Model-Checking Timed Temporal Logics

Patricia Bouyer

LSV – CNRS & ENS de Cachan – France

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:

property:

G (request → F grant)
Model-checking

system:

property:

G (request → F grant)
Model-checking

system:  \[
\begin{align*}
\text{request} & \rightarrow \text{grant} \\
\end{align*}
\]

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

model-checking algorithm

yes/no
The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

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

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

Linear-time temporal logic [Pnu77]

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

[Unsu77] Pnueli. The temporal logic of programs (FOCS’77).
The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

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

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

Linear-time temporal logic [Pnu77]

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

\[ \quad \quad \quad \quad \Downarrow \quad \quad X \bullet \]

\[ \quad \quad \quad \quad \Downarrow \quad \quad \bullet U \bullet \]

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

Linear-time temporal logic [Pnu77]

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

\[\begin{align*}
\begin{array}{c}
\rightarrow \quad \rightarrow \quad \rightarrow \quad \rightarrow \quad \rightarrow \\
\cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot
\end{array}
\end{align*}\]

\[\begin{align*}
\begin{array}{c}
\cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \\
\rightarrow \quad \rightarrow \quad \rightarrow \quad \rightarrow \quad \rightarrow \\
\cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot
\end{array}
\end{align*}\]

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

Linear-time temporal logic [Pnu77]

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

\[
\begin{array}{c}
\text{\textcolor{red}{\textbullet}} \quad \text{\textcolor{red}{\textbullet}} \quad \text{\textcolor{red}{\textbullet}} \quad \text{\textcolor{red}{\textbullet}} \quad \text{\textcolor{red}{\textbullet}} \quad \text{\textcolor{red}{\textbullet}} \quad \text{\textcolor{red}{\textbullet}} \quad \ldots \\
\text{\textcolor{cyan}{\textbullet}} \quad \text{\textcolor{cyan}{\textbullet}} \quad \text{\textcolor{cyan}{\textbullet}} \quad \text{\textcolor{cyan}{\textbullet}} \quad \text{\textcolor{cyan}{\textbullet}} \quad \text{\textcolor{cyan}{\textbullet}} \quad \text{\textcolor{cyan}{\textbullet}} \quad \ldots \\
\text{\textcolor{blue}{\textbullet}} \quad \text{\textcolor{blue}{\textbullet}} \quad \text{\textcolor{blue}{\textbullet}} \quad \text{\textcolor{blue}{\textbullet}} \quad \text{\textcolor{blue}{\textbullet}} \quad \text{\textcolor{blue}{\textbullet}} \quad \text{\textcolor{blue}{\textbullet}} \quad \ldots
\end{array}
\]

\[
\begin{array}{c}
\models X \cdot \\
\models \cdot U \cdot \\
\models F \cdot \equiv \text{tt} U \cdot
\end{array}
\]

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

Linear-time temporal logic [Pnu77]

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

\[
\begin{align*}
&= X \cdot \\
&= \cdot U \cdot \\
&= F \cdot \equiv \top U \cdot
\end{align*}
\]

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

Linear-time temporal logic [Pnu77]

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

\[
|\quad X \cdot |
\]

\[
|\quad \cdot \ U \cdot |
\]

\[
|\quad F \cdot \equiv \top \ U \cdot |
\]

\[
|\quad G \cdot \equiv \neg F \neg \cdot |
\]

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

Linear-time temporal logic [Pnu77]

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

response property:

\[
G (\bullet \rightarrow F \bullet)
\]
The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

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

\begin{itemize}
  \item response property:
    \[
    G (\bullet \rightarrow F \bullet)
    \]
  \item liveness property:
    \[
    G F \bullet
    \]
\end{itemize}

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

Linear-time temporal logic \[\text{[Pnu77]}\]

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

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

- liveness property:
  \[GF \bullet\]

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

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

Linear-time temporal logic \[\text{[Pnu77]}\]

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

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

- liveness property:
  \[
  GF \bullet
  \]

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

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

\[\text{[Pnu77]}\] 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)

MTL ∈ ϕ ::= a | ¬ϕ | ϕ \lor ϕ | ϕ \land ϕ | ϕ U_I ϕ

where I is an interval with integral bounds.

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)

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

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)

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

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

- 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:

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

MTL formulas are interpreted over timed words:

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

\(\implies\) the system is observed only when actions happen
The pointwise semantics

MTL formulas are interpreted over timed words:

\((\bullet, 0.6)(\bullet, 1.1)(\bullet, 1.2)(\bullet, 1.3)\ldots\)

\(\rightarrow\) the system is observed only when actions happen

\(\in [1, 2]\)

\(\models \quad U_{[1, 2]}\)
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:

\[ t \in [0, \cdot 6] \mapsto \square \]
\[ t \in (\cdot 6, \cdot 1) \mapsto \]  
\[ t \in [1\cdot 1, 1\cdot 2) \mapsto \]  

...
The continuous semantics

MTL formulas are interpreted over (finitely variable) signals:

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

... 

込 the system is observed continuously
The continuous semantics

MTL formulas are interpreted over (finitely variable) signals:

\[ t \in [0, 0.6] \mapsto \top \]
\[ t \in (0.6, 1.1) \mapsto \neg \top \]
\[ t \in [1.1, 1.2) \mapsto \bot \]

...the system is observed continuously

\[ t \in [1, 2] \]

\[ \models \top \cup [1, 2] \]
The continuous semantics

MTL formulas are interpreted over (finitely variable) signals:

- \( t \in [0, 0.6) \mapsto \checkmark \)
- \( t \in (0.6, 1.1) \mapsto \times \)
- \( t \in [1.1, 1.2) \mapsto \blacksquare \)

The system is observed continuously

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

\( \not\models \bullet G_{[2, 3]} \bullet \)
Some examples

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

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

- “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”
  \[G\left(\text{problem} \rightarrow (F_{\leq 15} \text{repair} \lor G_{[12,15)} \text{alarm})\right)\]
Some further extensions

- Timed Propositional Temporal Logic (TPTL)  

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

Some further extensions

- Timed Propositional Temporal Logic (TPTL) \[\text{[AH89]}\]

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

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

\[\text{[AH89]}\] Alur, Henzinger. A really temporal logic (FOCS’89).
Some further extensions

- Timed Propositional Temporal Logic (TPTL) \[\text{[AH89]}\]

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

\[
\begin{align*}
G(\text{problem} \rightarrow F_{\leq 56} \text{alarm}) & \equiv G(\text{problem} \rightarrow x.F(\text{alarm} \land x \leq 56)) \\
G(\text{problem} \rightarrow x.F(\text{alarm} \land F(\text{failsafe} \land x \leq 56)))
\end{align*}
\]

Some further extensions

- **Timed Propositional Temporal Logic (TPTL)**  
  \[ TPTL = LTL + \text{clock variables} + \text{clock constraints} \]

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

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

- **MTL+Past**: add past-time modalities  
  \[ [AH92] \]

Some further extensions

- Timed Propositional Temporal Logic (TPTL)  
  \[ TPTL = \text{LTL} + \text{clock variables} + \text{clock constraints} \]

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

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

- MTL+Past: add past-time modalities
  \[ G(\text{alarm} \rightarrow F_{\leq 56} \text{problem}) \]

A note on the expressiveness

**Theorem**

$LTL + \text{Past}$ 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**

\( \text{LTL+Past is as expressive as LTL} \) [Kam68, GPSS80].

**Theorem**

\( \text{MTL is strictly less expressive than MTL+Past and TPTL} \) [BCM05].

**Conjecture in 1990:** the TPTL formula

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

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

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

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

\[ G \bullet \rightarrow \begin{cases} F \leq 1 \land F \leq 1 & \text{for } x \in [1, 2] \\ F \leq 1 & \text{for } x \leq 1 \\ F \leq 1 & \text{for } x \geq 2 \end{cases} \]
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

\[ G (\bullet \rightarrow x. F (\bullet \land F (\bullet \land x \leq 2))) = F_{\leq 1} \land F_{[1,2]} \]
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
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
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

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

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

The timed automaton model

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

▶ as the signal
   safe → alarm → repairing → failsafe → repairing → safe
Timed automata

The timed automaton model

Can be viewed:

as the timed word
(problem, 23)(delayed, 38.6)(repair, 40.9)(done, 63)

as the signal

safe

alarm

failsafe

repairing

problem, x:=0

Repairing

repair, y=0

delayed, y:=0

15\leq x \leq 16

15\leq y \leq 25

2\leq y \wedge x \leq 56

x 0

y 0

safe

repair

done

delayed

reparation

safe

alarm

failsafe

repairing

problem, x:=0

Repairing

repair, y=0

delayed, y:=0

15\leq x \leq 16

15\leq y \leq 25

2\leq y \wedge x \leq 56

x 0

y 0

safe

repair

done

delayed

reparation

safe

alarm

failsafe

repairing

problem, x:=0

Repairing

repair, y=0

delayed, y:=0

15\leq x \leq 16

15\leq y \leq 25

2\leq y \wedge x \leq 56

x 0

y 0

safe

repair

done

delayed

reparation

safe

alarm

failsafe

repairing

problem, x:=0

Repairing

repair, y=0

delayed, y:=0

15\leq x \leq 16

15\leq y \leq 25

2\leq y \wedge x \leq 56

x 0

y 0

safe

repair

done

delayed

reparation

safe

alarm

failsafe

repairing

problem, x:=0

Repairing

repair, y=0

delayed, y:=0

15\leq x \leq 16

15\leq y \leq 25

2\leq y \wedge x \leq 56

x 0

y 0

safe

repair

done

delayed

reparation

safe
Timed automata

The timed automaton model
The timed automaton model

Timed automata

Can be viewed:

- as the timed word
- as the signal
The timed automaton model

**Timed automata**

The timed automaton model can be viewed:

- as the timed word \((\text{problem}, 23) (\text{delayed}, 38.6) (\text{repair}, 40.9) (\text{done}, 63.0)\)
- as the signal

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>23</td>
</tr>
<tr>
<td>0</td>
<td>23</td>
</tr>
<tr>
<td>0</td>
<td>15.6</td>
</tr>
<tr>
<td>23</td>
<td>38.6</td>
</tr>
</tbody>
</table>
The timed automaton model
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

\(x\)
\[
\begin{array}{c|c|c|c}
 & \text{safe} & \text{problem} & \text{alarm} \\
\hline
x & 0 & 23 & 0 \\
y & 0 & 23 & 23 \\
\end{array}
\]

\(y\)
\[
\begin{array}{c|c|c|c|c}
 & \text{safe} & \text{problem} & \text{alarm} & \text{delayed} & \text{failsafe} \\
\hline
x & 0 & 23 & 0 & 15.6 & 15.6 \\
y & 0 & 23 & 23 & 38.6 & 0 \\
\end{array}
\]

\(y\)
\[
\begin{array}{c|c|c|c}
 & \text{safe} & \text{repair} & \text{done} \\
\hline
x & 0 & 23 & 0 \\
y & 0 & 23 & 23 \\
\end{array}
\]

\(y\)
\[
\begin{array}{c|c|c|c|c}
 & \text{safe} & \text{repair} & \text{done} \\
\hline
x & 0 & 23 & 0 \\
y & 0 & 23 & 23 \\
\end{array}
\]

\(y\)
\[
\begin{array}{c|c|c|c}
 & \text{safe} & \text{repair} & \text{done} \\
\hline
x & 0 & 23 & 0 \\
y & 0 & 23 & 23 \\
\end{array}
\]

\(y\)
\[
\begin{array}{c|c|c|c|c}
 & \text{safe} & \text{repair} & \text{done} \\
\hline
x & 0 & 23 & 0 \\
y & 0 & 23 & 23 \\
\end{array}
\]
Timed automata

The timed automaton model

Can be viewed:

▶ as the timed word

(problem, 23)
(delayed, 38.6)
(repair, 40.9)
(done, 63)

▶ as the signal

safe alarm failsafe repairing

\[
\begin{align*}
x &= 0 & 23 & safe & problem & alarm & 15.6 & alarm & delayed & failsafe & 2 & 3 & failsafe & repair & reparation \\
y &= 0 & 23 & & 23 & 0 & 15.6 & 15.6 & 17.9 & 17.9 & 0 & 2.3 & 0
\end{align*}
\]
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
The timed automaton model

Timed automata

Can be viewed:

▶ as the timed word

(problem, 23)(delayed, 38.6)(repair, 40)(done, 40)

▶ as the signal

safe → alarm → delayed → failsafe → repair → reparation → done → safe

x 0 23 0 15.6 15.6 2.3 17.9 17.9 0 40 40
y 0 23 23 38.6 15.6 0 2.3 22.1 0 22.1 22.1
The timed automaton model

Timed automata

Can be viewed:

- as the timed word (problem,23)(delayed,38.6)(repair,40.9)(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 \textsc{PSPACE}-complete) for timed automata \cite{AD94}.

\cite{AD94} Alur, Dill. A theory of timed automata (TCS, 1994).
The timed automaton model

Basic result on timed automata

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

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:

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

model-checking algorithm

\[ \text{yes/no} \]

property:
Back to the model-checking problem

system:

Timed automaton

property:

G (request → F grant)

model-checking algorithm

yes/no
Back to the model-checking problem

system:

property:

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

model-checking algorithm

yes/no

MTL formula
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).
The model-checking problem

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!

[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>pointwise sem.</th>
<th>continuous sem.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MTL</strong></td>
<td>decidable, NPR [OW05]</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>

- 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>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)
- All hardness results: by reduction to the halting problem for FIFO channel machines

[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**!
- The gap between branching-time and linear-time dramatically increases in the timed framework...
  (reminder: model-checking TCTL is PSPACE-complete)
- All hardness results: by reduction to the halting problem for FIFO channel machines

---

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


From LTL to alternating automata

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

\[ \mathbf{G} (a \rightarrow \mathbf{F} b) \]
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

\[ G(a \rightarrow Fb) \]
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

\[ G(a \rightarrow F_{[1,2]} b) \]
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 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): Higman ⊑ Higmount.
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

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>
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></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.
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 $G (\bullet \rightarrow F_{\geq 1} \bullet)$ is undecidable”
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?
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?

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

\[
\text{MITL } \exists \varphi ::= a \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \varphi U_{/} \varphi
\]

with / 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):

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

with $I$ 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 $G(\bullet \rightarrow F_{=1} \bullet)$ is undecidable”

- What if we forbid punctual constraints in MTL?

Metric Interval Temporal Logic (MITL):

$$\text{MITL} \ni \varphi ::= a \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \varphi \cup_{I} \varphi$$

with $I$ 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 \textit{EXPSPACE}-complete \cite{AFH96}.
Model-checking MITL is “easy”

Theorem

The model-checking problem for MITL is EXPSPACE-complete \cite{AFH96}.

- we can bound the \textit{variability} of the signals
Model-checking MITL is “easy”

Theorem

The model-checking problem for MITL is EXPSPACE-complete [AFH96].

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

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

**Theorem**

The model-checking problem for MITL is EXPSPACE-complete [AFH96].

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

**Example:** consider the formula $\varphi = \mathsf{G}_{(0,1)} (\bullet \rightarrow \mathsf{F}_{[1,2]} \cdot)$

- 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 \( \text{EXPSPACE}\)-complete [AFH96].

- 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
Model-checking MITL is “easy”

Theorem
The model-checking problem for MITL is \textsc{ExpSpace}-complete \cite{afh96}.

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
  something more clever needs to be done

\cite{hr04} Hirshfeld, Rabinovich. Logics for real time: decidability and complexity (Fundamenta Informaticae, 2004).
Some interesting fragments

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

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

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

This idea can be extended to any formula in MITL.
\[ \varphi = \mathbf{G}_{(0,1)} \left( a \rightarrow \mathbf{F}_{[1,2]} b \right) \]
\[ \varphi = \mathbf{G}_{(0,1)}(a \rightarrow \mathbf{F}_{[1,2]} b) \]

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

- Do punctual constraints really need to be banned?
A co-flat fragment of MTL

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

\[
\text{coFlat-MTL} \ni \phi ::= a \mid \neg a \mid \phi \lor \phi \mid \phi \land \phi \mid \phi_U \psi \mid \psi \sim U \phi
\]

where \( I \) unbounded \( \Rightarrow \) \( \psi \) \( \in \) LTL

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

\[\text{[BMOW07] Bouyer, Markey, Ouaknine, Worrell. The cost of punctuality (LICS'07).}\]
A co-flat fragment of MTL

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

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

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

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 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:
  - \( G (\bullet \rightarrow F_{=1} \bullet) \) is 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 coFlat-MTL:

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

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

- Examples:
  - $G (\bullet \rightarrow F_{=1} \bullet)$ 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 coFlat-MTL:

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

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

- Examples:
  - \( G (\bullet \to F_{=1} \bullet) \) is in coFlat-MTL
  - \( F G_{\leq 1} \bullet \) is not in coFlat-MTL
  - coFlat-MTL contains Bounded-MTL (all modalities are time-bounded)

Model-checking coFlat-MTL is “easy”

Theorem
The model-checking problem for coFlat-MTL or Bounded-MTL is \textsc{ExpSpace}-complete [BMOW07].
Model-checking coFlat-MTL is “easy”

Theorem

The model-checking problem for coFlat-MTL or Bounded-MTL is EXPSPACE-complete [BMOW07].

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

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

Theorem

The model-checking problem for coFlat-MTL or Bounded-MTL is EXPSPACE-complete [BMOW07].

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

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

Theorem
The model-checking problem for coFlat-MTL or Bounded-MTL is EXPSPACE-complete [BMOW07].

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

\[ \varphi_n \equiv \Diamond \land G_{[0,2^n]} \varphi_D \quad \text{with} \quad \varphi_D = (\rightarrow F = 1 (\Diamond \land F \leq 1)) \land (\rightarrow F = 1 (\Diamond \land F \leq 1)) \]

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

\[ G \leq 1 (\Diamond \land F = 1) \land G \leq 1 (\Diamond \land F = 1) \] defines the context-free language \{n 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].

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

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

Theorem

The model-checking problem for coFlat-MTL or Bounded-MTL is EXPSPACE-complete [BMOW07].

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

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

with

\[ \varphi_D = (\bullet \rightarrow F_{=1} (\bullet \wedge F_{\leq 1} \bullet)) \wedge (\bullet \rightarrow F_{=1} (\bullet \wedge F_{\leq 1} \bullet)) \]
Model-checking coFlat-MTL is “easy”

Theorem
The model-checking problem for coFlat-MTL or Bounded-MTL is EXPSPACE-complete [BMOW07].

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

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

- 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

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

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

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

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

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

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

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

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

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 coFlat-MTL

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

\( \varphi \mapsto \) 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
Algorithm for coFlat-MTL

\[ \varphi \mapsto \text{alternating timed automata } B_{\neg \varphi} \text{ for } \neg \varphi \text{ with a ‘flatness’ property} \]

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
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
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]!
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$\text{MITL}$ unifies coFlat-MTL and MITL, and is EXPSPACE-complete [BMOW08]!
- No real data structures do exist for these logics.