Modèles et Algorithmes pour la Vérification des Systèmes Temporisés

Patricia Bouyer

LSV – CNRS UMR 8643 & ENS de Cachan
Necessity of checking software-dependent systems

✔ software is everywhere!
✔ some famous crashes
Necessity of checking software-dependent systems

✔ software is everywhere!
✔ some famous crashes

Formal methods aim at proposing techniques to prevent crashes in (critical) systems.

✔ Automatic test generation
✔ Theorem proving
✔ Model-checking
Does the system satisfy the property?

Modelling
Model-checking

Does the system satisfy the property?

Modelling

Model-checking Algorithm

\( \models \varphi \)
Several directions of research

✔ Development of models:
  - expressiveness
  - compactness
  - decidability

✔ Development of efficient algorithms:
  - heuristics
  - data structures

✔ Tools & case studies:
  - implementation
  - applying tools to real-life examples
The basic model

Timed automata [AD 90’s]

$x, y$: clocks

$q_0 \xrightarrow{x \leq 5, a, y := 0} q_1 \xrightarrow{x - y > 3, b, x := 0} q_2$
The basic model

Timed automata [AD 90’s]

$x, y$: clocks

$q_0 \xrightarrow{x \leq 5, a, y := 0} q_1 \xrightarrow{x - y > 3, b, x := 0} q_2$

$q_0 \xrightarrow{\epsilon(4.1)} q_0 \xrightarrow{a} q_1 \xrightarrow{\epsilon(1.4)} q_1 \xrightarrow{b} q_2$

$x \quad 0 \quad 4.1 \quad 4.1 \quad 5.5 \quad 0$

$y \quad 0 \quad 4.1 \quad 0 \quad 1.4 \quad 1.4$
Results / Roadmap of the talk

(I) An extension of timed automata with updates (UTA)
   ✓ Precise decidability results
   ✓ Expressiveness

(II) Looking for a robust timed languages theory
   ✓ Proposition of a Kleene theorem for UTA
   ✓ Data languages, an algebraic framework
   ✓ Data languages, a logical characterization

(III) Reachability, an important question
   ✓ An implementable forward analysis algorithm
     (application to UTA)
   ✓ Test automata: the limit of reachability testing

(IV) A case study, the PGM protocol
Roadmap of the talk (cont.)

(I) An extension of timed automata with updates (UTA)
   ✔ Precise decidability results
   ✔ Expressiveness

(II) Looking for a robust timed language theory
   ✔ Proposition of a Kleene theorem for UTA
   ✔ Data languages, an algebraic framework
   ✔ Data languages, a logical characterization

(III) Reachability, an important question
   ✔ An implementable forward analysis algorithm
      (application to UTA)
   ✔ Test automata: the limit of reachability testing

(IV) A case study, the PGM protocol
Updatable timed automata

Part of the model of the ABR protocol [BF99]

$\text{newRM}, \quad R > 0$

$\text{snapshot}, \quad S = t$

$\text{Wait} \rightarrow \text{EndE}$

$\text{UpdE} \rightarrow \text{Idle}$

$t < S+a$

$S+a \leq t < S+b \land R \leq E$

$S+a \leq t < S+b \land R > E$

$t \geq S+b, \quad E := R$

$\text{Idle} \rightarrow \text{EndI}$

$\text{EndI}$

Models and Algorithms for the Verification of Timed Systems – PhD Defense – p.8
Operations on clocks:

- ✔ $x := c$, $x := y + c$  
  deterministic updates

- ✔ $x \sim c$, $x \sim y + c$  
  non-deterministic updates

$\sim \in \{<; \leq; \geq; >\}$
Decidability results

\[ x := x - 1, \quad x := x + 1 \quad \text{Undecidable (UTA)} \]

\[ x := 0 \quad \text{Decidable (TA)} \]
### Decidability results

<table>
<thead>
<tr>
<th>Action</th>
<th>Decidability</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x := x - 1, x := x + 1$</td>
<td>Undecidable</td>
<td>(UTA)</td>
</tr>
<tr>
<td>$x := 0$</td>
<td>Decidable</td>
<td>(TA)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action</th>
<th>Decidability</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x := c, x := y$</td>
<td>PSPACE-complete</td>
<td></td>
</tr>
<tr>
<td>$x := x + 1$</td>
<td>Undecidable</td>
<td></td>
</tr>
<tr>
<td>$x := y + c$</td>
<td>Undecidable</td>
<td></td>
</tr>
<tr>
<td>$x := x - 1$</td>
<td>Undecidable</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action</th>
<th>Decidability</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x &lt;: c$</td>
<td>PSPACE-complete</td>
<td></td>
</tr>
<tr>
<td>$x :&gt; c$</td>
<td>Undecidable</td>
<td></td>
</tr>
<tr>
<td>$x :\sim y + c$</td>
<td>Undecidable</td>
<td></td>
</tr>
<tr>
<td>$y + c :&lt; x :&lt; y + d$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$y + c :&lt; x :&lt; z + d$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$\sim \in \{<, \leq, \geq, >\}$

[Bouyer, Dufourd, Fleury, Petit - CAV’00]
### Decidability results

- \( x := x - 1, \ x := x + 1 \) \hspace{1cm} \text{Undecidable} \hspace{1cm} \text{(UTA)}
- \( x := 0 \) \hspace{1cm} \text{Decidable} \hspace{1cm} \text{(TA)}

<table>
<thead>
<tr>
<th>Expression</th>
<th>( x \sim c, \ x - y \sim c )</th>
<th>( x \sim c )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x := c, \ x := y )</td>
<td>PSPACE-complete</td>
<td></td>
</tr>
<tr>
<td>( x := x + 1 )</td>
<td></td>
<td>PSPACE-complete</td>
</tr>
<tr>
<td>( x := y + c )</td>
<td>Undecidable</td>
<td></td>
</tr>
<tr>
<td>( x := x - 1 )</td>
<td></td>
<td>Undecidable</td>
</tr>
<tr>
<td>( x :&lt; c )</td>
<td>PSPACE-complete</td>
<td></td>
</tr>
<tr>
<td>( x :&gt; c )</td>
<td></td>
<td>PSPACE-complete</td>
</tr>
<tr>
<td>( x :\sim y + c )</td>
<td>Undecidable</td>
<td></td>
</tr>
<tr>
<td>( y + c :&lt; x :&lt; y + d )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( y + c :&lt; x :&lt; z + d )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( \sim \in \{<, \leq, \geq, >\} \)

[Bouyer,Dufourd,Fleury,Petit - CAV’00]
Decidability results (cont.)

Decidability algorithm:

\[ \mathcal{A} \rightsquigarrow \text{Diophantine linear inequations system} \]
\[ \rightsquigarrow \text{Is there a solution?} \]
\[ \rightsquigarrow \text{If yes, belongs to a decidable class} \]
Decidability results (cont.)

Decidability algorithm:

\( A \leadsto \) Diophantine linear inequations system

\( \leadsto \) Is there a solution? (decidable)

\( \leadsto \) If yes, belongs to a decidable class
Decidability results (cont.)

Decidability algorithm:
\[ A \sim \rightarrow \text{Diophantine linear inequations system} \]
\[ \sim \rightarrow \text{Is there a solution? (decidable)} \]
\[ \sim \rightarrow \text{If yes, belongs to a decidable class} \]

Examples:

✔ constraint \( x - y \sim c \) involves the condition \[ c \leq \max_{x,y} \]

✔ update \( x :\sim y + c \) involves the condition \[ \max_{x} \leq \max_{y} + c \]

The solutions \((\max_{x})\) and \((\max_{x,y})\) define a set of regions.
A taste of expressiveness

UTA from a decidable class

state explosion

TA with silent actions

[Bouyer, Dufourd, Fleury, Petit - MFCS'00]
Roadmap of the talk (cont.)

(I) An extension of timed automata with updates (UTA)
- Precise decidability results
- Expressiveness

(II) Looking for a robust timed language theory
- Proposition of a Kleene theorem for UTA
- Data languages, an algebraic framework
- Data languages, a logical characterization

(III) Reachability, an important question
- An implementable forward analysis algorithm
  (application to UTA)
- Test automata: the limit of reachability testing

(IV) A case study, the PGM protocol
### Equivalences

<table>
<thead>
<tr>
<th>Untimed case</th>
<th>Timed case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finite automata</td>
<td></td>
</tr>
<tr>
<td>Rational expressions</td>
<td></td>
</tr>
<tr>
<td>Logical characterization</td>
<td></td>
</tr>
<tr>
<td>MSO((&lt;)), LTL</td>
<td></td>
</tr>
<tr>
<td>Algebraic characterization</td>
<td></td>
</tr>
<tr>
<td>(using monoids)</td>
<td></td>
</tr>
<tr>
<td>Untimed case</td>
<td>Timed case</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>Finite automata</td>
<td>Timed automata</td>
</tr>
<tr>
<td></td>
<td>[AD90]</td>
</tr>
<tr>
<td>Rational expressions</td>
<td>Some proposals</td>
</tr>
<tr>
<td></td>
<td>[ACM97/01,Asa98]</td>
</tr>
<tr>
<td></td>
<td>[Bouyer,Petit - ICALP’99]</td>
</tr>
<tr>
<td></td>
<td>[Bouyer,Petit - To appear in JALC]</td>
</tr>
<tr>
<td>Logical characterization</td>
<td>Some ad hoc logics</td>
</tr>
<tr>
<td>MSO(&lt;), LTL</td>
<td>[Wilke94]</td>
</tr>
<tr>
<td>Algebraic characterization</td>
<td></td>
</tr>
<tr>
<td>(using monoids)</td>
<td></td>
</tr>
</tbody>
</table>
Algebraic mechanism

\[ a_1 \ldots a_n \in \Sigma^* \rightarrow \text{Mechanism} \rightarrow m \in M \]
Algebraic mechanism

- \( a_1 \ldots a_n \in \Sigma^* \rightarrow \text{Mechanism} \rightarrow m \in M \)

- Need for a more elaborate mechanism than a simple morphism

\[
L = \{ (a, t)(a, t') \mid t < t'; \ t, t' \in \mathbb{Q}^+ \}
\]
Algebraic mechanism

\[ a_1 \ldots a_n \in \Sigma^* \quad \rightarrow \quad \text{Mechanism} \quad \rightarrow \quad m \in M \]

Need for a more elaborate mechanism than a simple morphism

\[ L = \{(a, t)(a, t') \mid t < t'; \; t, t' \in \mathbb{Q}^+\} \]

A more general framework: we consider a set of data \( D \), with an initial data \( \bot \).

Example: \( D = \mathbb{Q}^+ \) and \( \bot = 0 \)

We are interested in the data languages, i.e. on the alphabet \( \Sigma \times D \)
Our definition

We are given:

✔ a finite monoid $M$

✔ a finite “memory” consisting in $k$ registers
We are given:

✓ a finite monoid $M$
✓ a finite “memory” consisting in $k$ registers

$\theta'_i = \theta_i$ or $d$ depending on $m$ and $a$
✓ $m'$ depends on $m$, on $a$ and finitely on $(\theta'_i)_{i=1...k}$
Data automaton with \( k \) registers:

\[
\begin{align*}
\cdots & \quad q \quad g, \quad a, \quad up, \quad g' \quad (a, d) \quad q' \quad \cdots \\
& \quad r_1 \quad \begin{pmatrix} d_1 \\ \vdots \\ d_k \end{pmatrix} \quad \begin{pmatrix} d'_1 \\ \vdots \\ d'_k \end{pmatrix} \\
& \quad r_k
\end{align*}
\]

with \((d_i)_{i=1}^{n} \in g, (d'_i)_{i=1}^{n} \in g'\) and \(d'_i = d_i\) if \(r_i \not\in up\) and \(d'_i = d\) if \(r_i \in up\).
The data language

\[ L = \{(a, \tau)(a, 2\tau) \ldots (a, n\tau) \mid n \in \mathbb{N}, \ \tau > 0\} \]

is accepted by:

\[ (a, t_0)(a, t_1) \ldots (a, t_n) \in L \iff t_0 = t_{i+1} - t_i \text{ for all } i \]
**Theorem:** equivalence between monoid recognizability and data automata acceptance.

- Proof close to the one in the untimed case.
- The same number of registers.

**Property:** The monoid plays a fundamental role. “Two distinct varieties of monoids generate two different classes of data languages.”

[Bouyer, Petit, Thérien - CONCUR'01]
[Bouyer, Petit, Thérien - Sub. to Inf. and Comp.]
## Summary

<table>
<thead>
<tr>
<th>Untimed case</th>
<th>Data case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finite automata</td>
<td>Data automata</td>
</tr>
<tr>
<td>Rational expressions</td>
<td></td>
</tr>
<tr>
<td>Logical characterization</td>
<td></td>
</tr>
<tr>
<td>MSO(&lt;), LTL</td>
<td></td>
</tr>
<tr>
<td>Algebraic characterization</td>
<td></td>
</tr>
<tr>
<td>(using monoids)</td>
<td>✔</td>
</tr>
<tr>
<td>Untimed case</td>
<td>Data case</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>Finite automata</td>
<td>Data automata</td>
</tr>
<tr>
<td>Rational expressions</td>
<td></td>
</tr>
<tr>
<td>Logical characterization</td>
<td></td>
</tr>
<tr>
<td>MSO(&lt;), LTL</td>
<td>[Bouyer - To appear in IPL]</td>
</tr>
<tr>
<td>Algebraic characterization</td>
<td>✔</td>
</tr>
<tr>
<td>(using monoids)</td>
<td></td>
</tr>
</tbody>
</table>
## Summary

<table>
<thead>
<tr>
<th><strong>Untimed case</strong></th>
<th><strong>Data case</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Finite automata</td>
<td>Data automata</td>
</tr>
<tr>
<td>Rational expressions</td>
<td>Adaptation of our Kleene theorem in the data framework</td>
</tr>
<tr>
<td>Logical characterization</td>
<td>[Bouyer - To appear in IPL]</td>
</tr>
<tr>
<td><strong>MSO((&lt;)), LTL</strong></td>
<td>✔</td>
</tr>
<tr>
<td>Algebraic characterization (using monoids)</td>
<td>✔</td>
</tr>
</tbody>
</table>

[Bouyer - To appear in IPL]
Roadmap of the talk (cont.)

(I) An extension of timed automata with updates (UTA)
   ✔ Precise decidability results
   ✔ Expressiveness

(II) Looking for a robust timed language theory
   ✔ Proposition of a Kleene theorem for UTA
   ✔ Data languages, an algebraic framework
   ✔ Data languages, a logical characterization

(III) Reachability, an important question
   ✔ An implementable forward analysis algorithm
     (application to UTA)
   ✔ Test automata: the limit of reachability testing

(IV) A case study, the PGM protocol
Two main methods

✔ Forward analysis
Two main methods

✔ Forward analysis
Two main methods

- ✔ Forward analysis
- ✔ Backward analysis
Two main methods

✔ Forward analysis

✔ Backward analysis
In TA, **extrapolation** is necessary to ensure termination of the forward analysis.

The usual extrapolation **is not correct** for UTA.

Proposal of an extension of the notion of extrapolation for UTA.

Proof of correctness w.r.t. reachability.

Adaptation of the **DBM data structure**.
The limit of reachability

✔ Efficient algorithms for reachability testing are implemented
✔ How can we use them for testing other properties?
The limit of reachability

✔ Efficient algorithms for reachability testing are implemented
✔ How can we use them for testing other properties?

The idea [ABL98]:

Test automaton
What kind of properties?

“No cash is given before the PIN code is correct”
Let $L_{\forall S}$ be the logic defined inductively by:

$$\varphi ::= \text{ff} \mid \varphi_1 \land \varphi_2 \mid g \lor \varphi \mid \forall_S \varphi \,(S \,\text{urgent}) \mid [a] \varphi$$

$$\mid \langle a \rangle \,\mathbf{tt} \,(a \,\text{urgent}) \mid x \,\text{in} \varphi \mid Z \mid \max(Z, \varphi)$$

$$g ::= x \sim p \mid x - y \sim p \,(\sim \in \{<; \leq; =; \geq; >\})$$
Let $L_{\forall S}$ be the logic defined inductively by:

$$
\varphi ::= \top | \varphi_1 \wedge \varphi_2 | g \lor \varphi | \forall_S \varphi \ (S \text{ urgent}) | [a] \varphi
$$
$$
g ::= x \sim p | x - y \sim p \ (\sim \in \{<; \leq; =; \geq; >\})
$$

$\rightarrow$ previous property: $\forall_{\{\text{pin_code}\}}[\text{cash}]\top$
**Property:** every formula from $\mathcal{L}_{\forall S}$ is **testable**.

$\forall \varphi \in \mathcal{L}_{\forall S}, \exists T\varphi$ test automaton s.t. $\forall A,$

\[
A \models \varphi \iff A \parallel T\varphi \not\models \varnothing
\]

**Property:** $\mathcal{L}_{\forall S}$ is a **compositional logic**, that is:

$\forall T, \forall \varphi \in \mathcal{L}_{\forall S}, \exists \psi \in \mathcal{L}_{\forall S}$ s.t. $\forall A,$

\[
A \parallel T \models \varphi \iff A \models \psi
\]

**Corollary:** $\mathcal{L}_{\forall S}$ is a **complete logic** w.r.t. test automata.

$\forall T$ test automaton, $\exists \varphi_T \in \mathcal{L}_{\forall S}$ s.t. $\forall A,$

\[
A \parallel T \not\models \varnothing \iff A \models \varphi_T
\]

[Aceto,Bouyer,Burgueño,Larsen - FST&TCS’98]
[Aceto,Bouyer,Burgueño,Larsen - To appear in TCS]
Roadmap of the talk (cont.)

(I) An extension of timed automata with updates (UTA)
   ✔ Precise decidability results
   ✔ Expressiveness

(II) Looking for a robust timed language theory
   ✔ Proposition of a Kleene theorem for UTA
   ✔ Data languages, an algebraic framework
   ✔ Data languages, a logical characterization

(III) Reachability, an important question
   ✔ An implementable forward analysis algorithm
      (application to UTA)
   ✔ Test automata: the limit of reachability testing

(IV) A case study, the PGM protocol
What’s PGM?

Pragmatic General Multicast

Source

Transmission

Receiver

O/RDATA SPM

NAK

NAK

Models and Algorithms for the Verification of Timed Systems – PhD Defense – p.29
Pragmatic General Multicast

- Some simplifications
- Two properties to be tested
- Modelling in UPPAAL
- 7 parameters which can be changed
- Memory explosion...
- ... but valuable results
One of the experiments

“Each data which is detected as lost, is repaired.”

Results for two receivers:

| PERIODE_NAK | 3 | 3 | 3 | 3 |
| MIN_PERIODE_DATA | 1 | 2 | 3 | 4 |
| Answer | False | False | True | True |
| Options | Breadth-first | Active-clock reduction | Breadth-first | Active-clock reduction | Convex-hull approximation |
| Time (in s) | 4743 | 5775 | 918 | 939 |
| Memory (in KB) | 1239832 | 1402200 | 630608 | 597440 |
| SPM_DELAI | 1 | | | |
| MAX_DATA_DELAI | 3 | | | |
| MAX_PERIODE_SPM | 5 | | | |
| MIN_DATA_DELAI | 2 | | | |
| MIN_PERIODE_SPM | 3 | | | |

[Sun Ultra 220R
2x450 MHz CPU
2048 MB RAM]

[Bérard,Bouyer,Petit - Calife report]
Conclusion

(I) An extension of timed automata with updates (UTA)
  ✔ Precise decidability results [Bouyer,Dufourd,Fleury,Petit - CAV’00]
  ✔ Expressiveness [Bouyer,Dufourd,Fleury,Petit - MFCS’00]

(II) Looking for a robust timed languages theory
  ✔ Data languages, an algebraic framework
    [Bouyer,Petit,Thérien - CONCUR’01 - Sub. to Inf. and Comp.]
  ✔ Data languages, a logical characterization [Bouyer - To appear in IPL]

(III) Reachability, an important question
  ✔ An implementable forward analysis algorithm (application to UTA)
    [Bouyer - Sub. to Formal Methods in System Design]
  ✔ Test automata: the limit of reachability testing [Aceto,Bouyer,Burgueño,Larsen - FST&TCS’98 - To appear in TCS]

(IV) A case study, the PGM protocol [Bérard,Bouyer,Petit - Calife report]
Further work

✔ Updatable timed automata:
  - get a high speed-up for reachability checking
  - backward analysis algorithm + TCTL algorithm
  - implementation

✔ Data languages:
  - adaptation of the Kleene theorem
  - remarkable subclasses
The usual extrapolation is not correct for UTA

\[ a, \ y := 0 \quad \rightarrow \quad b, \ y := 1 \quad \rightarrow \quad x - y < 1, \ c \]