AURA 2018

30th October 2018
Chennai, India

Monitoring cyber-physical systems under uncertainty

Étienne André, Ichiro Hasuo and Masaki Waga

LIPN, Université Paris 13, CNRS, France
National Institute of Informatics, Japan
JFLI, UMI CNRS, Tokyo, Japan

Supported by JST ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603) and the ANR national research program PACS (ANR-14-CE28-0002).

Motivation: automotive industry

- Modern cars embed several processors and produce logs

Motivation: automotive industry

- Modern cars embed several processors and produce logs

- Log: sequences of events and timestamps

```
start    2.3
gear1    5.8
gear2    9.2
gear3   18.5
gear2   42.1
```

Motivation: automotive industry

- Modern cars embed several processors and produce logs

  - Log: sequences of events and timestamps

```
start    2.3
gear1    5.8
gear2    9.2
gear3   18.5
gear2   42.1
```

- How to ensure on-the-fly that some properties are satisfied on a log?
  - “It never happens that gear1 and gear3 are separated by less than 5 s”

Motivation: automotive industry

- Modern cars embed several processors and produce logs

  - Log: sequences of events and timestamps

```
start    2.3
gear1    5.8
gear2    9.2
gear3   18.5
gear2   42.1
```

- How to ensure on-the-fly that some properties are satisfied on a log?
  - “It never happens that gear1 and gear3 are separated by less than 5 s”

⇒ Monitoring

Larger motivation: data collection and management

- Personal mobile devices collect large amounts of data

Larger motivation: data collection and management

- Personal mobile devices collect large amounts of data

These data can also come in the form of a timed log

```
start walking                                   2.3
```

Larger motivation: data collection and management

- Personal mobile devices collect large amounts of data

These data can also come in the form of a timed log

```
start walking                                   2.3
walk faster                                     6.3
```

Larger motivation: data collection and management

- Personal mobile devices collect large amounts of data

These data can also come in the form of a timed log

```
start walking          2.3
walk faster            6.3
receive SMS           15.8
```

Larger motivation: data collection and management

- Personal mobile devices collect large amounts of data

These data can also come in the form of a timed log

```
start walking                          2.3
walk faster                            6.3
receive SMS                           15.8
read SMS                              19.2
```

Larger motivation: data collection and management

- Personal mobile devices collect large amounts of data

These data can also come in the form of a timed log

```
start walking                          2.3
walk faster                            6.3
receive SMS                           15.8
read SMS                              19.2
sound of someone bumping into a lamp  22.5
```

Larger motivation: data collection and management

- Personal mobile devices collect large amounts of data

These data can also come in the form of a timed log

```
start walking                        2.3
walk faster                          6.3
receive SMS                         15.8
read SMS                            19.2
sound of someone bumping into a lamp 22.5
```

- Key challenge: manage these data
  - Verify properties: "has the owner bumped into a street lamp"?
    - key applications (health, ...)
  - Deduce information:
    - "what are the minimum/maximum intervals without visiting this shop"?
    - "is the user visiting this place more or less periodically?" (without knowing the actual period)

Outline

1. Pattern matching
2. Methodology
3. Experiments
4. Perspectives

Untimed pattern matching: example

- Naive algorithm for pattern matching

$\texttt{c}\quad\texttt{r}\quad\texttt{e}\quad\texttt{p}\quad\texttt{e}\quad\texttt{s}\quad \in ?L\Big(\{c|i|d\}^{?}r^{*}e\Big)$

Untimed pattern matching: example

- Naive algorithm for pattern matching

c r e p e s $\in ? L\left(\{c|i|d\}^{?} r^{*} e\right)$

c

Untimed pattern matching: example

- Naive algorithm for pattern matching

c	r	e	p	e	s	$\in ? L\Big(\{c \mid i \mid d\}^{?} r^{*} e\Big)$
c	r					

Untimed pattern matching: example

- Naive algorithm for pattern matching

c	r	e	p	e	s	$\in ? L\left(\{c \mid i \mid d\}^{?} r^{*} e\right)$
c	r	e				

Untimed pattern matching: example

- Naive algorithm for pattern matching

c	r	e	p	e	s	$\in ? L\Big(\{c \mid i \mid d\}^{?} r^{*} e\Big)$
c	r	e				✓

Untimed pattern matching: example

- Naive algorithm for pattern matching

c	r	e	p	e	s	$\in ? L\left(\{c \mid i \mid d\}^{?} r^{*} e\right)$
c	r	e				✓
	r					

Untimed pattern matching: example

- Naive algorithm for pattern matching

c	r	e	p	e	s	$\in ? L\Big(\{c\vert i\vert d\}^{?} r^{*} e\Big)$
c	r	e				✓
	r	e				

Untimed pattern matching: example

- Naive algorithm for pattern matching

c	r	e	p	e	s	$\in ? L\left(\{c \vert i \vert d\}^? r^* e\right)$
c	r	e				$\surd$
	r	e				$\surd$

Untimed pattern matching: example

- Naive algorithm for pattern matching

c	r	e	p	e	s	$\in ? L\big(\{c\vert i\vert d\}^{?} r^{*} e\big)$
c	r	e				✓
	r	e				✓
		e				

Untimed pattern matching: example

- Naive algorithm for pattern matching

c	r	e	p	e	s	$\in ?L\Big(\{c\|i\|d\}^{?}r^{*}e\Big)$
c	r	e				$\surd$
	r	e				$\surd$
		e				$\surd$

Untimed pattern matching: example

- Naive algorithm for pattern matching

c	r	e	p	e	s	$\in ? L\Big(\{c\|i\|d\}^{?} r^{*} e\Big)$
c	r	e				$\surd$
	r	e				$\surd$
		e				$\surd$
			p			

Untimed pattern matching: example

- Naive algorithm for pattern matching

c	r	e	p	e	s	$\in ? L\left(\{c \mid i \mid d\}^{?} r^{*} e\right)$
c	r	e				✓
	r	e				✓
		e				✓
			p			×

Untimed pattern matching: example

- Naive algorithm for pattern matching

c	r	e	p	e	s	$\in ? L\left(\{c \mid i \mid d\}^{?} r^{*} e\right)$
c	r	e				✓
	r	e				✓
		e				✓
			p			×
				e		

Untimed pattern matching: example

- Naive algorithm for pattern matching

c	r	e	p	e	s	$\in ? L\big(\{c \mid i \mid d\}^? r^* e\big)$
c	r	e				✓
	r	e				✓
		e				✓
			p			×
				e		✓

Untimed pattern matching: example

- Naive algorithm for pattern matching

c	r	e	p	e	s	$\in ? L\big(\{c \mid i \mid d\}^? r^* e\big)$
c	r	e				✓
	r	e				✓
		e				✓
			p			×
				e		✓
					s	

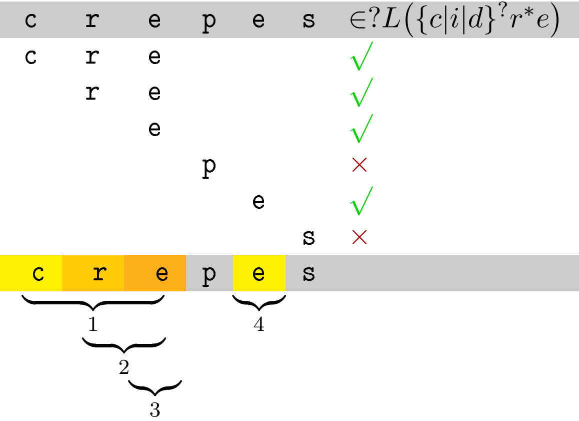
Untimed pattern matching: example

- Naive algorithm for pattern matching

c	r	e	p	e	s	$\in ? L\Big(\{c \mid i \mid d\}^{?} r^{*} e\Big)$
c	r	e				✓
	r	e				✓
		e				✓
			p			×
				e		✓
					s	×

Untimed pattern matching: example

- Naive algorithm for pattern matching



c	r	e	p	e	s	$\in ?L\left(\{c\|i\|d\}^{?} r^{*} e\right)$
c	r	e				✓
	r	e				✓
		e				✓
			p			×
				e		✓
					s	×
c	r	e	p	e	s	

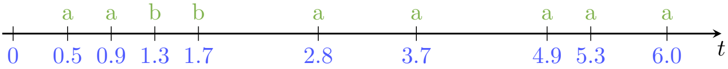
Timed pattern matching: timed word

Timed word

[Alur and Dill, 1994]

=

sequence of actions and timestamps



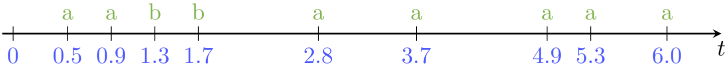
Timed pattern matching: timed word

Timed word

[Alur and Dill, 1994]

=

sequence of actions and timestamps

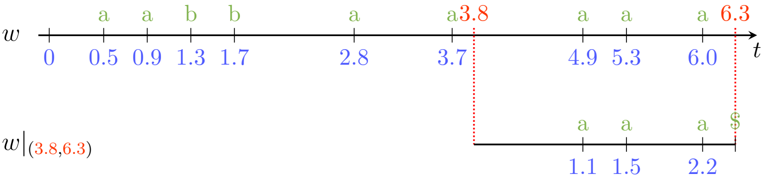


Timed word segment

[Waga et al., 2016]

=

projection of a segment of the timed word onto a given interval



Timed pattern matching: timed automaton

How to express a (timed) property on a log?

Example

"At least 1 time unit after the start of the segment, a is observed.
Then, within strictly less than 1 time unit, another a is observed.
Then, within strictly less than 1 time unit, another a is observed."

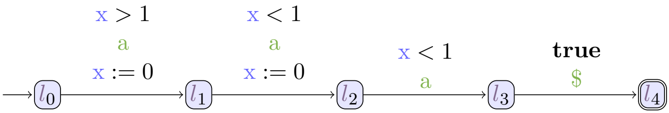
Timed pattern matching: timed automaton

How to express a (timed) property on a log?

Example

“At least 1 time unit after the start of the segment, a is observed.
Then, within strictly less than 1 time unit, another a is observed.
Then, within strictly less than 1 time unit, another a is observed.”

A solution: timed automata [Alur and Dill, 1994]



- expressive
- well-studied
- supported by well-established model-checkers

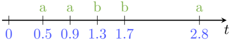
Timed pattern matching: principle

Timed pattern matching

- Inputs

A log

(timed word)

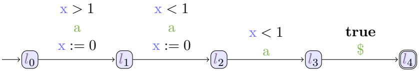


A property

usually a specification of faults

(timed automaton)

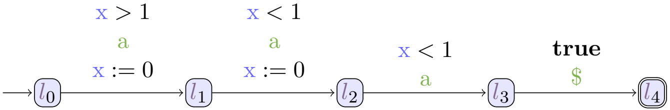
[Alur and Dill, 1994]



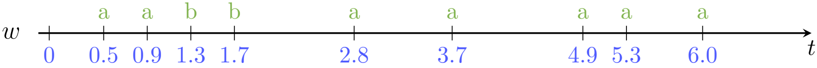
- Output
  - The set of time intervals where faults are detected
    ⇒ Set of matching intervals $\{(t, t') \mid w|_{(t,t')} \in \mathcal{L}(\mathcal{A})\}$

Timed pattern matching: example

Our property:

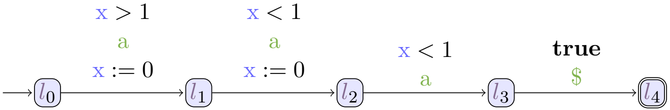


Our log:

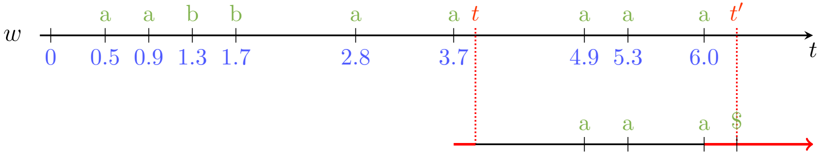


Timed pattern matching: example

Our property:

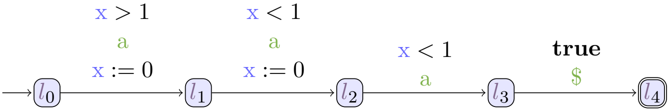


Our log:

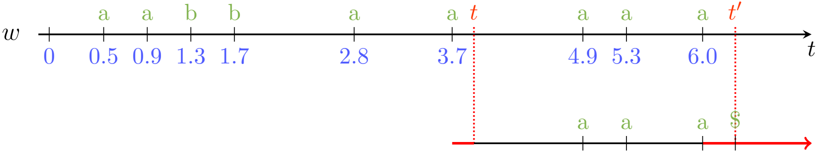


Timed pattern matching: example

Our property:



Our log:



Set of matching intervals:

$$\{(t,t') \mid w|_{(t,t')} \in \mathcal{L}(\mathcal{A})\} = \{(t,t') \mid t \in (3.7, 3.9), t' \in [6.0, \infty)\}$$

Previous works

- Timed pattern matching with signals

  [Ulus et al., 2014, Ulus et al., 2016, Ulus, 2017]

  - logs are encoded by signals (i. e., values that vary over time)
    - *state-based* view, while our timed words are *event-based*
  - specification is encoded by timed regular expressions (TREs)

- Timed pattern matching with timed words and timed automata

  [Waga et al., 2016, Waga et al., 2017]

  - [Waga et al., 2016]: brute-force and Boyer-Moore algorithm
  - [Waga et al., 2017]: online algorithm that employs skip values from the Franek–Jennings–Smyth string matching algorithm [Franek et al., 2007]

Goal: Extend timed pattern matching for uncertainty

Challenges

- The property may not be known with full certainty:
  - Detect a periodic event but without knowing the period
    - “is the user visiting this place more or less periodically?” (without knowing the actual period)
- Optimization problems
  - Find minimal/maximal timings for which some property holds
    - “what are the minimum/maximum intervals without visiting this shop”?

Goal: Extend timed pattern matching for uncertainty

Challenges

- The property may not be known with full certainty:
  - Detect a periodic event but without knowing the period
    - “is the user visiting this place more or less periodically?” (without knowing the actual period)
- Optimization problems
  - Find minimal/maximal timings for which some property holds
    - “what are the minimum/maximum intervals without visiting this shop”?

Objective

Find intervals of time and values of parameters for which a property holds

Outline

1 Pattern matching

2 Methodology

3 Experiments

4 Perspectives

Methodology

Main idea

Use parametric timed model checking

- parametric timed automata [Alur et al., 1993]
- parameter synthesis
- IMITATOR [André et al., 2012]

Methodology

Main idea

Use parametric timed model checking

- parametric timed automata [Alur et al., 1993]
- parameter synthesis
- IMITATOR [André et al., 2012]

Methodology step by step

1. Encode the property using a PTA
2. Add two parameters t and t'
3. Apply a (mild) transformation to the property a PTA
4. Transform the timed word into a PTA
5. Perform the composition of both PTA
6. Apply reachability synthesis to the product

Outline

1 Pattern matching

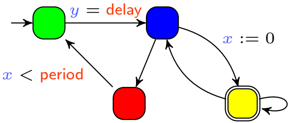
2 Methodology

- Parametric timed automata

3 Experiments

4 Perspectives

timed model checking



A model of the system

$\overset{?}{\models}$

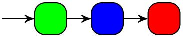
🟥 is unreachable

A property to be satisfied

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

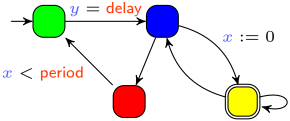
Yes

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...

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

Property: parametric timed automaton

Expressing a parametric timed property on a log

Example

“At least p_1 time units after the start of the segment, a is observed.
Then, within strictly less than p_2 time units, another a is observed.
Then, within strictly less than p_2 time units, another a is observed.”

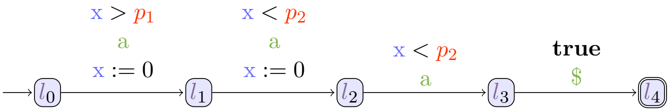
Property: parametric timed automaton

Expressing a parametric timed property on a log

Example

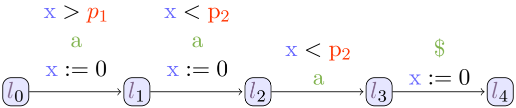
“At least p_1 time units after the start of the segment, a is observed.
Then, within strictly less than p_2 time units, another a is observed.
Then, within strictly less than p_2 time units, another a is observed.”

Our solution: parametric timed automata [Alur et al., 1993]



Modifying the property pattern

Add some start and end gadgets for completeness of the method

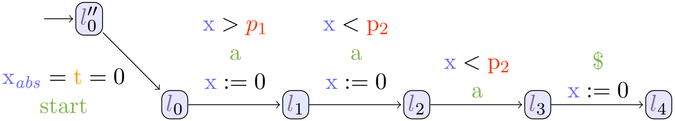


See manuscript for formal transformation and proofs

Modifying the property pattern

Add some start and end gadgets for completeness of the method

1. Add an initial transition in 0-time
   - Captures segments starting from 0

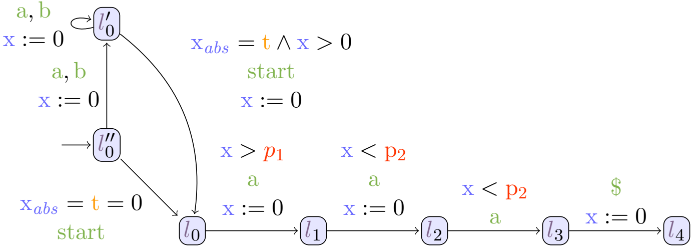


See manuscript for formal transformation and proofs

Modifying the property pattern

Add some start and end gadgets for completeness of the method

1. Add an initial transition in 0-time
   - Captures segments starting from 0
2. Add a new location with a self-loop
   - Captures segments not starting from the beginning of the word

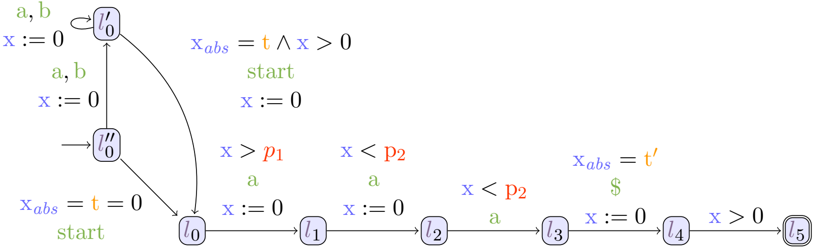


See manuscript for formal transformation and proofs

Modifying the property pattern

Add some start and end gadgets for completeness of the method

1. Add an initial transition in 0-time
   - Captures segments starting from 0
2. Add a new location with a self-loop
   - Captures segments not starting from the beginning of the word
3. Add a new final transition in > 0 time
   - To match the usual definition that the segment must end in > 0 time after the last action



See manuscript for formal transformation and proofs

Transforming a log into a (parametric) timed automaton

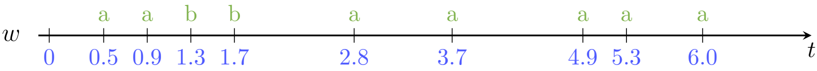
Essentially easy:

1. Add one clock never reset (absolute time)
2. Convert pairs (action, time) into transitions

Transforming a log into a (parametric) timed automaton

Essentially easy:

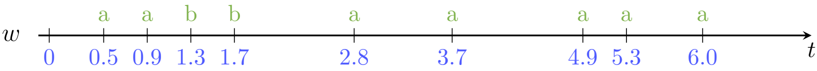
1. Add one clock never reset (absolute time)
2. Convert pairs (action, time) into transitions



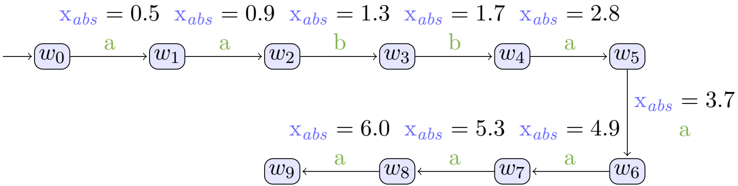
Transforming a log into a (parametric) timed automaton

Essentially easy:

1. Add one clock never reset (absolute time)
2. Convert pairs (action, time) into transitions



$$\Downarrow$$



Product and reachability synthesis

Result

The set of parameter valuations $\mathrm{t}, \mathrm{t}', p_1, p_2 \ldots$ reaching the final location of the property is exactly the answer to the parametric pattern matching problem

Product and reachability synthesis

Result

The set of parameter valuations $\mathrm{t}, \mathrm{t}', p_1, p_2 \ldots$ reaching the final location of the property is exactly the answer to the parametric pattern matching problem

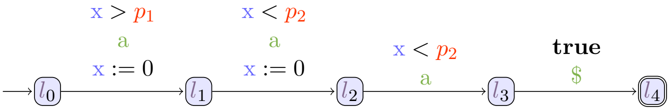
Remark

This problem is decidable… in contrast to most problems using PTAs!

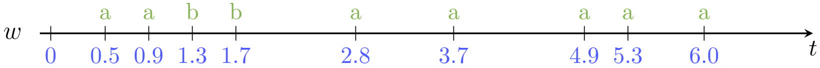
[André, 2018]

Product and reachability synthesis: example

Our property:

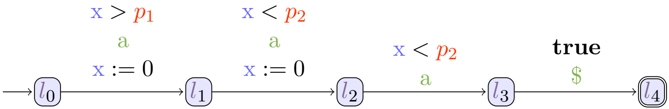


Our log:

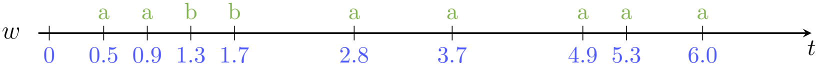


Product and reachability synthesis: example

Our property:



Our log:

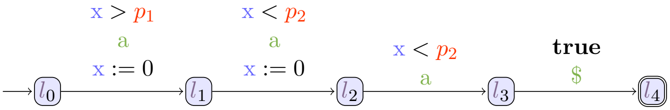


Set of matching intervals:

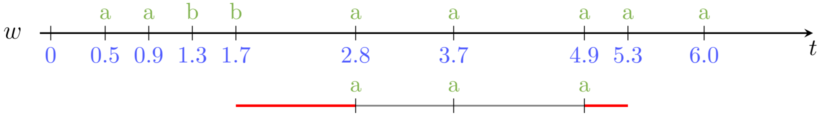
$$
\begin{aligned}
& 1.7 < t < 2.8 - p_1 \wedge 4.9 \leq t' < 5.3 \wedge p_2 > 1.2 \\
\vee\ & 2.8 < t < 3.7 - p_1 \wedge 5.3 \leq t' < 6 \wedge p_2 > 1.2 \\
\vee\ & 3.7 < t < 4.9 - p_1 \wedge t' \geq 6 \wedge p_2 > 0.7
\end{aligned}
$$

Product and reachability synthesis: example

Our property:



Our log:

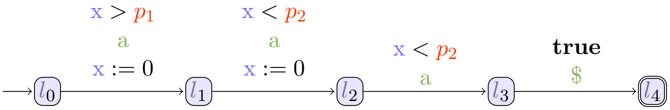


Set of matching intervals:

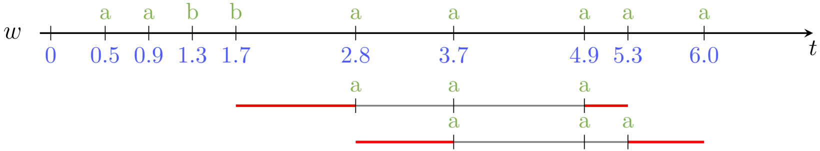
$$
\begin{aligned}
& 1.7 < t < 2.8 - p_1 \land 4.9 \leq t' < 5.3 \land p_2 > 1.2 \\
\lor\ & 2.8 < t < 3.7 - p_1 \land 5.3 \leq t' < 6 \land p_2 > 1.2 \\
\lor\ & 3.7 < t < 4.9 - p_1 \land t' \geq 6 \land p_2 > 0.7
\end{aligned}
$$

Product and reachability synthesis: example

Our property:



Our log:

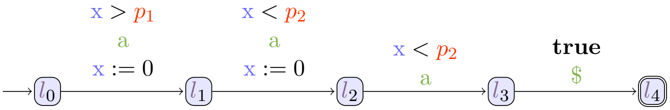


Set of matching intervals:

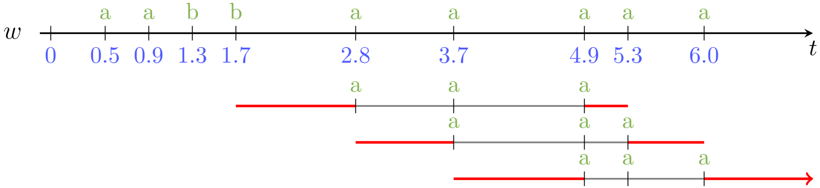
$$
\begin{aligned}
& 1.7 < t < 2.8 - p_1 \wedge 4.9 \leq t' < 5.3 \wedge p_2 > 1.2 \\
\vee\ & 2.8 < t < 3.7 - p_1 \wedge 5.3 \leq t' < 6 \wedge p_2 > 1.2 \\
\vee\ & 3.7 < t < 4.9 - p_1 \wedge t' \geq 6 \wedge p_2 > 0.7
\end{aligned}
$$

Product and reachability synthesis: example

Our property:



Our log:



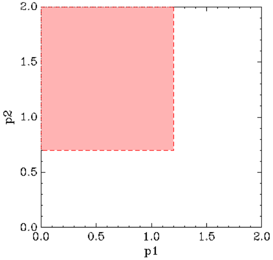
Set of matching intervals:

$$
\begin{aligned}
& 1.7 < t < 2.8 - p_1 \wedge 4.9 \leq t' < 5.3 \wedge p_2 > 1.2 \\
\vee\ & 2.8 < t < 3.7 - p_1 \wedge 5.3 \leq t' < 6 \wedge p_2 > 1.2 \\
\vee\ & 3.7 < t < 4.9 - p_1 \wedge t' \geq 6 \wedge p_2 > 0.7
\end{aligned}
$$

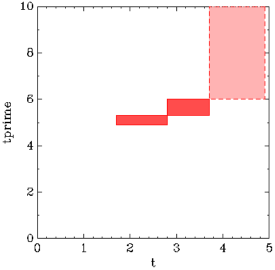
Exemple: graphical representation

$$\begin{aligned} & 1.7 < \mathrm{t} < 2.8 - \mathrm{p}_1 \land 4.9 \leq \mathrm{t}' < 5.3 \land \mathrm{p}_2 > 1.2 \\ \lor\ & 2.8 < \mathrm{t} < 3.7 - \mathrm{p}_1 \land 5.3 \leq \mathrm{t}' < 6 \land \mathrm{p}_2 > 1.2 \\ \lor\ & 3.7 < \mathrm{t} < 4.9 - \mathrm{p}_1 \land \mathrm{t}' \geq 6 \land \mathrm{p}_2 > 0.7 \end{aligned}$$

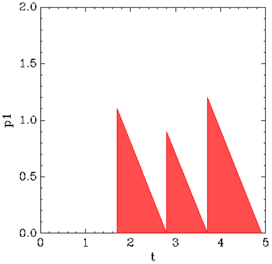
Projections in 2 dimensions:



On p_1 and p_2



On t and t'



On t and p_1

Outline

1 Pattern matching

2 Methodology

3 Experiments

4 Perspectives

Outline

1 Pattern matching

2 Methodology

3 Experiments

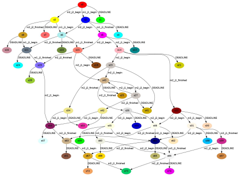
- IMITATOR in a nutshell
- Benchmarks

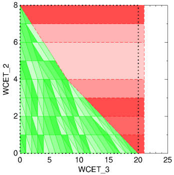
4 Perspectives

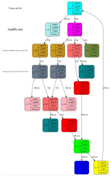
IMITATOR

- 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







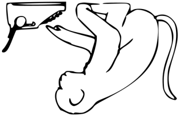
IMITATOR

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



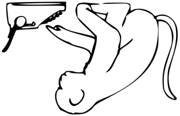
IMITATOR

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]

- Formal timing analysis of music scores [Fanchon and Jacquemard, 2013]

- Solution to a challenge related to a distributed video processing system by Thales

- Offline monitoring

Outline

1 Pattern matching

2 Methodology

3 Experiments

- IMITATOR in a nutshell
- Benchmarks

4 Perspectives

Experimental environment

Toolkit

- Simple Python script to transform timed words into IMITATOR PTAs
- Slightly modified version of IMITATOR
  - To handle PTAs with dozens of thousands of locations
  - To manage n-parameter constraints with dozens of thousands of disjuncts

Sources, binaries, models, logs can be found at `www.imitator.fr/static/ICECCS18`

Case study 1: GEAR (description)

Monitoring the gear change of an automatic transmission system

- Obtained by simulation of the Simulink model of an automatic transmission system [Hoxha et al., 2014]
- S-TaLiRo [Annpureddy et al., 2011] used to generate an input to this model (generates a gear change signal that is fed to the model)
- Gear chosen from $\{g_1, g_2, g_3, g_4\}$
- Generated gear change recorded in a timed word

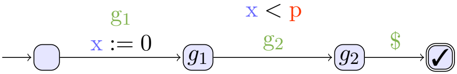
Property

"If the gear is changed to 1, it should not be changed to 2 within p seconds."

This condition is related to the requirement ϕ_5^{AT} proposed in [Hoxha et al., 2014] (the nominal value for p in [Hoxha et al., 2014] is 2).

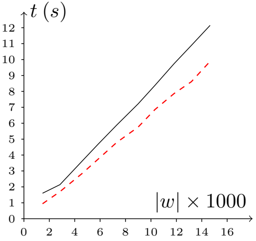
Case study 1: GEAR (experiments)

Property: “If the gear is changed to 1, it should not be changed to 2 within p seconds.”



Experiments data:

Model		PTPM				PTPM$_{opt}$	
Length	**Time frame**	**States**	**Matches**	**Parsing (s)**	**Comp. (s)**	**States**	**Comp. (s)**
1,467	1,000	4,453	379	0.02	1.60	3,322	0.94
2,837	2,000	8,633	739	0.33	2.14	6,422	1.70
4,595	3,000	14,181	1,247	0.77	3.63	10,448	2.85
5,839	4,000	17,865	1,546	1.23	4.68	13,233	3.74
7,301	5,000	22,501	1,974	1.94	5.88	16,585	4.79
8,995	6,000	27,609	2,404	2.96	7.28	20,413	5.76
10,316	7,000	31,753	2,780	4.00	8.38	23,419	6.86
11,831	8,000	36,301	3,159	5.39	9.75	26,832	7.87
13,183	9,000	40,025	3,414	6.86	10.89	29,791	8.61
14,657	10,000	44,581	3,816	8.70	12.15	33,141	9.89



PTPM$_{opt}$: alternative procedure to find the minimum/maximum value of a parameter along the log

Case study 2: ACCEL (description)

Monitoring the acceleration of an automated transmission system

- Also obtained by simulation from the Simulink model of [Hoxha et al., 2014]
- (discretized) value of three state variables recorded in the log:
  - engine RPM (discretized to “high” and “low” with a certain threshold)
  - velocity (discretized to “high” and “low” with a certain threshold)
  - 4 gear positions

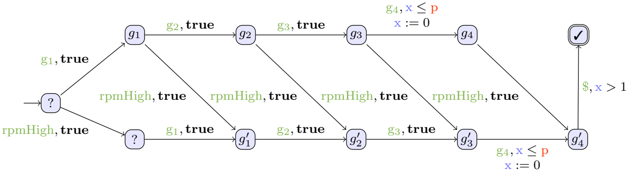
Property

“If a gear changes from 1 to 2, 3, and 4 in this order in p seconds and engine RPM becomes large during this gear change, then the velocity of the car must be sufficiently large in one second.”

This condition models the requirement ϕ_8^{AT} proposed in [Hoxha et al., 2014] (the nominal value for p in [Hoxha et al., 2014] is 10).

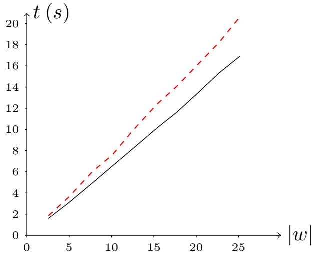
Case study 2: ACCEL (experiments)

Property: “If a gear changes from 1 to 2, 3, and 4 in this order in p seconds and engine RPM becomes large during this gear change, then the velocity of the car must be sufficiently large in one second.”



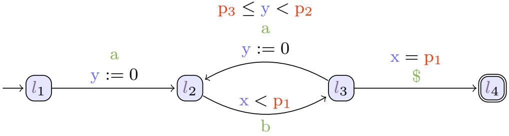
Experiments data:

Model		PTPM				PTPM$_{\text{opt}}$	
Length	**Time frame**	**States**	**Matches**	**Parsing (s)**	**Comp. (s)**	**States**	**Comp. (s)**
2,559	1,000	6,504	2	0.27	1.60	6,502	1.85
4,894	2,000	12,429	2	0.86	3.04	12,426	3.57
7,799	3,000	19,922	7	2.21	4.98	19,908	6.06
10,045	4,000	25,520	3	3.74	6.51	25,514	7.55
12,531	5,000	31,951	9	6.01	8.19	31,926	9.91
15,375	6,000	39,152	7	9.68	10.14	39,129	12.39
17,688	7,000	45,065	9	13.40	11.61	45,039	14.06
20,299	8,000	51,660	10	18.45	13.52	51,629	16.23
22,691	9,000	57,534	11	24.33	15.33	57,506	18.21
25,137	10,000	63,773	13	31.35	16.90	63,739	20.61



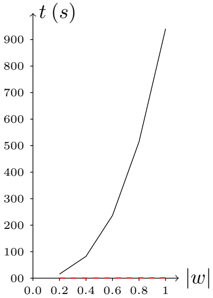
Case study 3: Blowup

Property made on purpose to test our scalability



Experiments data:

Model		PTPM				PTPM$_{\text{opt}}$	
Length	**Time frame**	**States**	**Matches**	**Parsing (s)**	**Comp. (s)**	**States**	**Comp. (s)**
200	101	20,602	5,050	0.01	15.31	515	0.24
400	202	81,202	20,100	0.02	82.19	1,015	0.49
600	301	181,802	45,150	0.03	236.80	1,515	0.71
800	405	322,402	80,200	0.05	514.57	2,015	1.05
1,000	503	503,002	125,250	0.06	940.74	2,515	1.24



Outline

1 Pattern matching

2 Methodology

3 Experiments

4 Perspectives

Summary

- New method to monitor logs of real-time systems

- Methodology: parametric timed model checking

- Applications: automotive industry
  - Linear in the size of the log
  - Able to handle logs of dozens of thousands of events

Summary

- New method to monitor logs of real-time systems

- Methodology: parametric timed model checking

- Applications: automotive industry
  - Linear in the size of the log
  - Able to handle logs of dozens of thousands of events

- Article Offline timed pattern matching under uncertainty by André, Hasuo and Waga to be presented at ICECCS 2018 (December)

- An ~~offline~~ online algorithm
  - We believe our algorithm is in fact essentially online
    - No need for the whole log to start the analysis
    - The word could be fed to IMITATOR in an incremental manner
  - But the speed may need to be improved further

Perspectives

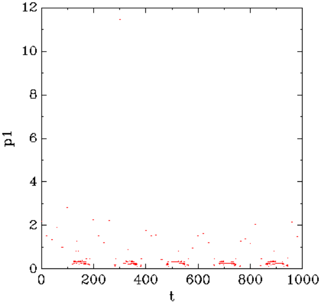
- Extensions
  - Improve the efficiency with skipping [Waga et al., 2017]
  - Exploit the polarity of parameters [Asarin et al., 2011]
  - Use and extend the `MONAA` library [Waga et al., 2018]

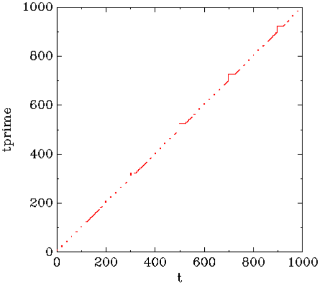
- Graphical representation and interpretation
  - How to interpret dozens of thousands of matches?

Perspectives

- Extensions
  - Improve the efficiency with skipping [Waga et al., 2017]
  - Exploit the polarity of parameters [Asarin et al., 2011]
  - Use and extend the `MONAA` library [Waga et al., 2018]

- Graphical representation and interpretation
  - How to interpret dozens of thousands of matches?





Bibliography

References I

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

Alur, R., Henzinger, T. A., and Vardi, M. Y. (1993).
Parametric real-time reasoning.
In Kosaraju, S. R., Johnson, D. S., and Aggarwal, A., editors, *STOC*, pages 592–601, New York, NY, USA. ACM.

André, É. (2018).
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 Giannakopoulou, D. and Méry, D., editors, *FM*, volume 7436 of *LNCS*, pages 33–36. Springer.

Annpureddy, Y., Liu, C., Fainekos, G. E., and Sankaranarayanan, S. (2011).
S-TaLiRo: A tool for temporal logic falsification for hybrid systems.
In Abdulla, P. A. and Leino, K. R. M., editors, *TACAS*, volume 6605 of *LNCS*, pages 254–257. Springer.

Asarin, E., Donzé, A., Maler, O., and Nickovic, D. (2011).
Parametric identification of temporal properties.
In *RV*, volume 7186 of *LNCS*, pages 147–160. Springer.

References II

Fanchon, L. and Jacquemard, F. (2013).
Formal timing analysis of mixed music scores.
In *ICMC*. Michigan Publishing.

Franek, F., Jennings, C. G., and Smyth, W. F. (2007).
A simple fast hybrid pattern-matching algorithm.
Journal of Discrete Algorithms, 5(4):682–695.

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

Hoxha, B., Abbas, H., and Fainekos, G. E. (2014).
Benchmarks for temporal logic requirements for automotive systems.
In Frehse, G. and Althoff, M., editors, *ARCH@CPSWeek*, volume 34 of *EPiC Series in Computing*, pages 25–30. EasyChair.

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.

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.

References III

Ulus, D. (2017).
Montre: A tool for monitoring timed regular expressions.
In Majumdar, R. and Kuncak, V., editors, *CAV, Part I*, volume 10426 of *LNCS*, pages 329–335. Springer.

Ulus, D., Ferrère, T., Asarin, E., and Maler, O. (2014).
Timed pattern matching.
In Legay, A. and Bozga, M., editors, *FORMATS*, volume 8711 of *LNCS*, pages 222–236. Springer.

Ulus, D., Ferrère, T., Asarin, E., and Maler, O. (2016).
Online timed pattern matching using derivatives.
In Chechik, M. and Raskin, J., editors, *TACAS*, volume 9636 of *LNCS*, pages 736–751. Springer.

Waga, M., Akazaki, T., and Hasuo, I. (2016).
A Boyer-Moore type algorithm for timed pattern matching.
In Fränzle, M. and Markey, N., editors, *FORMATS*, volume 9884 of *LNCS*, pages 121–139. Springer.

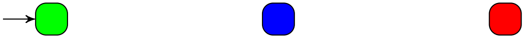
Waga, M., Hasuo, I., and Suenaga, K. (2017).
Efficient online timed pattern matching by automata-based skipping.
In Abate, A. and Geeraerts, G., editors, *FORMATS*, volume 10419 of *LNCS*, pages 224–243. Springer.

Waga, M., Hasuo, I., and Suenaga, K. (2018).
MONAA: A tool for timed pattern matching with automata-based acceleration.
In *MT@CPSWeek*, pages 14–15. IEEE.

Additional explanation

Timed automaton (TA)

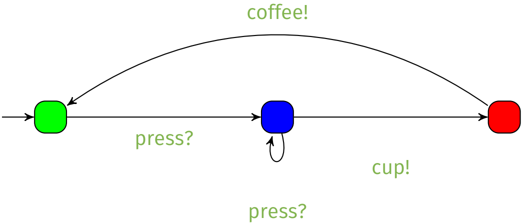
- Finite state automaton (sets of locations)



🟩 idle
🟦 adding sugar
🟥 delivering coffee

Timed automaton (TA)

- Finite state automaton (sets of locations and actions)



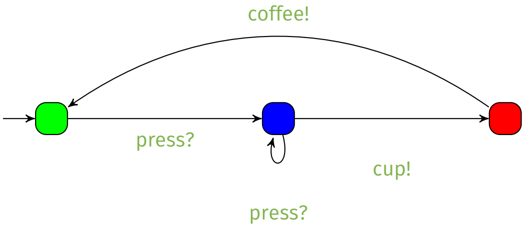
🟩 idle

🟦 adding sugar

🟥 delivering coffee

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



🟩 idle

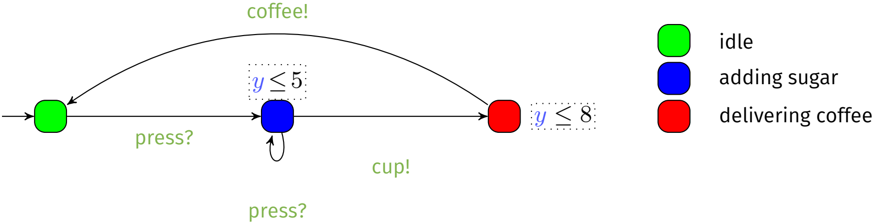
🟦 adding sugar

🟥 delivering coffee

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

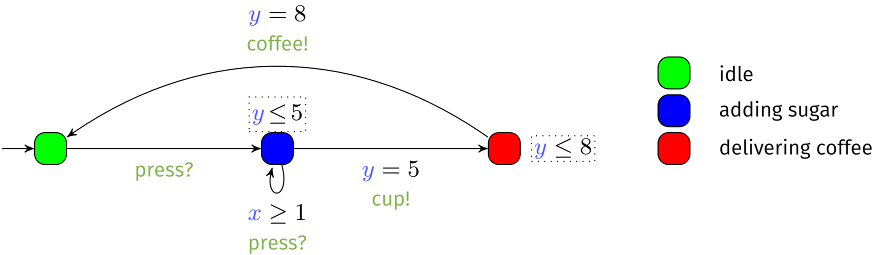
- Features
  - Location invariant: property to be verified to stay at a location



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

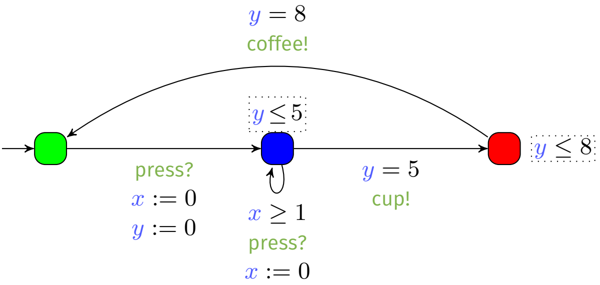
- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property to be verified to enable a transition



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



$y = 8$

coffee!

$y \leq 5$

$y \leq 8$

press?

$x := 0$

$y := 0$

$y = 5$

cup!

$x \geq 1$

press?

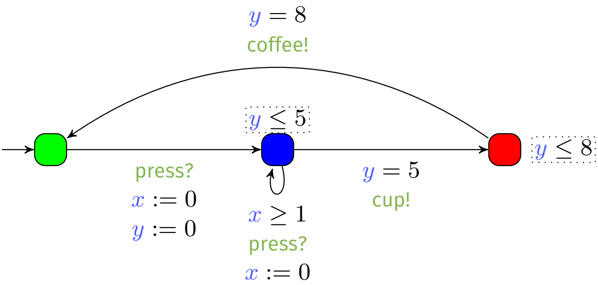
$x := 0$

🟩 idle

🟦 adding sugar

🟥 delivering coffee

The most critical system: The coffee machine

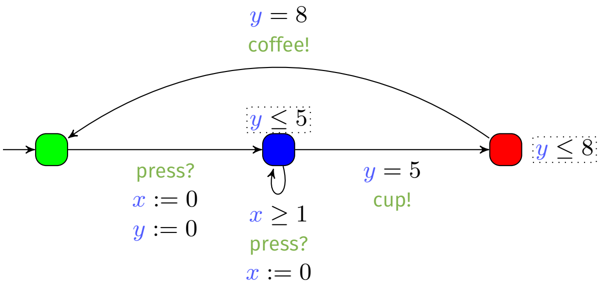


🟩 idle

🟦 adding sugar

🟥 delivering coffee

The most critical system: The coffee machine



🟩 idle

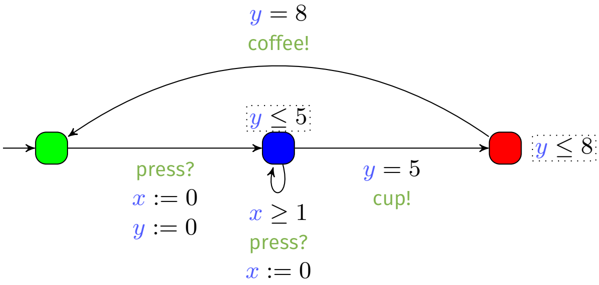
🟦 adding sugar

🟥 delivering coffee

- 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

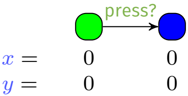


🟩 idle

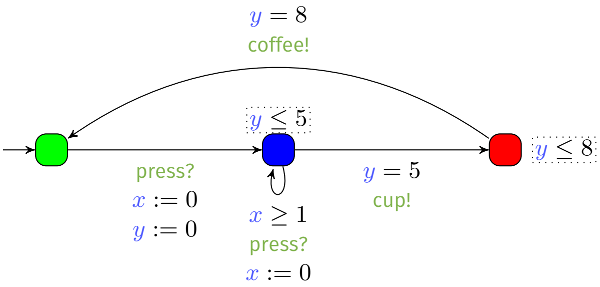
🟦 adding sugar

🟥 delivering coffee

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

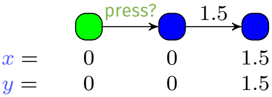


The most critical system: The coffee machine

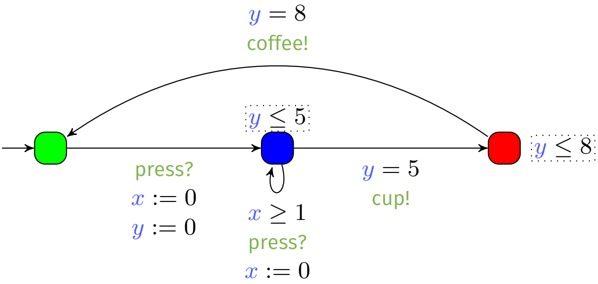


- 🟩 idle
- 🟦 adding sugar
- 🟥 delivering coffee

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



The most critical system: The coffee machine

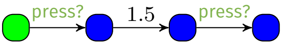


🟩 idle

🟦 adding sugar

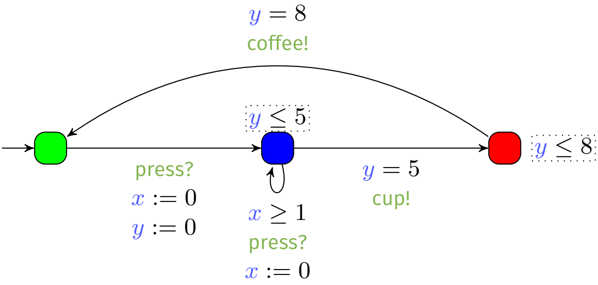
🟥 delivering coffee

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



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

The most critical system: The coffee machine

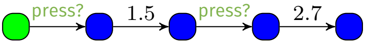


🟩 idle

🟦 adding sugar

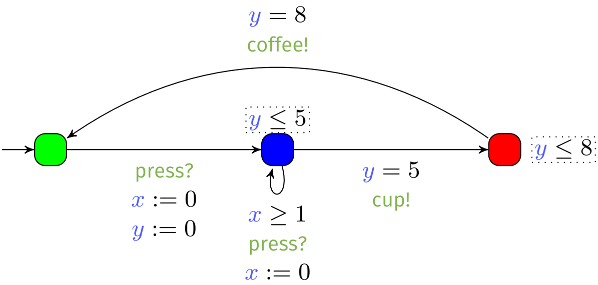
🟥 delivering coffee

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



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

The most critical system: The coffee machine

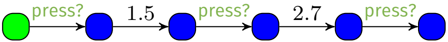


🟩 idle

🟦 adding sugar

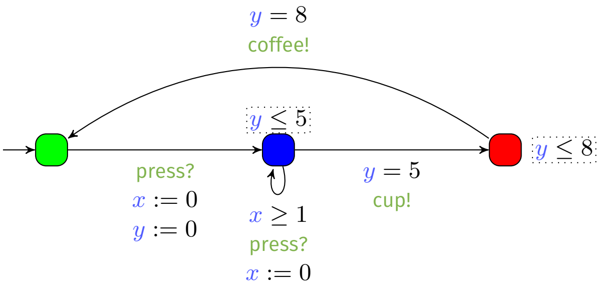
🟥 delivering coffee

- Example of concrete run for the coffee machine
  - 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

The most critical system: The coffee machine

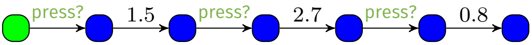


🟩 idle

🟦 adding sugar

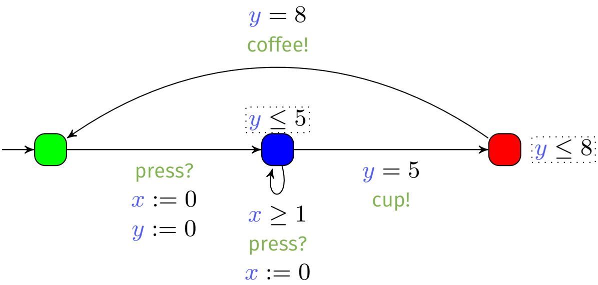
🟥 delivering coffee

- Example of concrete run for the coffee machine
  - 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

The most critical system: The coffee machine

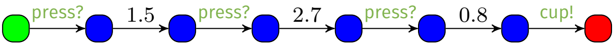


🟩 idle

🟦 adding sugar

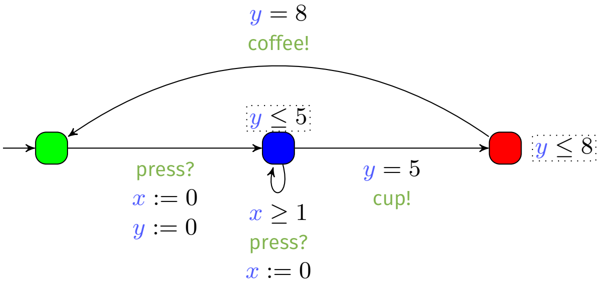
🟥 delivering coffee

- Example of concrete run for the coffee machine
  - 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

The most critical system: The coffee machine

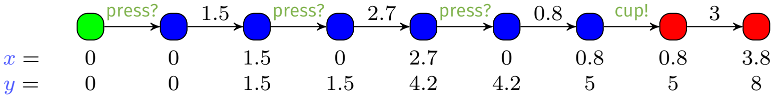


🟩 idle

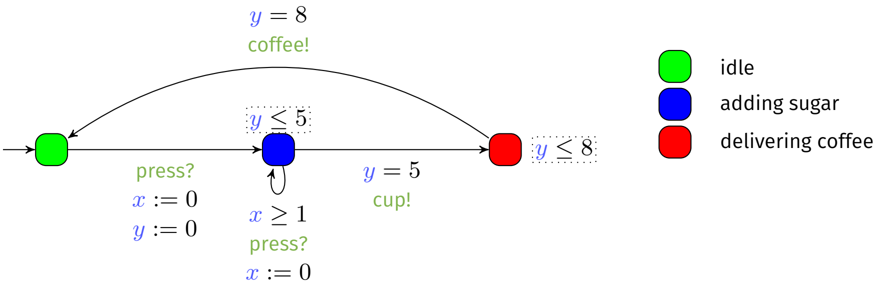
🟦 adding sugar

🟥 delivering coffee

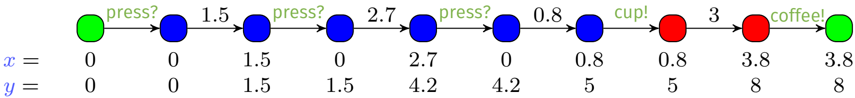
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar


press?
1.5
press?
2.7
press?
0.8
cup!
3
$x =$ 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

The most critical system: The coffee machine



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



press?
1.5
press?
2.7
press?
0.8
cup!
3
coffee!
$x =$ 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

Concrete semantics of timed automata

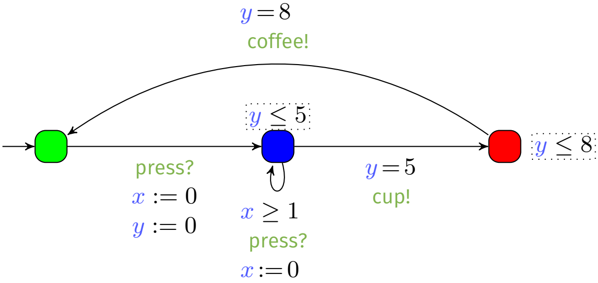
- Concrete state of a TA: pair (l, w), where
  - l is a location,
  - w is a valuation of each clock

  Example: $\Big($🟦$, \begin{pmatrix} x=1.2 \\ y=3.7 \end{pmatrix}\Big)$

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

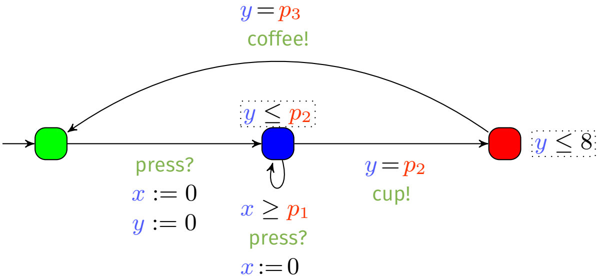
Parametric Timed Automaton (PTA)

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



Parametric Timed Automaton (PTA)

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



Symbolic semantics of parametric timed automata

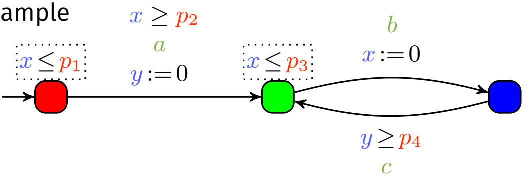
- Symbolic state of a PTA: pair (l, C), where
  - l is a location,
  - C is a convex polyhedron over X and P with a special form, called parametric zone [Hune et al., 2002]

Symbolic semantics of parametric timed automata

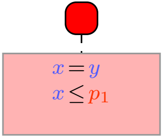
- Symbolic state of a PTA: pair (l, C), where
  - l is a location,
  - C is a convex polyhedron over X and P with a special form, called parametric zone [Hune et al., 2002]
- Symbolic run: alternating sequence of symbolic states and actions

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

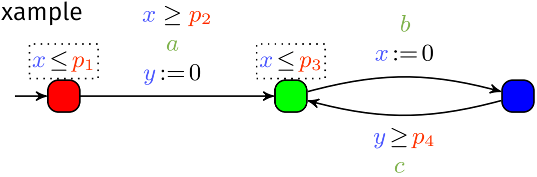


  - Possible symbolic run for this PTA

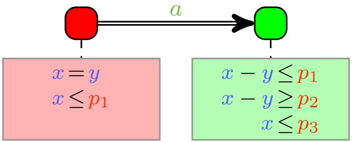


Symbolic semantics of parametric timed automata

- 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

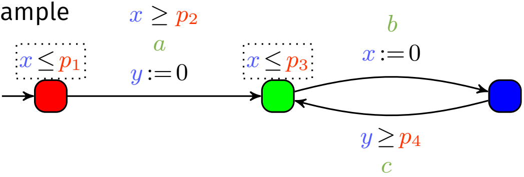


$x = y$
$x \leq p_1$

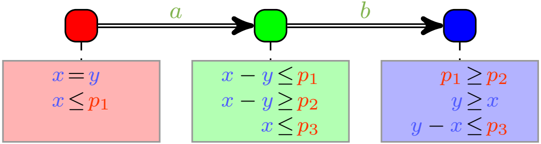
$x - y \leq p_1$
$x - y \geq p_2$
$x \leq p_3$

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



Licensing

Source of the graphics used I

Title: 1960 Citroen DS19
Author: Joc281
Source: `https://en.wikipedia.org/wiki/File:800px_1973_377_Citroen_DS19_automatically_guided_motor`
License: CC by-sa 3.0

Title: A Cartoon Businessman Reading A Text Message
Author: Vector Toons
Source: `https://en.wikipedia.org/wiki/File:800px_1973_377_Citroen_DS19_automatically_guided_motor`
License: CC by-sa 4.0

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

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

License of this document

Author: **Étienne André**