Model-Checking Timed Temporal Logics

Patricia Bouyer

LSV – CNRS & ENS de Cachan – France
Oxford University Computing Laboratory – UK

Based on joint works with Fabrice Chevalier, Nicolas Markey, Joël Ouaknine and James Worrell
Outline

1. Introduction

2. Definition of the logics

3. The timed automaton model

4. The model-checking problem

5. Some interesting fragments

6. Conclusion
Model-checking

system:

property:
Model-checking

system: elevator with people

property: G (request → F grant)
Model-checking

system:

property:

G (request → F grant)

model-checking algorithm
Model-checking

system:

property:

G (request → F grant)

model-checking algorithm

yes/no
The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

\[
\text{LTL } \varphi ::= p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \neg \varphi \mid X \varphi \mid \varphi U \varphi
\]

[Pnu77] Pnueli. The temporal logic of programs (FOCS’77).
The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

\[
\text{LTL } \ni \phi ::= p \mid \phi \land \phi \mid \phi \lor \phi \mid \neg \phi \mid X \phi \mid \phi U \phi
\]

\[= \quad X \bullet \quad \]

[Pnu77] Pnueli. The temporal logic of programs (FOCS’77).
The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

\[
\text{LTL } \varphi ::= p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \neg \varphi \mid X \varphi \mid \varphi U \varphi
\]

[Pnu77] Pnueli. The temporal logic of programs (FOCS’77).
The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

\[ \text{LTL } \exists \varphi ::= p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \neg \varphi \mid X \varphi \mid \varphi U \varphi \]

\[ \models X \cdot \]

\[ \models \cdot U \cdot \]

[Pnu77] Pnueli. The temporal logic of programs (FOCS’77).
The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

\[ \text{LTL } \varphi ::= p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \neg \varphi \mid X \varphi \mid \varphi \mathbf{U} \varphi \]

[\text{Pnu77}] Pnueli. The temporal logic of programs (FOCS'77).
The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

\[
\begin{align*}
\text{LTL } &\varphi \ ::= \ p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \neg \varphi \mid X \varphi \mid \varphi U \varphi \\
&= X\bullet \\
&= \bullet U \bullet \\
&= F\bullet \equiv \text{tt}\ U \bullet
\end{align*}
\]

[Pnu77] Pnueli. The temporal logic of programs (FOCS’77).
The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

\[ \text{LTL } \varphi ::= p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \neg \varphi \mid X \varphi \mid \varphi U \varphi \]

\[ |= X \bullet \]

\[ |= \bullet U \bullet \]

\[ |= F \bullet \equiv \text{tt U} \bullet \]

[Pnu77] Pnueli. The temporal logic of programs (FOCS’77).
The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

\[
\text{LTL } \varphi ::= p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \neg \varphi \mid X \varphi \mid \varphi U \varphi
\]

\[
\begin{align*}
\Downarrow & \quad X \cdot & \Downarrow & \quad .U. & \Downarrow & \quad F \cdot \equiv \text{tt } U \cdot \\
\Downarrow & \quad . & \Downarrow & \quad G \cdot \equiv \neg F \neg \cdot
\end{align*}
\]
The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

\[ \text{LTL } \exists \varphi ::= p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \neg \varphi \mid X \varphi \mid \varphi U \varphi \]

- response property:

\[ G (\bullet \rightarrow F \bullet) \]

[Pnu77] Pnueli. The temporal logic of programs (FOCS'77).
The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

\[ \text{LTL } \exists \varphi ::= \ p \ | \ \varphi \land \varphi \ | \ \varphi \lor \varphi \ | \ \neg \varphi \ | \ X \varphi \ | \ \varphi U \varphi \]

- response property:
  \[ G (\bullet \rightarrow F \bullet) \]

- liveness property:
  \[ GF \bullet \]

[Pnu77] Pnueli. The temporal logic of programs (FOCS’77).
The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

\[
\text{LTL} \ni \varphi ::= p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \neg \varphi \mid X \varphi \mid \varphi U \varphi
\]

- response property:

  \(G (\bullet \rightarrow F \bullet)\)

- liveness property:

  \(GF \bullet\)

- safety property:

  \(G \neg \bullet\)

[Pnu77] Pnueli. The temporal logic of programs (FOCS'77).
The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

\[ \text{LTL } \varphi ::= p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \neg \varphi \mid X \varphi \mid \varphi U \varphi \]

- response property:
  \[ G (\bullet \rightarrow F \bullet) \]

- liveness property:
  \[ GF \bullet \]

- safety property:
  \[ G \neg \bullet \]

- a more complex property:
  \[ (\bullet \land (F \bullet \lor G \bullet)) U \bullet \]

[Pnu77] Pnueli. The temporal logic of programs (FOCS’77).
Adding timing requirements

- Need for **timed models**
  - the behaviour of most systems depends on time;
  - faithful modelling has to take time into account.

  timed automata, time(d) Petri nets, timed process algebras...
Adding timing requirements

- Need for **timed models**
  - the behaviour of most systems depends on time;
  - faithful modelling has to take time into account.
  - timed automata, time(d) Petri nets, timed process algebras...

- Need for **timed specification languages**
  - the behaviour of most systems depends on time;
  - untimed specifications are not sufficient
    (for instance, bounded response timed, etc...)
  - TCTL, MTL, TPTL, timed $\mu$-calculus...
Outline

1. Introduction

2. Definition of the logics

3. The timed automaton model

4. The model-checking problem

5. Some interesting fragments

6. Conclusion
Metric Temporal Logic (MTL)

\[ \text{MTL } \exists \varphi ::= a \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \varphi \mathcal{U}_I \varphi \]

where \( I \) is an interval with integral bounds.

Definition of the logics

Metric Temporal Logic (MTL)

\[
\text{MTL } \exists \varphi ::= a \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \varphi U I \varphi
\]

where \( I \) is an interval with integral bounds.

- This is a timed extension of LTL

Metric Temporal Logic (MTL)

MTL \varphi ::= a \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \varphi \mathbb{U} I \varphi

where \( I \) is an interval with integral bounds.

- This is a timed extension of LTL
- Can be interpreted over timed words, or over signals
  - this distinction is fundamental

Metric Temporal Logic (MTL)

MTL ⊨ φ ::= a | ¬φ | φ ∨ φ | φ ∧ φ | φ Uᵢ φ

where I is an interval with integral bounds.

- This is a timed extension of LTL
- Can be interpreted over timed words, or over signals
  - this distinction is fundamental
- Can be interpreted over finite or infinite behaviours
  - this distinction is fundamental

The pointwise semantics

MTL formulas are interpreted over timed words:

\[(\bullet, .6)(\circ, 1.1)(\bullet, 1.2)(\bullet, 1.3) \ldots\]
The pointwise semantics

MTL formulas are interpreted over timed words:

\[(\bullet, .6)(\bullet, 1.1)(\bullet, 1.2)(\bullet, 1.3)\ldots\]

the system is observed only when actions happen
The pointwise semantics

MTL formulas are interpreted over timed words:

\[(\bullet, .6)(\bullet, 1.1)(\bullet, 1.2)(\bullet, 1.3) \ldots\]

\[\models\bullet U_{[1,2]} \bullet\]

the system is observed only when actions happen
The pointwise semantics

MTL formulas are interpreted over timed words:

\[(\bullet, .6)(\bullet, 1.1)(\bullet, 1.2)(\bullet, 1.3) \ldots\]

the system is observed only when actions happen
The continuous semantics

MTL formulas are interpreted over (finitely variable) signals:

<table>
<thead>
<tr>
<th>t ∈ [0, .6]</th>
<th>t ∈ (.6, 1.1)</th>
<th>t ∈ [1.1, 1.2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

...
The continuous semantics

MTL formulas are interpreted over (finitely variable) signals:

\[ t \in [0, 0.6] \mapsto \square \]
\[ t \in (0.6, 1.1) \mapsto \]
\[ t \in [1.1, 1.2) \mapsto \]

... the system is observed continuously
The continuous semantics

MTL formulas are interpreted over (finitely variable) signals:

\[ t \in [0,0.6] \mapsto \square \]
\[ t \in (0.6, 1.1) \mapsto \]
\[ t \in [1.1, 1.2] \mapsto \]

\[ \vdash \mathbf{U}_{[1,2]} \bullet \]

the system is observed continuously
The continuous semantics

MTL formulas are interpreted over (finitely variable) signals:

\[ t \in [0, .6] \mapsto \ \square \]
\[ t \in (.6, 1.1) \mapsto \ \square \]
\[ t \in [1.1, 1.2) \mapsto \ \square \]

... the system is observed continuously

\[ \models \ U_{[1,2]} \]
\[ \not\models \ G_{[2,3]} \]
Some examples

- “Every problem is followed within 56 time units by an alarm”
  \[ \mathbf{G} (\text{problem} \rightarrow \mathbf{F}_{\leq 56} \text{alarm}) \]
Some examples

- “Every problem is followed within 56 time units by an alarm”
  \[ G(\text{problem} \rightarrow F_{\leq 56} \text{alarm}) \]

- “Each time there is a problem, it is either repaired within the next 15 time units, or an alarm rings during 3 time units 12 time units later”
  \[ G(\text{problem} \rightarrow (F_{\leq 15} \text{repair} \lor G_{[12,15]} \text{alarm})) \]
Some examples

- “Every problem is followed within 56 time units by an alarm”
  \[G(\text{problem} \rightarrow F_{\leq 56} \text{alarm})]\]

- “Each time there is a problem, it is either repaired within the next 15 time units, or an alarm rings during 3 time units 12 time units later”
  \[G(\text{problem} \rightarrow (F_{\leq 15} \text{repair} \lor G_{[12,15]} \text{alarm}))]\]

- \(F_{=2} \text{repair} \quad \text{vs} \quad F_{=1}(F_{=1} \text{repair})\)
Some examples

► “Every problem is followed within 56 time units by an alarm”

\[ \mathbf{G} (\text{problem} \rightarrow \mathbf{F}_{\leq 56} \text{alarm}) \]

► “Each time there is a problem, it is either repaired within the next 15 time units, or an alarm rings during 3 time units 12 time units later”

\[ \mathbf{G} (\text{problem} \rightarrow (\mathbf{F}_{\leq 15} \text{repair} \lor \mathbf{G}_{[12,15)} \text{alarm})) \]

► \( \mathbf{F}_{=2} \text{repair} \) vs \( \mathbf{F}_{=1} (\mathbf{F}_{=1} \text{repair}) \)

\[ \models \mathbf{F}_{=2} \text{•} \quad \not\models \mathbf{F}_{=1} (\mathbf{F}_{=1} \text{•}) \]
Some examples

▶ “Every problem is followed within 56 time units by an alarm”
\[ G(\text{problem} \rightarrow F_{\leq 56} \text{alarm}) \]

▶ “Each time there is a problem, it is either repaired within the next 15 time units, or an alarm rings during 3 time units 12 time units later”
\[ G(\text{problem} \rightarrow (F_{\leq 15} \text{repair} \lor G_{[12,15]} \text{alarm})) \]

▶ \( F_2 \text{repair} \) vs \( F_1 (F_1 \text{repair}) \)

\[ \models F_2 \text{repair} \quad \models F_1 (F_1 \text{repair}) \]
Some examples

▸ “Every problem is followed within 56 time units by an alarm”
\[ G(\text{problem} \rightarrow F_{\leq 56} \text{alarm}) \]

▸ “Each time there is a problem, it is either repaired within the next 15 time units, or an alarm rings during 3 time units 12 time units later”
\[ G(\text{problem} \rightarrow (F_{\leq 15} \text{repair} \lor G_{[12,15]} \text{alarm})) \]

▸ \( F_2 \text{repair} \) vs \( F_1 (F_1 \text{repair}) \)

\[ \models F_2 \quad \not\models F_1 (F_1) \]

▸ in the pointwise semantics, \( F_2 \not\equiv F_1 F_1 \)

▸ in the continuous semantics, \( F_2 \equiv F_1 F_1 \)
Some further extensions

- Timed Propositional Temporal Logic (TPTL) [AH89]

\[ TPTL = LTL + \text{clock variables} + \text{clock constraints} \]

Some further extensions

- Timed Propositional Temporal Logic (TPTL)  

\[ \text{TPTL} = \text{LTL} + \text{clock variables} + \text{clock constraints} \]

\[
G(\text{problem} \rightarrow F_{\leq 56} \text{alarm}) \equiv G(\text{problem} \rightarrow x.F(\text{alarm} \land x \leq 56))
\]

Some further extensions

- Timed Propositional Temporal Logic (TPTL) [AH89]

\[ TPTL = LTL + \text{clock variables} + \text{clock constraints} \]

\[
G(\text{problem} \rightarrow F_{\leq 56} \text{alarm}) \equiv G(\text{problem} \rightarrow x. F(\text{alarm} \land x \leq 56))
\]

\[
G(\text{problem} \rightarrow x. F(\text{alarm} \land F(\text{failsafe} \land x \leq 56)))
\]

Some further extensions

- **Timed Propositional Temporal Logic (TPTL)**

  \[ \text{TPTL} = \text{LTL} + \text{clock variables} + \text{clock constraints} \]

  \[ G(\text{problem} \rightarrow F_{\leq 56} \text{alarm}) \equiv G(\text{problem} \rightarrow x. F(\text{alarm} \land x \leq 56)) \]

  \[ G(\text{problem} \rightarrow x. F(\text{alarm} \land F(\text{failsafe} \land x \leq 56))) \]

- **MTL+Past**: add past-time modalities

  \[ \text{[AH92]} \]

Some further extensions

- Timed Propositional Temporal Logic (TPTL)  

\[ \text{TPTL} = \text{LTL} + \text{clock variables} + \text{clock constraints} \]

\[
G(\text{problem} \rightarrow F_{\leq 56} \text{ alarm}) \equiv G(\text{problem} \rightarrow x. F(\text{alarm} \land x \leq 56))
\]

\[
G(\text{problem} \rightarrow x. F(\text{alarm} \land F(\text{failsafe} \land x \leq 56)))
\]

- MTL+Past: add past-time modalities  

\[
G(\text{alarm} \rightarrow F_{\leq 56}^{-1} \text{ problem})
\]


A note on the expressiveness

**Theorem**

$LTL + Past$ is as expressive as $LTL$ [Kam68,GPSS80].

---


A note on the expressiveness

**Theorem**

$LTL+Past$ is as expressive as $LTL$ [Kam68,GPSS80].

**Theorem**

$MTL$ is strictly less expressive than $MTL+Past$ and $TPTL$ [BCM05].

---

[BCM05] Bouyer, Chevalier, Markey. On the expressiveness of MTL and TPTL (FSTTCS’05).
A note on the expressiveness

Theorem

LTL+Past is as expressive as LTL [Kam68,GPSS80].

Theorem

MTL is strictly less expressive than MTL+Past and TPTL [BCM05].

Conjecture in 1990: the TPTL formula

\[ G (\bullet \rightarrow x. F (\bullet \land F (\bullet \land x \leq 2))) \]

cannot be expressed in MTL.

[BCM05] Bouyer, Chevalier, Markey. On the expressiveness of MTL and TPTL (FSTTCS’05).
A note on the expressiveness

**Theorem**

$LTL + Past$ is as expressive as $LTL$ [Kam68, GPSS80].

**Theorem**

$MTL$ is strictly less expressive than $MTL + Past$ and $TPTL$ [BCM05].

**Conjecture in 1990:** the $TPTL$ formula

$$G(\bullet \rightarrow x. F (\bullet \land F (\bullet \land x \leq 2)))$$

cannot be expressed in $MTL$.

- This is true in the **pointwise** semantics.

---

[BCM05] Bouyer, Chevalier, Markey. On the expressiveness of MTL and TPTL (FSTTCS’05).
A note on the expressiveness

**Theorem**

LTL+Past is as expressive as LTL [Kam68, GPSS80].

**Theorem**

MTL is strictly less expressive than MTL+Past and TPTL [BCM05].

**Conjecture in 1990:** the TPTL formula

\[
G (\bullet \rightarrow x. F (\bullet \land F (\bullet \land x \leq 2)))
\]

cannot be expressed in MTL.

- This is true in the pointwise semantics.
- This is wrong in the continuous semantics!

---


[BCM05] Bouyer, Chevalier, Markey. On the expressiveness of MTL and TPTL (FSTTCS’05).
The TPTL formula

\[ G (\bullet \rightarrow x. F (\bullet \land F (\bullet \land x \leq 2))) \]

can be expressed in MTL in the continuous semantics
The TPTL formula

\[ G(\bullet \rightarrow x.F(\bullet \land F(\bullet \land x \leq 2))) \]

can be expressed in MTL in the continuous semantics

\[
\begin{align*}
G(\bullet) & \rightarrow \left\{ \\
& \begin{array}{l}
F \leq 1 \land F(1,2) \lor F \leq 1 (F \leq 1 \land F = 1)
\end{array}
\right.
\end{align*}
\]
The TPTL formula

\[ G (\bullet \rightarrow \quad \mathbf{F} (\bullet \land \mathbf{F} (\bullet \land x \leq 2))) \]

can be expressed in MTL in the continuous semantics

\[ G \bullet \rightarrow \left\{ \begin{array}{c}
\mathbf{F}_{\leq 1} \quad \bullet \land \quad \mathbf{F}_{[1,2]} \quad \bullet
\end{array} \right. \]
The TPTL formula

\[ G(\bullet \rightarrow x.F(\bullet \land F(\bullet \land x \leq 2))) \]

can be expressed in MTL in the continuous semantics

\[ G \bullet \rightarrow \left\{ \begin{array}{c}
F_{\leq 1} \bullet \land F_{[1,2]} \\
\lor \\
F_{\leq 1} (\bullet \land F_{\leq 1} \bullet )
\end{array} \right. \]
The TPTL formula

\[ G (\bullet \rightarrow x. F (\bullet \land F (\bullet \land x \leq 2))) \]

can be expressed in MTL in the continuous semantics

\[
G \bullet \rightarrow \begin{cases}
F_{\leq 1} \bullet \land F_{[1,2]} \bullet \\
\lor \\
F_{\leq 1} (\bullet \land F_{\leq 1} \bullet)
\end{cases}
\]
The TPTL formula

\[ G (\bullet \rightarrow x.F (\bullet \land F (\bullet \land x \leq 2))) \]

can be expressed in MTL in the continuous semantics

\[ G (\bullet \rightarrow x.F (\bullet \land F (\bullet \land x \leq 2))) \leq_1 \]

\[ F_{\leq_1} \land F_{[1,2]} \]

\[ \lor \]

\[ F_{\leq_1} (\bullet \land F_{\leq_1}) \]
The TPTL formula

\[ \mathbf{G} (\bullet \rightarrow x. \mathbf{F} (\bullet \land \mathbf{F} (\bullet \land x \leq 2))) \]

can be expressed in MTL in the continuous semantics

\[
\mathbf{G} \bullet \rightarrow \begin{cases} 
\mathbf{F}_{\leq 1} \bullet \land \mathbf{F}_{[1,2]} \bullet \\
\lor \\
\mathbf{F}_{\leq 1} (\bullet \land \mathbf{F}_{\leq 1}) \\
\lor
\end{cases}
\]
The TPTL formula

\[ G (\bullet \rightarrow x. F (\bullet \land F (\bullet \land x \leq 2))) \]

can be expressed in MTL in the continuous semantics

\[
G (\bullet \rightarrow \bullet. F (\bullet \land F (\bullet \land x \leq 2)))
\]

0 \quad 1 \quad 2

\[ F = 1 \]

\[ G \bullet \rightarrow \begin{cases} F \leq 1 \bullet \land F_{[1,2]} \bullet \\ \lor \\ F \leq 1 (\bullet \land F \leq 1 \bullet) \\ \lor \\ F \leq 1 (F \leq 1 \bullet \land F = 1 \bullet) \end{cases} \]
Outline

1. Introduction

2. Definition of the logics

3. The timed automaton model

4. The model-checking problem

5. Some interesting fragments

6. Conclusion
Timed automata

Can be viewed:

▶ as the timed word

(problem, 23)

(delayed, 38.6)

(repair, 40.9)

(done, 63)

▶ as the signal
Timed automata
Timed automata

The timed automaton model

The timed automaton model can be viewed:

- as the timed word 
  
  \[(\text{problem}, 23) (\text{delayed}, 38.6) (\text{repair}, 40.9) (\text{done}, 63)\]

- as the signal

\[
\begin{array}{c|c|c}
\text{x} & 0 & 23 \\
\text{y} & 0 & 23 \\
\end{array}
\]
Timed automata

The timed automaton model

Can be viewed:
▶ as the timed word \((\text{problem}, 23)(\text{delayed}, 38.6)(\text{repair}, 40.9)(\text{done}, 63)\)

▶ as the signal
\[
\begin{array}{cccccccc}
 0 & 10 & 20 & 30 & 40 & 50 & 60 & 70 \\
\text{safe} & \text{alarm} & \text{failsafe} & \text{repairing} & \text{done}
\end{array}
\]

\[
\begin{array}{cccc}
x & 0 & 23 & 23 & 0 \\
y & 0 & 23 & 23 & 23
\end{array}
\]
Timed automata

The timed automaton model

Can be viewed:

- as the timed word
- as the signal

<table>
<thead>
<tr>
<th>x</th>
<th>safe</th>
<th>23</th>
<th>safe</th>
<th>problem</th>
<th>alarm</th>
<th>15.6</th>
<th>alarm</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>0</td>
<td>23</td>
<td>23</td>
<td>0</td>
<td>15.6</td>
<td>15.6</td>
<td>38.6</td>
</tr>
</tbody>
</table>

safe

alarm

repairing

done, 22 ≤ y ≤ 25

repair, x ≤ 15

repair

2 ≤ y ∧ x ≤ 56

y := 0

15 ≤ x ≤ 16

delayed, y := 0

failsafe

x := 0

y := 0
Timed automata

\[
\begin{align*}
x &:= 0, & y &:= 0 \\
15 \leq x &\leq 16 & 2 \leq y &\wedge x \leq 56 \\
\text{delayed} &\iff y = 0 \\
\text{repair} &\iff y = 0 \\
\text{done} &\iff 22 \leq y \leq 25 \\
\text{repairing} &\iff 2 \leq y \wedge x \leq 56 \\
\text{failsafe} &\iff x \leq 15
\end{align*}
\]
The timed automaton model

Timed automata

Can be viewed:
- as the timed word
  \((\text{problem}, 23)(\text{delayed}, 38.6)(\text{repair}, 40.9)(\text{done}, 63)\)

- as the signal

\[
\begin{array}{c|c|c|c|c|c|c}
\text{x} & \text{y} & 23 & 23 & 15.6 & 23 & 17.9 \\
\hline
0 & 0 & 23 & 23 & 38.6 & 0 & 2.3 \\
\end{array}
\]
The timed automaton model

Timed automata

Can be viewed: as the timed word \((\text{problem}, 23)(\text{delayed}, 38.6)(\text{repair}, 40.9)(\text{done}, 63)\)
Timed automata

The timed automaton model can be viewed as the timed word

\[
\text{problem, } x := 0 \rightarrow \text{alarm, } y := 0, 15 \leq x \leq 16 \rightarrow \text{delayed, } y := 0 \rightarrow \text{failsafe, } 2 \leq y \land x \leq 56 \rightarrow \text{repair, } x \leq 15 \rightarrow \text{reparation, } 22 \leq y \leq 25 \rightarrow \text{done}
\]

\[
x \begin{array}{c|c|c}
\text{safe} & 23 & \text{safe} \\
0 & 23 & 0
\end{array}
\begin{array}{c|c|c}
\text{problem} & \text{alarm} & 15.6 \\
23 & 0 & 15.6
\end{array}
\begin{array}{c|c|c}
\text{delayed} & \text{failsafe} & 2.3 \\
17.9 & 15.6 & 0
\end{array}
\begin{array}{c|c|c}
\text{repair} & \text{reparation} & 22.1 \\
17.9 & 40 & 0
\end{array}
\begin{array}{c|c|c}
\text{reparation} & 22.1 & \text{done}
\end{array}
\]
Timed automata

The timed automaton model can be viewed:

- as the timed word \((\text{problem, } x=0, 23)(\text{delayed, } y=0, 38.6)(\text{repair, } x \leq 15, 22.1)(\text{done, } y \leq 25, 40)\)
- as the signal

| \(x\) | 0 | 23 | 0 | 15.6 | 0 | 17.9 | 40 | 22.1 | 40 |
| \(y\) | 0 | 23 | 0 | 23  | 0 | 23   | 0  | 22.1 | 22.1|

\(x \leq 15\) and \(2 \leq y \land x \leq 56\)
Timed automata

Can be viewed:

- as the timed word \((\text{problem}, 23)(\text{delayed}, 38.6)(\text{repair}, 40.9)(\text{done}, 63)\)
Timed automata

Can be viewed:

- as the timed word \((\text{problem}, 23)(\text{delayed}, 38.6)(\text{repair}, 40.9)(\text{done}, 63)\)

- as the signal
Basic result on timed automata

**Theorem**

The reachability problem is decidable (and PSPACE-complete) for timed automata [AD94].

Theorem

The reachability problem is decidable (and $\text{PSPACE}$-complete) for timed automata [AD94].

Basic result on timed automata

**Theorem**

The reachability problem is decidable (and \texttt{PSPACE}-complete) for timed automata [AD94].

It can be extended to model-check TCTL [ACD93].


Outline

1. Introduction

2. Definition of the logics

3. The timed automaton model

4. The model-checking problem

5. Some interesting fragments

6. Conclusion
Back to the model-checking problem

system:

property:

\[ G(\text{request} \rightarrow F\text{grant}) \]

model-checking algorithm

yes/no
Back to the model-checking problem

system:

property:

G (request → F grant)

model-checking algorithm

yes/no
Back to the model-checking problem

system:

property:

G(request → F grant)

model-checking algorithm

yes/no
## Results

### Theorem

Over finite runs, the model-checking problem is:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MTL</td>
<td>decidable, NPR [OW05]</td>
<td>undecidable [AFH96]</td>
</tr>
<tr>
<td>MTL+Past</td>
<td>undecidable</td>
<td>undecidable</td>
</tr>
<tr>
<td>TPTL</td>
<td>undecidable [AH94]</td>
<td>undecidable [AH94]</td>
</tr>
</tbody>
</table>

---

[OW05] Ouaknine, Worrell. On the decidability of metric temporal logic (LICS’05).
## Results

### Theorem

Over finite runs, the model-checking problem is:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MTL</td>
<td>decidable, NPR [OW05]</td>
<td>undecidable [AFH96]</td>
</tr>
<tr>
<td>MTL+Past</td>
<td>undecidable</td>
<td>undecidable</td>
</tr>
<tr>
<td>TPTL</td>
<td>undecidable [AH94]</td>
<td>undecidable [AH94]</td>
</tr>
</tbody>
</table>

- Model-checking linear-time timed temporal logics is **hard**!
## Results

### Theorem

Over finite runs, the model-checking problem is:

<table>
<thead>
<tr>
<th></th>
<th>pointwise sem.</th>
<th>continuous sem.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTL</td>
<td>decidable, NPR [OW05]</td>
<td>undecidable [AFH96]</td>
</tr>
<tr>
<td>MTL+Past</td>
<td>undecidable</td>
<td>undecidable</td>
</tr>
<tr>
<td>TPTL</td>
<td>undecidable [AH94]</td>
<td>undecidable [AH94]</td>
</tr>
</tbody>
</table>

- Model-checking linear-time timed temporal logics is **hard**!
- The gap between branching-time and linear-time dramatically increases in the timed framework...
  
  (reminder: model-checking TCTL is **PSPACE**-complete)

---

[OW05] Ouaknine, Worrell. On the decidability of metric temporal logic (LICS’05).


The model-checking problem

Results

Theorem

Over finite runs, the model-checking problem is:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MTL</td>
<td>Decidable, NPR [OW05]</td>
<td>Undecidable [AFH96]</td>
</tr>
<tr>
<td>MTL+Past</td>
<td>Undecidable</td>
<td>Undecidable</td>
</tr>
<tr>
<td>TPTL</td>
<td>Undecidable [AH94]</td>
<td>Undecidable [AH94]</td>
</tr>
</tbody>
</table>

- Model-checking linear-time timed temporal logics is hard!
- The gap between branching-time and linear-time dramatically increases in the timed framework...
  (reminder: model-checking TCTL is PSPACE-complete)

we will explain this high complexity, following [Che07]

[OW05] Ouaknine, Worrell. On the decidability of metric temporal logic (LICS’05).
A short visit to channel machines (1)

A channel machine = a finite automaton + a FIFO channel

\[ \text{\(s_1\)} \rightarrow \text{\(s_2\)} \rightarrow \text{\(s_3\)} \rightarrow \text{\(s_4\)} \rightarrow \text{\(s_5\)} \]

\[ \bullet \text{!} \rightarrow \bullet \text{!} \rightarrow \bullet \text{?} \rightarrow \bullet \text{?} \rightarrow \bullet \text{!} \rightarrow \bullet \text{?} \]

\[ \text{\(s_5\)} \text{is not reachable} \]

\[ \text{\(s_5\)} \text{is reachable} \]
A short visit to channel machines (1)

A channel machine = a finite automaton + a FIFO channel
A short visit to channel machines (1)

A channel machine $= \text{a finite automaton} + \text{a FIFO channel}$

$\begin{align*}
S_1 & \xrightarrow{!} S_2 \\
S_2 & \xrightarrow{!} S_3 \\
S_3 & \xrightarrow{?} S_4 \\
S_4 & \xrightarrow{?} S_5
\end{align*}$
A short visit to channel machines (1)

A channel machine = a finite automaton + a FIFO channel

The model-checking problem
A short visit to channel machines (1)

A channel machine = a finite automaton + a FIFO channel

![Diagram of a channel machine](image)
A short visit to channel machines (1)

A channel machine = a finite automaton + a FIFO channel
A short visit to channel machines (1)

A channel machine = a finite automaton + a FIFO channel
A short visit to channel machines (1)

A channel machine = a finite automaton + a FIFO channel
A short visit to channel machines (1)

A channel machine = a finite automaton + a FIFO channel

\[ s_5 \text{ is not reachable} \]
A short visit to channel machines (1)

A channel machine = a finite automaton + a FIFO channel

- insertion errors: any letter can appear on the channel at any time
A short visit to channel machines (1)

A channel machine = a finite automaton + a FIFO channel

- insertion errors: any letter can appear on the channel at any time
A short visit to channel machines (1)

A channel machine = a finite automaton + a FIFO channel

- insertion errors: any letter can appear on the channel at any time
  - $s_5$ is reachable
A short visit to channel machines (2)

**Halting problem:** is there an execution ending in a halting state?
Halting problem: is there an execution ending in a halting state?

Proposition
- The halting problem is undecidable for channel machines [BZ83].
- The halting problem is NPR for channel machines with insertion errors [Sch02].

[Sch02] Schnoebelen. Verifying lossy channel systems has non-primitive recursive complexity (IPL, 2002).
Channel machines and timed words

We encode an execution of a channel machine as a timed word:

\[(q_0, \varepsilon) \xrightarrow{a!} (q_1, a) \xrightarrow{b!} (q_2, ab) \xrightarrow{a?} (q_3, b) \xrightarrow{c!} (q_4, bc) \xrightarrow{b?} (q_5, c) \cdots\]
We encode an execution of a channel machine as a timed word:

\[(q_0, \varepsilon) \xrightarrow{a!} (q_1, a) \xrightarrow{b!} (q_2, ab) \xrightarrow{a?} (q_3, b) \xrightarrow{c!} (q_4, bc) \xrightarrow{b?} (q_5, c) \cdots\]
Channel machines and timed words

We encode an execution of a channel machine as a timed word:

\[(q_0, \varepsilon) \xrightarrow{a!} (q_1, a) \xrightarrow{b!} (q_2, ab) \xrightarrow{a?} (q_3, b) \xrightarrow{c!} (q_4, bc) \xrightarrow{b?} (q_5, c) \cdots\]
Channel machines and timed words

We encode an execution of a channel machine as a timed word:

\[(q_0, \varepsilon) \xrightarrow{a!} (q_1, a) \xrightarrow{b!} (q_2, ab) \xrightarrow{a?} (q_3, b) \xrightarrow{c!} (q_4, bc) \xrightarrow{b?} (q_5, c) \cdots\]

We will give a formula \(\varphi\) such that

the channel machine* halts iff the formula \(\varphi\) is satisfiable

* possibly with insertion errors
Channel machines and timed words

We encode an execution of a channel machine as a timed word:

\[(q_0, \varepsilon) \xrightarrow{a!} (q_1, a) \xrightarrow{b!} (q_2, ab) \xrightarrow{a?} (q_3, b) \xrightarrow{c!} (q_4, bc) \xrightarrow{b?} (q_5, c) \cdots\]

We will give a formula \(\varphi\) such that

the channel machine* halts iff the formula \(\varphi\) is satisfiable
iff \(A_{\text{univ}} \not\models \neg \varphi\)

* possibly with insertion errors
Constraints satisfied by the timed word

- states and actions alternate, and the sequence satisfies the rules of the channel machine: LTL formula

\[G(a! \rightarrow F\underline{t.u.})=1\]

This formula is not sufficient!

\[\begin{align*}
q_0 & \rightarrow a! q_1 \\
q_1 & \rightarrow b! q_2 \\
a? & q_3 \\
c? & q_4 \\
b? & q_5 \\
\end{align*}\]

only encodes a channel machine with insertion errors!

Model-checking MTL is NPR.
Constraints satisfied by the timed word

- states and actions alternate, and the sequence satisfies the rules of the channel machine: LTL formula
- the channel is FIFO: for every letter $a$
  \[ \text{G} (a! \rightarrow \text{F}_{\geq 1} a?) \]
Constraints satisfied by the timed word

- states and actions alternate, and the sequence satisfies the rules of the channel machine: LTL formula
- the channel is FIFO: for every letter $a$,

\[ G(a! \rightarrow F_{\geq 1} a?) \]

This formula is not sufficient!
Constraints satisfied by the timed word

- states and actions alternate, and the sequence satisfies the rules of the channel machine: LTL formula
- the channel is FIFO: for every letter $a$,

$$G(a! \rightarrow F_{\geq 1} a?)$$

This formula is not sufficient!
Constraints satisfied by the timed word

- states and actions alternate, and the sequence satisfies the rules of the channel machine: LTL formula
- the channel is FIFO: for every letter $a$,

$$G(a! \rightarrow F_{=1} a?)$$

This formula is not sufficient!

only encodes a channel machine with insertion errors!
Constraints satisfied by the timed word

- states and actions alternate, and the sequence satisfies the rules of the channel machine: **LTL formula**
- the channel is FIFO: for every letter $a$,
  \[ G(a! \rightarrow F_{\geq 1} a?) \]

This formula is not sufficient!

\[ q_0 \xrightarrow{a!} q_1 \xrightarrow{b!} q_2 \xrightarrow{a?} q_3 \xrightarrow{c?} q_4 \xrightarrow{b?} q_5 \ldots \]

\[ \text{only encodes a channel machine with insertion errors!} \]

\[ \text{model-checking MTL is NPR} \]
We need to express the property:

“Every \(a?\)-event is preceded one time unit earlier by an \(a!\)-event”
We need to express the property:

“Every $a?$-event is preceded one time unit earlier by an $a!$-event”

- Why not reverse the previous implication?
  \[ G \left( (\text{F}_{=1} a?) \rightarrow a! \right) \]
We need to express the property:

“Every $a?$-event is preceded one time unit earlier by an $a!$-event”

- Why not reverse the previous implication?
  
  $G ( (F_{=1} a?) \rightarrow a! )$

  - correct in the continuous semantics
We need to express the property:

“Every $a?$-event is preceded one time unit earlier by an $a!$-event”

- Why not reverse the previous implication?
  \[
  G \left( (F_{\leq 1} a?) \rightarrow a! \right)
  \]
  - correct in the continuous semantics
  - not correct in the pointwise semantics
We need to express the property:

“Every \( a? \)-event is preceded one time unit earlier by an \( a! \)-event”

- Why not reverse the previous implication?
  \[
  G \left( (F_{=1} a?) \rightarrow a! \right)
  \]
  - correct in the continuous semantics
  - not correct in the pointwise semantics

- Why not look back in the past?
  \[
  G \left( a? \rightarrow F_{=1}^{-1} a! \right)
  \]
We need to express the property:

“Every $a?$-event is preceded one time unit earlier by an $a!$-event”

- Why not reverse the previous implication?
  \[ \mathsf{G} \left( (\mathsf{F}_{1} a?) \rightarrow a! \right) \]
  - correct in the continuous semantics
  - not correct in the pointwise semantics

- Why not look back in the past?
  \[ \mathsf{G} \left( a? \rightarrow \mathsf{F}_{-1} a! \right) \]
  - correct for $\mathsf{MTL} + \mathsf{Past}$ (in the continuous and in the pointwise sem.)
We need to express the property:

“Every $a?$-event is preceded one time unit earlier by an $a!$-event”

▶ Why not reverse the previous implication?

$$G \left( (F_{=1} a?) \rightarrow a! \right)$$

▶ correct in the continuous semantics
▶ not correct in the pointwise semantics

▶ Why not look back in the past?

$$G \left( a? \rightarrow F^{-1}_{=1} a! \right)$$

▶ correct for MTL+Past (in the continuous and in the pointwise sem.)
▶ no direct translation into MTL
We need to express the property:

“Every \( a? \)-event is preceded one time unit earlier by an \( a! \)-event”

- Why not reverse the previous implication?
  \[
  \mathsf{G} ( ( \mathsf{F}_{=1} a? ) \rightarrow a! )
  \]
  - correct in the continuous semantics
  - not correct in the pointwise semantics

- Why not look back in the past?
  \[
  \mathsf{G} ( a? \rightarrow \mathsf{F}_{=1}^{-1} a! )
  \]
  - correct for \( \mathsf{MTL} + \mathsf{Past} \) (in the continuous and in the pointwise sem.)
  - no direct translation into \( \mathsf{MTL} \)

- A more tricky way:
  \[
  \neg \left( \mathsf{F} x. \mathsf{X} y. \mathsf{F} ( x > 1 \land y < 1 \land c?) \right)
  \]
We need to express the property:

“Every $a?$-event is preceded one time unit earlier by an $a!$-event”

- Why not reverse the previous implication?
  \[
  G \left( (F_{=1} a?) \rightarrow a! \right)
  \]
  - correct in the continuous semantics
  - not correct in the pointwise semantics

- Why not look back in the past?
  \[
  G \left( a? \rightarrow F_{=1}^{-1} a! \right)
  \]
  - correct for MTL+Past (in the continuous and in the pointwise sem.)
  - no direct translation into MTL

- A more tricky way:
  \[
  \neg \left( F_x.X_y.F \left( x > 1 \land y < 1 \land c? \right) \right)
  \]

\[
\begin{array}{cccccccc}
q_0 & a! & q_1 & b! & q_2 & a? & q_3 & c? & q_4 & b? & q_5 & \ldots \\
\hline
\end{array}
\]

=1 t.u.
We need to express the property:

“Every $a?$-event is preceded one time unit earlier by an $a!$-event”

- Why not reverse the previous implication?
  \[ G((F_{=1} a?) \rightarrow a!) \]
  - correct in the continuous semantics
  - not correct in the pointwise semantics

- Why not look back in the past?
  \[ G(a? \rightarrow F_{=1}^{-1} a!) \]
  - correct for MTL+Past (in the continuous and in the pointwise sem.)
  - no direct translation into MTL

- A more tricky way:
  \[ \neg\left(F x. X y. F (x > 1 \land y < 1 \land c?)\right) \]

- this formula is in TPTL (pointwise sem.), not in MTL
What we have proved so far

**Theorem**

Over finite runs, the model-checking problem is:

<table>
<thead>
<tr>
<th></th>
<th>pointwise sem.</th>
<th>continuous sem.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTL</td>
<td>NPR [OW07]</td>
<td>undecidable [AFH96]</td>
</tr>
<tr>
<td>MTL+Past</td>
<td>undecidable</td>
<td>undecidable</td>
</tr>
<tr>
<td>TPTL</td>
<td>undecidable [AH94]</td>
<td>undecidable [AH94]</td>
</tr>
</tbody>
</table>
The model-checking problem

What remains to be proved

<table>
<thead>
<tr>
<th></th>
<th>pointwise sem.</th>
<th>continuous sem.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MTL</strong></td>
<td>decidable, NPR [OW07]</td>
<td>undecidable [AFH96]</td>
</tr>
<tr>
<td><strong>MTL+Past</strong></td>
<td>undecidable</td>
<td>undecidable</td>
</tr>
<tr>
<td><strong>TPTL</strong></td>
<td>undecidable [AH94]</td>
<td>undecidable [AH94]</td>
</tr>
</tbody>
</table>
From LTL to alternating automata

LTL formulas can be turned into linear alternating ( Büchi ) automata

$$G ( a \rightarrow F b )$$
From LTL to alternating automata

LTL formulas can be turned into linear alternating (Büchi) automata

$$G (a \rightarrow F b)$$
From LTL to alternating automata

LTL formulas can be turned into linear alternating (Büchi) automata

\[ G(a \rightarrow F b) \]
From LTL to alternating automata

LTL formulas can be turned into linear alternating (Büchi) automata

\[ G (a \rightarrow F b) \]
From MTL to alternating timed automata

MTL formulas can be turned into linear alternating timed automata

\[ G(a \rightarrow F_{[1,2]} b) \]
From MTL to alternating timed automata

MTL formulas can be turned into linear alternating timed automata

\[ G(a \rightarrow F_{[1,2]} b) \]
From MTL to alternating timed automata

MTL formulas can be turned into linear alternating timed automata

\[ G(a \rightarrow F_{[1,2]} b) \]
An abstract transition system

We order elements in a slice of the tree w.r.t. their fractional part, and we forget the precise values of the fractional parts. This defines an abstract (infinite) transition system that is (time-abstract) bisimilar to the transition system of the alternating timed automata.
An abstract transition system

We order elements in a slice of the tree w.r.t. their fractional part, and we forget the precise values of the fractional parts.
An abstract transition system

We order elements in a slice of the tree w.r.t. their fractional part, and we forget the precise values of the fractional parts.

This defines an abstract (infinite) transition system.
An abstract transition system

We order elements in a slice of the tree w.r.t. their fractional part, and we forget the precise values of the fractional parts.

- this defines an abstract (infinite) transition system
- it is (time-abstract) bisimilar to the transition system of the alternating timed automata
An abstract transition system

we order elements in a slice of the tree w.r.t. their fractional part, and we forget the precise values of the fractional parts.

this defines an abstract (infinite) transition system.

it is (time-abstract) bisimilar to the transition system of the alternating timed automata.

there is a well quasi-order on the set of abstract configurations (subword relation):

\[ \text{higman} \sqsubseteq \text{highmountain} \]
Summary

The model-checking problem

Over finite runs, the model-checking problem is:

<table>
<thead>
<tr>
<th></th>
<th>pointwise sem.</th>
<th>continuous sem.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTL</td>
<td>decidable, NPR [OW05]</td>
<td>undecidable [AFH96]</td>
</tr>
<tr>
<td>MTL+Past</td>
<td>undecidable</td>
<td>undecidable</td>
</tr>
<tr>
<td>TPTL</td>
<td>undecidable [AH94]</td>
<td>undecidable [AH94]</td>
</tr>
</tbody>
</table>
What about infinite behaviours?

- the previous algorithm cannot be lifted to the infinite behaviours framework
What about infinite behaviours?

- the previous algorithm cannot be lifted to the infinite behaviours framework
- there is a problem with the accepting condition
  (in the untimed case, we use the Miyano-Hayashi construction [MH84])
What about infinite behaviours?

- the previous algorithm cannot be lifted to the infinite behaviours framework
- there is a problem with the accepting condition
  (in the untimed case, we use the Miyano-Hayashi construction [MH84])

**Theorem**

Over finite runs, the model-checking problem is:

<table>
<thead>
<tr>
<th>Logic</th>
<th>Pointwise sem.</th>
<th>Continuous sem.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTL</td>
<td>undecidable [OW06]*</td>
<td>undecidable [AFH96]</td>
</tr>
<tr>
<td>MTL+Past</td>
<td>undecidable</td>
<td>undecidable</td>
</tr>
<tr>
<td>TPTL</td>
<td>undecidable [AH94]</td>
<td>undecidable [AH94]</td>
</tr>
</tbody>
</table>

* by reduction of the recurrence problem for channel machines

---

[OW06] Ouaknine, Worrell. On metric temporal logic and faulty Turing machines (FoSSaCS’06).
Outline

1. Introduction

2. Definition of the logics

3. The timed automaton model

4. The model-checking problem

5. Some interesting fragments

6. Conclusion
The fragment without punctuality

- The undecidability/NPR proofs heavily rely on punctual constraints.

Old claim: 
"Any logic strong enough to express the property $G(\cdot \rightarrow F)=1$ is undecidable"

What if we forbid punctual constraints in MTL?

Metric Interval Temporal Logic (MITL): $\text{MITL} \ni \phi::=a | \neg \phi | \phi \lor \phi | \phi \land \phi | \phi U I \phi$ with $I$ a non-punctual interval

Examples:
- $G(\cdot \rightarrow F)=1$ is not in MITL
- $G(\cdot \rightarrow F[1,2])$ is in MITL

The fragment without punctuality

- The undecidability/NPR proofs heavily rely on punctual constraints.

**Old claim:** “Any logic strong enough to express the property $G(\bullet \rightarrow F_{\geq 1} \bullet)$ is undecidable”

- Metric Interval Temporal Logic (MITL):
  
  $\text{MITL} \ni \phi ::= a | \neg \phi | \phi \lor \phi | \phi \land \phi | \phi U I \phi$ with $I$ a non-punctual interval

- Examples:
  
  $G(\bullet \rightarrow F_{\geq 1} \bullet)$ is not in MITL
  
  $G(\bullet \rightarrow F_{[1,2]} \bullet)$ is in MITL
The fragment without punctuality

- The undecidability/NPR proofs heavily rely on punctual constraints.

  **Old claim:** “Any logic strong enough to express the property \( G(\bullet \rightarrow F_{\geq 1} \bullet) \) is undecidable”

- What if we forbid punctual constraints in MTL?

**Ref: \[AFH96\] Alur, Feder, Henzinger. The benefits of relaxing punctuality (Journal of the ACM, 1996).**
Some interesting fragments

The fragment without punctuality

- The undecidability/NPR proofs heavily rely on punctual constraints.

  **Old claim:** “Any logic strong enough to express the property $G(a \rightarrow F_{\geq 1}a)$ is undecidable”

- What if we forbid punctual constraints in MTL?

**Metric Interval Temporal Logic (MITL):**

$$\text{MITL} \ni \varphi ::= a \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \varphi \mathcal{U} \ell \varphi$$

with $\ell$ a non-punctual interval

The fragment without punctuality

- The undecidability/NPR proofs heavily rely on punctual constraints.

  **Old claim:** “Any logic strong enough to express the property \( G (\bullet \rightarrow F_{=1} \bullet) \) is undecidable”

- What if we forbid punctual constraints in MTL?

Metric Interval Temporal Logic (MITL): [AFH96]

\[
\text{MITL} \ni \varphi ::= a | \neg \varphi | \varphi \lor \varphi | \varphi \land \varphi | \varphi U_{/} I \varphi
\]

with / a non-punctual interval

- Examples:
  - \( G (\bullet \rightarrow F_{=1} \bullet) \) is not in MITL

The fragment without punctuality

► The undecidability/NPR proofs heavily rely on punctual constraints.

**Old claim:** “Any logic strong enough to express the property $G(\bullet \rightarrow F_{\leq 1} \bullet)$ is undecidable”

► What if we forbid punctual constraints in MTL?

**Metric Interval Temporal Logic (MITL):**

$\exists \varphi ::= a \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \varphi U_{/} \varphi$

with $/ \,$ a non-punctual interval

► Examples:

► $G(\bullet \rightarrow F_{= 1} \bullet)$ is not in MITL

► $G(\bullet \rightarrow F_{[1,2]} \bullet)$ is in MITL

Model-checking MITL is “easy”

**Theorem**

The model-checking problem for MITL is EXPSPACE-complete [AFH96]. If constants are encoded in unary, it is even PSPACE-complete [HR04].
Model-checking MITL is “easy”

**Theorem**

The model-checking problem for MITL is \textsc{ExpSpace}-complete \cite{AFH96}. If constants are encoded in unary, it is even \textsc{PSpace}-complete \cite{HR04}.

we can bound the variability of the signals
Model-checking MITL is “easy”

Theorem

The model-checking problem for MITL is \textit{EXPSPACE}-complete \cite{AFH96}. If constants are encoded in unary, it is even \textit{PSPACE}-complete \cite{HR04}.

\begin{itemize}
  \item we can bound the \textit{variability} of the signals
  \item an MITL formula defines a timed regular language
\end{itemize}

\textbf{Example:} consider the formula $\varphi = G_{(0,1)} (\bullet \rightarrow F_{[1,2]} \bullet)$
Model-checking MITL is “easy”

**Theorem**

The model-checking problem for MITL is EXPSPACE-complete [AFH96]. If constants are encoded in unary, it is even PSPACE-complete [HR04].

- we can bound the variability of the signals
- an MITL formula defines a timed regular language

**Example:** consider the formula $\varphi = G_{(0,1)} (\bullet \rightarrow F_{[1,2]} \bullet)$

> each time an $\bullet$ occurs within the first time unit, start a new clock, and check that a $\bullet$ occurs between 1 and 2 time units afterwards
Model-checking MITL is “easy”

Theorem

The model-checking problem for MITL is \textsc{ExpSpace}-complete \cite{AFH96}. If constants are encoded in unary, it is even \textsc{PSPACE}-complete \cite{HR04}.

- we can bound the \textit{variability} of the signals
- an MITL formula defines a timed regular language

Example: consider the formula \( \varphi = G_{(0,1)} (\bullet \rightarrow F_{[1,2]} \bullet) \)

- each time an \( \bullet \) occurs within the first time unit, start a new clock, and check that a \( \bullet \) occurs between 1 and 2 time units afterwards
- this requires an unbounded number of clocks
Model-checking MITL is “easy”

**Theorem**

The model-checking problem for MITL is $\text{EXPSPACE}$-complete [AFH96]. If constants are encoded in unary, it is even $\text{PSPACE}$-complete [HR04].

- we can bound the **variability** of the signals
- an MITL formula defines a timed regular language

**Example:** consider the formula $\varphi = G_{(0,1)}(\bullet \rightarrow F_{[1,2]} \bullet)$

- each time an $\bullet$ occurs within the first time unit, start a new clock, and check that a $\bullet$ occurs between 1 and 2 time units afterwards
- this requires an unbounded number of clocks

[HR04] Hirshfeld, Rabinovich. Logics for real time: decidability and complexity (Fundamenta Informaticae, 2004).
\( \varphi = G_{(0,1)}(a \rightarrow F_{[1,2]} b) \)
Some interesting fragments

\[ \varphi = G_{(0,1)} (a \rightarrow F_{[1,2]} b) \]
Some interesting fragments

\[ \varphi = \mathbf{G}_{(0,1)}(a \rightarrow \mathbf{F}_{[1,2]} b) \]
\[ \varphi = G_{(0,1)}(a \rightarrow F_{[1,2]} b) \]
Some interesting fragments

\[ \varphi = G_{(0,1)} (a \rightarrow F_{[1,2]} b) \]

This idea can be extended to any formula in MITL
A co-flat fragment of MTL

- Do punctual constraints really need to be banned?

\[
\text{coFlat-MTL} \ni \varphi ::= a \mid \neg a \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \varphi U I \psi \mid \psi \not\text{U} I \varphi
\]

where \( I \) unbounded \( \Rightarrow \psi \in \text{LTL} \)

Examples:

- \( G (\bullet \rightarrow F \cdot 1 \bullet) \) is in coFlat-MTL
- \( FG \leq 1 \cdot \) is not in coFlat-MTL
- coFlat-MTL contains Bounded-MTL (all modalities are time-bounded)

A co-flat fragment of MTL

- Do punctual constraints really need to be banned?
- Does punctuality always lead to undecidability?
A co-flat fragment of MTL

- Do punctual constraints really need to be banned?
- Does punctuality always lead to undecidability?

We define `coFlat-MTL`:

\[
\text{coFlat-MTL} \ni \varphi ::= a \mid \neg a \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \varphi \mathsf{U}_I \psi \mid \psi \mathsf{U}_I \varphi
\]

where $I$ unbounded $\Rightarrow \psi \in \text{LTL}$

A co-flat fragment of MTL

▶ Do punctual constraints really need to be banned?
▶ Does punctuality always lead to undecidability?

We define $\text{coFlat-MTL}$:

\[
\text{coFlat-MTL} \ni \varphi ::= a \mid \neg a \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \varphi \mathbf{U}_I \psi \mid \psi \mathbf{\tilde{U}}_I \varphi
\]

where $I$ unbounded $\Rightarrow \psi \in \text{LTL}$

▶ Examples:
  ▶ $G \left( \bullet \rightarrow F_{=1} \bullet \right)$ is in $\text{coFlat-MTL}$

A co-flat fragment of MTL

- Do punctual constraints really need to be banned?
- Does punctuality always lead to undecidability?

We define coFlat-MTL:

\[ \text{coFlat-MTL} \ni \varphi ::= a \mid \neg a \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \varphi U_I \psi \mid \psi \tilde{U}_I \varphi \]

where \( I \) unbounded \( \Rightarrow \) \( \psi \in \text{LTL} \)

- Examples:
  - \( G (\bullet \rightarrow F_{\leq 1} \bullet) \) is in coFlat-MTL
  - \( F G_{\leq 1} \bullet \) is not in coFlat-MTL

A co-flat fragment of MTL

- Do punctual constraints really need to be banned?
- Does punctuality always lead to undecidability?

We define \( \text{coFlat-MTL} \):

\[
\text{coFlat-MTL} \ni \varphi ::= a \mid \neg a \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \varphi \mathbf{U}_I \psi \mid \psi \mathbf{\tilde{U}}_I \varphi
\]

where \( I \) unbounded \( \Rightarrow \psi \in \text{LTL} \)

- Examples:
  - \( \mathbf{G} (\bullet \rightarrow \mathbf{F}_{=1} \bullet) \) is in \( \text{coFlat-MTL} \)
  - \( \mathbf{F} \mathbf{G}_{\leq 1} \bullet \) is not in \( \text{coFlat-MTL} \)
  - \( \text{coFlat-MTL} \) contains \( \text{Bounded-MTL} \) (all modalities are time-bounded)

\[\text{[BMOW07]}\] Bouyer, Markey, Ouaknine, Worrell. The cost of punctuality (LICS'07).
Model-checking coFlat-MTL is “easy”

Theorem
The model-checking problem for coFlat-MTL or Bounded-MTL is EXPSPACE-complete [BMOW07]. If constants are encoded in unary, the model-checking of Bounded-MTL is PSPACE-complete.
Some interesting fragments

Model-checking coFlat-MTL is “easy”

Theorem

The model-checking problem for coFlat-MTL or Bounded-MTL is EXPSPACE-complete [BMOW07]. If constants are encoded in unary, the model-checking of Bounded-MTL is PSPACE-complete.

▷ The variability of a Bounded-MTL formula can be high (doubly-exp.):

\[ \varphi_n \equiv \bullet \land G_{[0,2^n]} \varphi_D \]

with

\[ \varphi_D = \left( \cdot \rightarrow F_{=1} (\bullet \land F_{\leq 1} \bullet) \right) \]

\[ \land \left( \bullet \rightarrow F_{=1} (\bullet \land F_{\leq 1} \bullet) \right) \]

36/40
Model-checking coFlat-MTL is “easy”

Theorem

The model-checking problem for coFlat-MTL or Bounded-MTL is EXPSPACE-complete [BMOW07]. If constants are encoded in unary, the model-checking of Bounded-MTL is PSPACE-complete.

- The variability of a Bounded-MTL formula can be high (doubly-exp.):

\[ \varphi_n \equiv \bullet \land G_{[0,2^n]} \varphi_D \]

with

\[ \varphi_D = (\bullet \rightarrow F=1 (\bullet \land F_{\leq 1} \bullet)) \land (\bullet \rightarrow F=1 (\bullet \land F_{\leq 1} \bullet)) \]

---

A Bounded-MTL formula may define a non timed-regular language:

\[ G \leq 1 (\bullet \rightarrow F=1 (\bullet \land F_{\leq 1} \bullet)) \land G \leq 1 (\bullet \rightarrow F=1 (\bullet \land F_{\leq 1} \bullet)) \]

defines the context-free language \{n \cdot m | n \leq m\}.
Model-checking coFlat-MTL is “easy”

Theorem

The model-checking problem for coFlat-MTL or Bounded-MTL is EXPSPACE-complete [BMOW07]. If constants are encoded in unary, the model-checking of Bounded-MTL is PSPACE-complete.

▶ The variability of a Bounded-MTL formula can be high (doubly-exp.):

\[ \varphi_n \equiv \bullet \land G_{[0,2^n]} \varphi_D \]

with

\[ \varphi_D = (\bullet \rightarrow F_{=1} (\bullet \land F_{\leq 1} \bullet)) \land (\bullet \rightarrow F_{=1} (\bullet \land F_{\leq 1} \bullet)) \]

\[ \cdots \]
Model-checking coFlat-MTL is “easy”

Theorem

The model-checking problem for coFlat-MTL or Bounded-MTL is EXPSPACE-complete [BMOW07]. If constants are encoded in unary, the model-checking of Bounded-MTL is PSPACE-complete.

- The variability of a Bounded-MTL formula can be high (doubly-exp.):

\[ \varphi_n \equiv \bullet \land G_{[0,2^n]} \varphi_D \quad \text{with} \quad \varphi_D = \left( \bullet \rightarrow F_{=1} (\bullet \land F_{\leq 1} \bullet) \right) \land \left( \bullet \rightarrow F_{=1} (\bullet \land F_{\leq 1} \bullet) \right) \]
Theorem

The model-checking problem for coFlat-MTL or Bounded-MTL is \(\text{EXPSPACE}\)-complete [BMOW07]. If constants are encoded in unary, the model-checking of Bounded-MTL is \(\text{PSPACE}\)-complete.

The variability of a Bounded-MTL formula can be high (doubly-exp.):

\[
\varphi_n \equiv \bullet \land G_{[0,2^n]} \varphi_D \quad \text{with} \quad \varphi_D = \left( \bullet \rightarrow F_{\leq 1} (\bullet \land F_{\leq 1} \bullet) \right) \land \left( \bullet \rightarrow F_{= 1} (\bullet \land F_{\leq 1} \bullet) \right)
\]
Model-checking coFlat-MTL is “easy”

Theorem

The model-checking problem for coFlat-MTL or Bounded-MTL is EXPSPACE-complete [BMOW07]. If constants are encoded in unary, the model-checking of Bounded-MTL is PSPACE-complete.

▶ The variability of a Bounded-MTL formula can be high (doubly-exp.):

$$\varphi_n \equiv \bullet \land G_{[0, 2^n]} \varphi_D$$

with

$$\varphi_D = (\bullet \rightarrow F=1 (\bullet \land F \leq 1 \bullet)) \land (\bullet \rightarrow F=1 (\bullet \land F \leq 1 \bullet))$$

▶ A Bounded-MTL formula may define a non timed-regular language:

$$G_{\leq 1} (\bullet \rightarrow F=1 \bullet) \land G_{\leq 1} \bullet \land G_{(1, 2]} \bullet$$

defines the context-free language $$\{ \bullet^n \bullet^m \mid n \leq m \}.$$
Algorithm for Bounded-MTL

Assume one wants to verify formula

\[ G_{<2}(\bullet \rightarrow F_{=1} \bullet) \]
Algorithm for Bounded-MTL

Assume one wants to verify formula

\[ G_{<2}\left( \bullet \rightarrow F_{=1} \bullet \right) \]
Algorithm for Bounded-MTL

Assume one wants to verify formula

$$G_{<2}(\cdot \rightarrow F_{=1} \cdot)$$

Offline, we stack all ‘relevant’ time units and use a sliding window:

3 t.u. = useful duration
Algorithm for Bounded-MTL

Assume one wants to verify formula

\[ G <_2 (\bullet \rightarrow F =_1 \bullet) \]

Offline, we stack all ‘relevant’ time units and use a sliding window:

3 t.u. = useful duration
Algorithm for Bounded-MTL

Assume one wants to verify formula

\[ G_{<2}\left( \bullet \rightarrow F_{=1} \bullet \right) \]

Offline, we stack all ‘relevant’ time units and use a sliding window:

3 t.u. = useful duration
Algorithm for Bounded-MTL

Assume one wants to verify formula

\[ G_{<2}(\bullet \rightarrow F_{=1} \bullet) \]

Offline, we stack all ‘relevant’ time units and use a sliding window:

3 t.u. = useful duration
Algorithm for Bounded-MTL

Assume one wants to verify formula

\[ G_{<2}(\bullet \rightarrow F_{=1} \bullet) \]

Offline, we stack all ‘relevant’ time units and use a sliding window:

3 t.u. = useful duration
Algorithm for Bounded-MTL

Assume one wants to verify formula

\[ G_{<2} \left( \bullet \rightarrow F_{=1} \bullet \right) \]

Offline, we stack all ‘relevant’ time units and use a sliding window:

3 t.u. = useful duration
Algorithm for Bounded-MTL

Assume one wants to verify formula

\[ G_{< 2} \left( \bullet \rightarrow F_{=1} \bullet \right) \]

Offline, we stack all ‘relevant’ time units and use a sliding window:

3 t.u. = useful duration
Algorithm for Bounded-MTL

Assume one wants to verify formula

\[ G <_2 (\bullet \rightarrow F_=^1 \bullet) \]

Offline, we stack all ‘relevant’ time units and use a sliding window:

3 t.u. = useful duration
Algorithm for Bounded-MTL

Assume one wants to verify formula

$$G_{<2}(\cdot \rightarrow F_{=1} \cdot)$$

Offline, we stack all ‘relevant’ time units and use a sliding window:

3 t.u. = useful duration
Algorithm for coFlat-MTL

\( \varphi \mapsto \) alternating timed automata \( B_{\neg \varphi} \) for \( \neg \varphi \) with a ‘flatness’ property
Algorithm for coFlat-MTL

\( \varphi \Leftrightarrow \) alternating timed automata \( B_{\neg \varphi} \) for \( \neg \varphi \) with a ‘flatness’ property

active

pure LTL

active

pure LTL

active

pure LTL

active

pure LTL

where - the number of active fragments is at most exponential
- the total duration of active fragments is at most exponential
Algorithm for coFlat-MTL

\( \varphi \Rightarrow \) alternating timed automata \( B_{\neg \varphi} \) for \( \neg \varphi \) with a ‘flatness’ property

where - the number of active fragments is at most exponential
- the total duration of active fragments is at most exponential

- active fragment = cycle-bounded computation in a channel machine
- pure LTL part = finite automaton computation
Outline

1. Introduction

2. Definition of the logics

3. The timed automaton model

4. The model-checking problem

5. Some interesting fragments

6. Conclusion
Conclusion

- Recent advances have raised a new interest for **linear-time timed temporal logics**
  - Not everything is undecidable
  - Some rather ‘efficient’ subclasses
    - non-punctual formulas
    - structurally (co-)flat formulas

---

A recent result: coFlat-MTL unifies MITL and MITL, and is EXPSPACE-complete \[\text{[BMOW08]}\]

- No real data structures do exist for these logics.

- An interesting phenomenon (in the continuous semantics): TCTL, MTL, Bounded-TCTL, Bounded-MTL m.-c. PSPACE sat. undec.

\begin{align*}
\text{NPR/undec.} & \\
\text{PSPACE} & \\
\text{\textdagger} & \text{ongoing work with Jenkins, Ouaknine, Worrell.}\text{[BMOW08]}\end{align*}

- Bouyer, Markey, Ouaknine, Worrell. A small-model theorem for real-time logics.
Conclusion

- Recent advances have raised a new interest for linear-time timed temporal logics
  - Not everything is undecidable
  - Some rather ‘efficient’ subclasses
    - non-punctual formulas
    - structurally (co-)flat formulas
- A recent result: \(\text{coFlat-MTL}_{\text{MITL}}\) unifies \(\text{coFlat-MTL}\) and \(\text{MITL}\), and is EXPSPACE-complete \([\text{BMOW08}]\)!
Recent advances have raised a new interest for linear-time timed temporal logics

- Not everything is undecidable
- Some rather ‘efficient’ subclasses
  - non-punctual formulas
  - structurally (co-)flat formulas

A recent result: $\text{coFlat-MTL}_{\text{MITL}}$ unifies $\text{coFlat-MTL}$ and MITL, and is $\text{EXPSPACE}$-complete [BMOW08]!

No real data structures do exist for these logics.
Conclusion

- Recent advances have raised a new interest for linear-time timed temporal logics
  - Not everything is undecidable
  - Some rather ‘efficient’ subclasses
    - non-punctual formulas
    - structurally (co-)flat formulas
- A recent result: $\text{coFlat-MTL}_{\text{MITL}}$ unifies $\text{coFlat-MTL}$ and $\text{MITL}$, and is $\text{EXPSPACE}$-complete [BMOW08]!
- No real data structures do exist for these logics.
- An interesting phenomenon (in the continuous semantics):

<table>
<thead>
<tr>
<th></th>
<th>TCTL</th>
<th>MTL</th>
<th>Bounded-TCTL</th>
<th>Bounded-MTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>m.-c.</td>
<td>PSPACE</td>
<td>NPR/undec.</td>
<td>PSPACE</td>
<td>PSPACE</td>
</tr>
<tr>
<td>sat.</td>
<td>undec.</td>
<td>NPR/undec.</td>
<td>non-elem.*</td>
<td>PSPACE</td>
</tr>
</tbody>
</table>

* ongoing work with Jenkins, Ouaknine, Worrell.