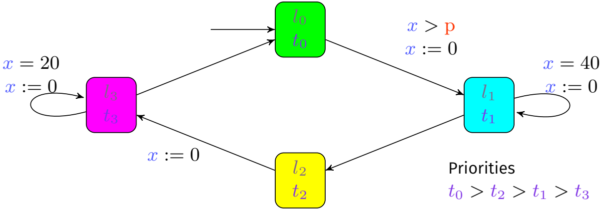


Task	B	W	D
t_0	0	1	2
t_1	4	4	20
t_2	0	1	p'
t_3	2	2	10

A unified formalism: Parametric task automata

Extension of task automata [Norström et al., 1999, Fersman et al., 2007] with parameters

[André, FMICS'17]



Task	B	W	D
t_0	0	1	2
t_1	4	4	20
t_2	0	1	p'
t_3	2	2	10

Parametric task automata can model

- Preemption
- Periodic tasks, sporadic tasks, pseudo-periodic tasks...
- Dependencies between tasks
- Offset, jitter
- Uncertainty
- Uniprocessor only

Parametric task automata: theory and practice

Schedulability-emptiness ("is the set of valuations for which the system is schedulable empty?")

- Undecidable in general
- Decidable under some assumptions [André, FMICS'17]

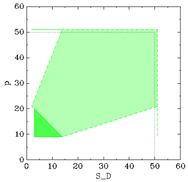
Parametric task automata: theory and practice

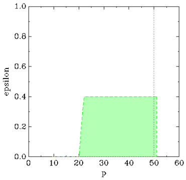
Schedulability-emptiness (“is the set of valuations for which the system is schedulable empty?”)

- Undecidable in general
- Decidable under some assumptions [André, FMICS’17]

Implementation in IMITATOR

- Translation into a network of parametric stopwatch automata
- Schedulability analysis
- Parametric and/or robust schedulability analysis





Outline

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

  [ÉA, 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

Parametric model checking applied to schedulability analysis

Advantages: Very expressive (in fact Turing-complete)

- ☺ Periodic tasks, sporadic tasks...
- ☺ Task dependencies
- ☺ Data transmission
- ☺ Jitters, offsets
- ☺ Uncertainty
- ☺ All possible schedulers

Drawback: slow (intractable for very large systems)

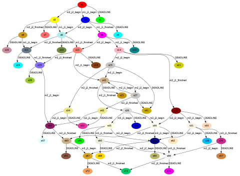
- ☹ limited to small or medium-sized systems

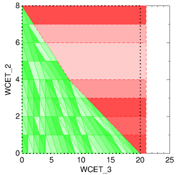
Outline

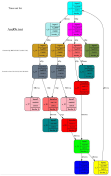
IMITATOR

- A tool for modeling and verifying timed concurrent 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

- Synthesis algorithms
  - (non-Zeno) parametric model checking (using a subset of TCTL)
  - Language and trace preservation, and robustness analysis
  - Parametric deadlock-freeness checking







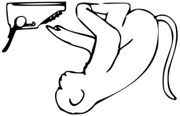
IMITATOR

Under continuous development since 2008 [André et al., FM'12]

A library of benchmarks

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

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



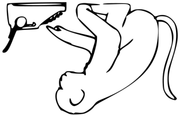
IMITATOR

Under continuous development since 2008 [André et al., FM'12]

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
- 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]
- Verification of software product lines [Luthmann et al., 2017]
- Offline monitoring [ÉA, Hasuo, Waga @ ICECCS'18]
- Formal timing analysis of music scores [Fanchon and Jacquemard, 2013]
- Solution to a challenge related to a distributed video processing system by Thales

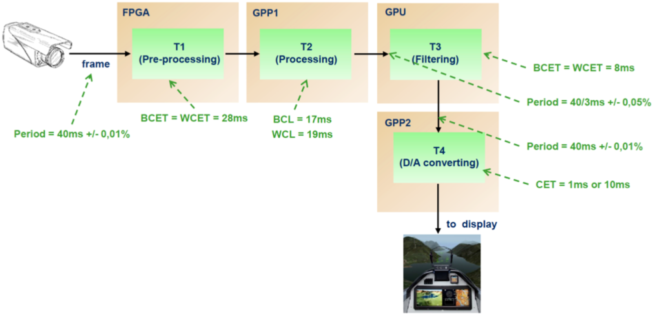
Outline

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): $\mathrm{E2E}$
- Others:
  - the register between task 2 and task 3: discrete variable $\mathrm{reg}_{2,3}$
  - the buffer between task 3 and task 4: $n = 1$ or $n = 3$

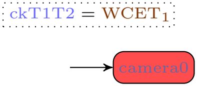
Simplification

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

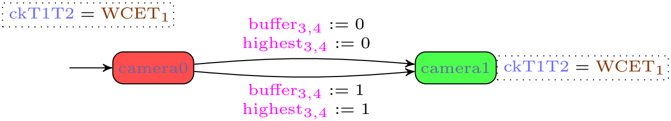
Simplification

- 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?)

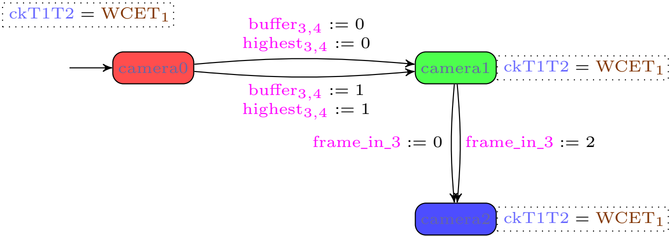
The initialization automaton



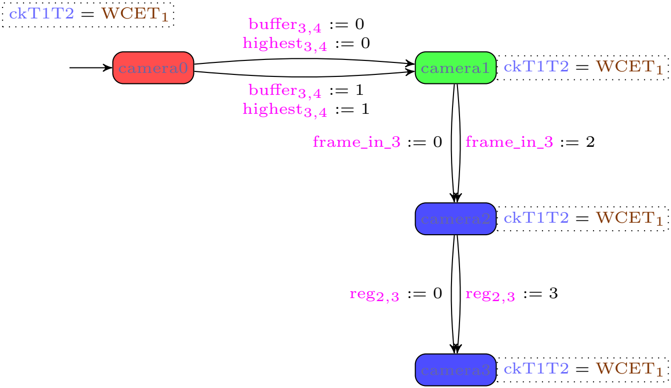
The initialization automaton



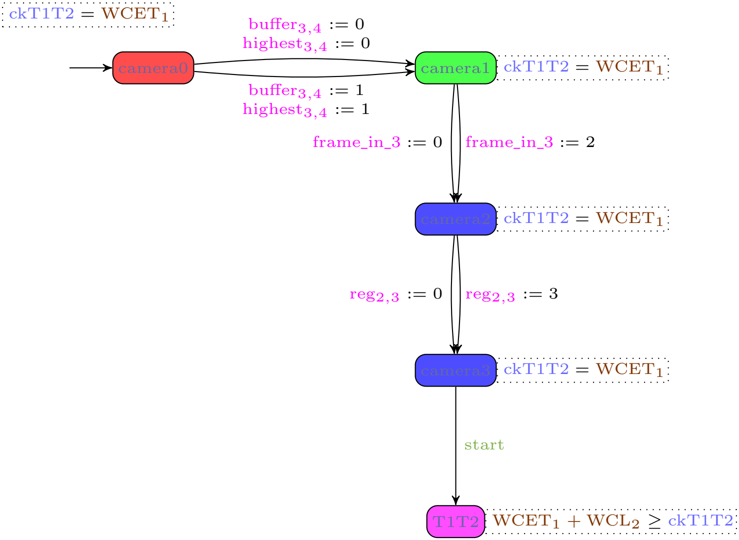
The initialization automaton



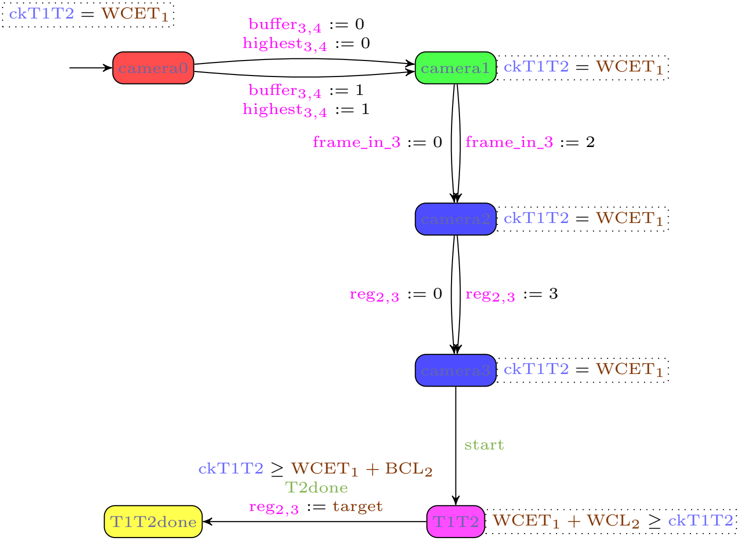
The initialization automaton



The initialization automaton

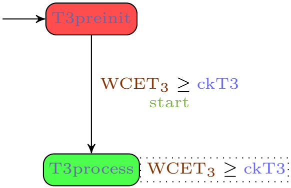


The initialization automaton

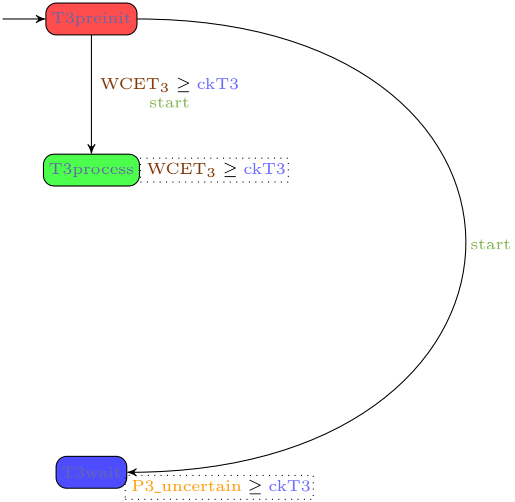


Task T3

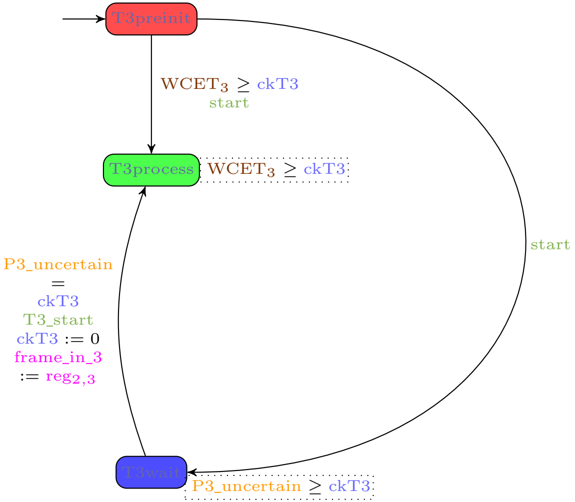
Task T3



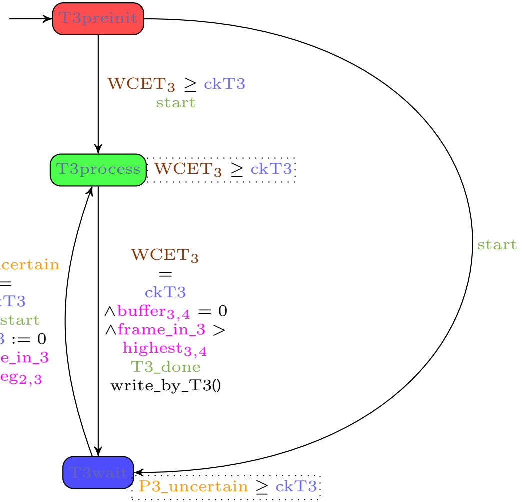
Task T3


T3preinit
WCET$_3$ ≥ ckT3
start
T3process
WCET$_3$ ≥ ckT3
start
T3wait
P3_uncertain ≥ ckT3

Task T3

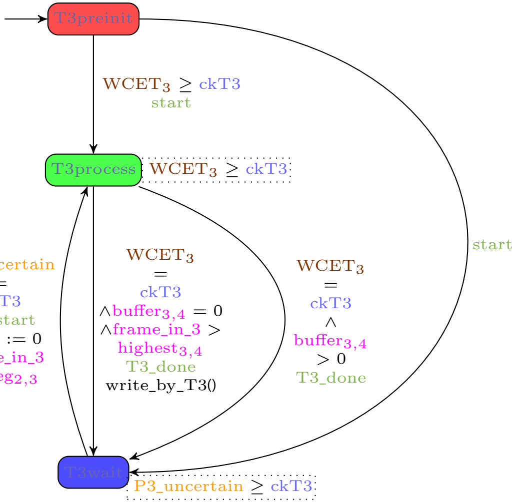

T3preinit
WCET₃ ≥ ckT3
start
T3process
WCET₃ ≥ ckT3
start
P3_uncertain
=
ckT3
T3_start
ckT3 := 0
frame_in_3
:= reg2,3
T3wait
P3_uncertain ≥ ckT3

Task T3

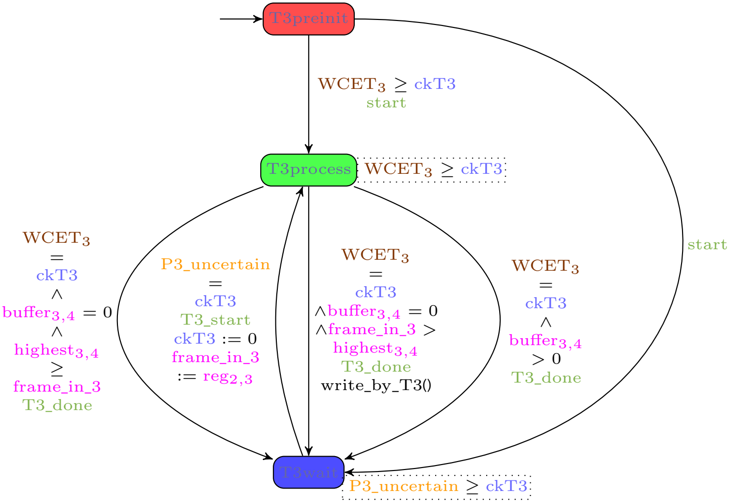


T3preinit
WCET$_3$ ≥ ckT3
start
T3process
WCET$_3$ ≥ ckT3
start
P3_uncertain
=
ckT3
T3_start
ckT3 := 0
frame_in_3
:= reg$_{2,3}$
WCET$_3$
=
ckT3
∧buffer$_{3,4}$ = 0
∧frame_in_3 >
highest$_{3,4}$
T3_done
write_by_T3()
T3wait
P3_uncertain ≥ ckT3

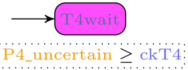
Task T3


T3preinit
WCET$_3$ ≥ ckT3
start
start
T3process
WCET$_3$ ≥ ckT3
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

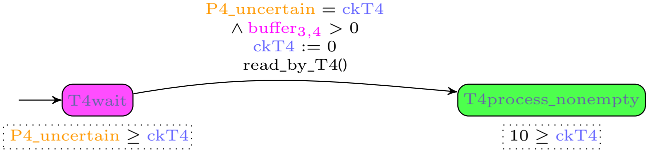
Task T3


T3preinit
WCET3 ≥ ckT3
start
T3process
WCET3 ≥ ckT3
WCET3 = ckT3 ∧ buffer3,4 = 0 ∧ highest3,4 ≥ frame_in_3
T3_done
P3_uncertain = ckT3
T3_start
ckT3 := 0
frame_in_3 := reg2,3
WCET3 = ckT3 ∧buffer3,4 = 0 ∧frame_in_3 > highest3,4
T3_done
write_by_T3()
WCET3 = ckT3 ∧ buffer3,4 > 0
T3_done
start
T3wait
P3_uncertain ≥ ckT3

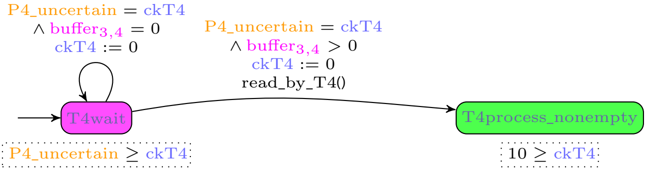
Task T4



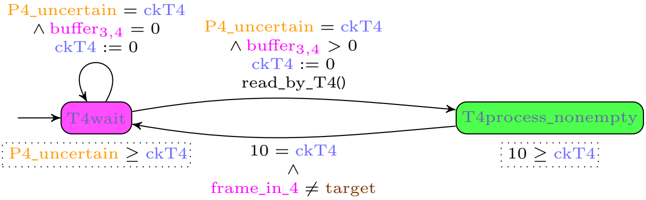
Task T4


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

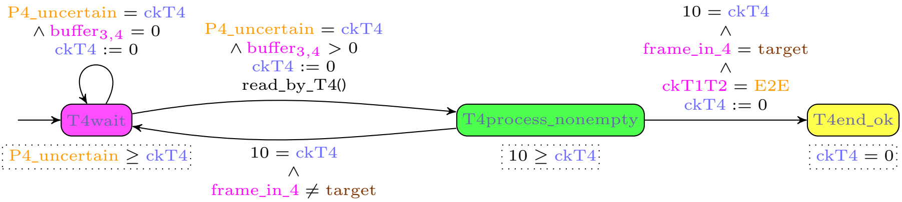
Task T4


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
∧ buffer3,4 = 0
ckT4 := 0
P4_uncertain = ckT4
∧ buffer3,4 > 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

[ÉA, Lipari, Sun @ WATERS'15]

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

- Distributed parameter synthesis
    - Distribution over a cluster [ÉA, Coti, Nguyen @ ICFEM 2015]
    - Multi-core synthesis [Laarman et al., 2013]
    - Distributed synthesis based on locations [Zhang et al., 2016]
- Machine learning [Li et al., 2017]

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
  - Communication protocols, failures...
- Another way to model systems known with limited precision

Security

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).
A unified formalism for monoprocessor schedulability analysis under uncertainty.
In Cavalcanti, A., Petrucci, L., and Seceleanu, C., editors, *FMICS-AVoCS*, volume 10471 of *Lecture Notes in Computer Science*, pages 100–115. Springer.
Best paper award.

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

André, É., Coti, C., and Nguyen, H. G. (2015a).
Enhanced distributed behavioral cartography of parametric timed automata.
In *ICFEM*, LNCS. Springer.

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., Hasuo, I., and Waga, M. (2018).
Offline timed pattern matching under uncertainty.
In *ICECCS*. IEEE.
To appear.

References III

André, É., Lipari, G., and Sun, Y. (2015b).
Verification of two real-time systems using parametric timed automata.
In Quinton, S. and Vardanega, T., editors, *WATERS*.

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

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

Fersman, E., Krcál, P., Pettersson, P., and Yi, W. (2007).
Task automata: Schedulability, decidability and undecidability.
Information and Computation, 205(8):1149–1172.

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.

References IV

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.

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.

Li, J., Sun, J., Gao, B., and André, É. (2017).
Classification based parameter synthesis for parametric timed automata.
In Duan, Z. and Ong, L., editors, *ICFEM*, volume 10610 of *Lecture Notes in Computer Science*, pages 243–261. Springer.

Luthmann, L., Stephan, A., Bürdek, J., and Lochau, M. (2017).
Modeling and testing product lines with unbounded parametric real-time constraints.
In Cohen, M. B., Acher, M., Fuentes, L., Schall, D., Bosch, J., Capilla, R., Bagheri, E., Xiong, Y., Troya, J., Cortés, A. R., and Benavides, D., editors, *SPLC, Volume A*, pages 104–113. ACM.

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

References V

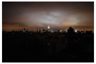
Norström, C., Wall, A., and Yi, W. (1999).
Timed automata as task models for event-driven systems.
In *RTCSA*, pages 182–189. IEEE Computer Society.

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.

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)

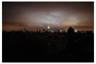
Error screen on the earliest versions of Macintosh

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

License of this document