On the Model Checking of Timed and Weighted Temporal Logics

Patricia Bouyer

LSV, CNRS & ENS Cachan, France
On the Model Checking of Timed and Weighted Temporal Logics

Patricia Bouyer

LSV, CNRS & ENS Cachan, France
Outline

1. Introduction

2. Definition of the logics

3. The timed automaton model

4. The model-checking problem

5. Some interesting fragments

6. Conclusion
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:

$G (\text{request} \rightarrow F \text{grant})$
Model-checking

system:

property:

$G (\text{request} \rightarrow F \text{grant})$
Model-checking

system:

property:

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

model-checking algorithm

yes/no
The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

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

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

Linear-time temporal logic [Pnu77]

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

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

Linear-time temporal logic \[\text{LTL} \ni \varphi ::= p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \neg \varphi \mid X \varphi \mid \varphi U \varphi\]

\[\begin{array}{c}
\varnothing
\end{array}\]
The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

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

\[ \begin{align*}
\text{---} & \quad \vdash \quad \text{---} \\
& \quad \text{---} \quad \quad \vdash \quad \text{---}
\end{align*} \]

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

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

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

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

Linear-time temporal logic [Pnu77]

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

Introductions
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]

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

[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:

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

[Pnu77] Pnueli. The temporal logic of programs (FOCS'77).
Introduction

The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

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

- 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]

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

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

- liveness property:
  \[G F \bullet\]

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

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

Linear-time temporal logic \([Pnu77]\)

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

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

- liveness property:
  \[
  GF \bullet
  \]

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

- 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)

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

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

Metric Temporal Logic (MTL)

MTL \models \varphi ::= a \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \varphi \mathbf{U}_I \varphi

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

- This is a timed extension of LTL
Metric Temporal Logic (MTL)

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

where \( l \) 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)

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

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:

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

\[\text{the system is observed only when actions happen}\]
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

\[\models \mathbf{U}_{[1,2]} \bullet\]
The pointwise semantics

MTL formulas are interpreted over timed words:

\((\cdot, .6)(\cdot, 1.1)(\cdot, 1.2)(\cdot, 1.3) \ldots\)

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

\(\not\models G_{[2,3]} \cdot\)

the system is observed only when actions happen
The continuous semantics

MTL formulas are interpreted over (finitely variable) signals:

\[
\begin{align*}
t \in [0, 0.6] & \mapsto \square_t \\
(0.6, 1.1) & \mapsto \\not\square_t \\
[1.1, 1.2) & \mapsto \square_t \\
\end{align*}
\]
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
Definition of the logics

The continuous semantics

MTL formulas are interpreted over (finitely variable) signals:

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

...the system is observed continuously

\[ \models U_{[1, 2]} \]
The continuous semantics

MTL formulas are interpreted over (finitely variable) signals:

The system is observed continuously

\[ t \in [0, 6) \Rightarrow \text{blue} \quad t \in (6, 11) \Rightarrow \text{red} \quad t \in [11, 12) \Rightarrow \text{green} \]

\[ \models \quad \text{U}_{[1, 2]} \]

\[ \not\models \quad \text{G}_{[2, 3]} \]
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”
  \[ \text{G}(\text{problem} \rightarrow \text{F}_{\leq 56} \text{alarm}) \]

- “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”
  \[ \text{G}(\text{problem} \rightarrow (\text{F}_{\leq 15} \text{repair} \lor \text{G}_{[12,15)} \text{alarm})) \]
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"
  \[ G(\text{problem} \rightarrow (F_{\leq 15} \text{repair} \lor G_{[12,15)} \text{alarm})) \]

- \( F_{=2} \text{repair} \) vs \( F_{=1}(F_{=1} \text{repair}) \)
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”
  \(G(\text{problem} \rightarrow (F_{\leq 15} \text{repair} \lor G_{[12,15]} \text{alarm}))\)

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

  \([-F_{=2} \quad \not\equiv \quad F_{=1} (F_{=1} \bullet)\)
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 \( F_{=1} (F_{=1} \text{ repair}) \)

\[ \models F_{=2} \bullet \quad \not\models F_{=1} (F_{=1} \bullet) \]

\[ \models F_{=2} \bullet \quad \models F_{=1} (F_{=1} \bullet) \]
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 \( F_{=1}(F_{=1} \text{repair}) \)

  \[
  \begin{align*}
  0 & \quad 1 & \quad 2 \\
  \models F_{=2} & \quad \not\models F_{=1}(F_{=1} \bullet) \\
  \models F_{=2} & \quad \models F_{=1}(F_{=1} \bullet)
  \end{align*}
  \]

  - in the pointwise semantics, \( F_{=2} \bullet \not\equiv F_{=1} F_{=1} \bullet \)
  - in the continuous semantics, \( F_{=2} \bullet \equiv F_{=1} F_{=1} \bullet \)
Some further extensions

- Timed Propositional Temporal Logic (TPTL) [AH89]
  \[ \text{TPTL} = \text{LTL} + \text{clock variables} + \text{clock constraints} \]

Some further extensions

- Timed Propositional Temporal Logic (TPTL) [AH89]

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

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

Some further extensions

- Timed Propositional Temporal Logic (TPTL) \[ \text{[AH89]} \]
  
  \[
  \text{TPTL} = \text{LTL} + \text{clock variables} + \text{clock constraints}
  \]

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

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

Some further extensions

- Timed Propositional Temporal Logic (TPTL) [AH89]
  \[ \text{TPTL} = \text{LTL} + \text{clock variables} + \text{clock constraints} \]

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

- MTL+Past: add past-time modalities [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 \[ \text{MTL+Past} \]

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


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

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

Theorem

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

Conjecture in 1990: the \(\text{TPTL} \) formula

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

cannot be expressed in \(\text{MTL} \).

\cite{Kam68} Kamp. Tense logic and the theory of linear order (PhD Thesis UCLA 1968).
\cite{GPSS80} Gabbay, Pnueli, Shelah, Stavi. On the temporal analysis of fairness (POPL'80).
\cite{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.

---


[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

![Diagram of continuous semantics]

$$G \bullet \rightarrow$$
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

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

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

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
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

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

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

$$F \leq 1 \bullet \land F_{[1,2]}$$

$$\lor F \leq 1 (\bullet \land F \leq 1 \bullet)$$

$$\lor F \leq 1 (F \leq 1 \bullet \land F = 1 \bullet)$$
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

\((\text{problem}, 0)(\text{delayed}, 38.6)(\text{repair}, 40.9)(\text{done}, 63)\)

as the signal
Timed automata
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

\(\text{safe} \rightarrow \text{alarm} \rightarrow \text{failsafe} \rightarrow \text{repairing} \rightarrow \text{done} \rightarrow \text{safe}\)
Timed automata

The timed automaton model can be viewed as the timed word $(\text{problem, } x:=0)$, $(\text{delayed, } y:=0)$, $(\text{repair, } x \leq 15)$, $(\text{done, } 22 \leq y \leq 25)$, $(\text{repairing, } y:=0)$, $(\text{repair, } 2 \leq y \land x \leq 56)$, $(\text{reparation, } 22 \leq y \leq 25)$, $(\text{done, } y:=0)$, $(\text{safe, } y:=0)$.
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 → problems → repairing → failsafe → repairing

\[
\begin{align*}
x & \text{ safe} & 23 & \text{ safe} & x := 0 \\
y & 0 & 23 & & y := 0
\end{align*}
\]

\[
\begin{align*}
x & \text{ problem} & 0 & \text{ alarm} & 15.6 & \text{ alarm} \\
y & 0 & 23 & 15.6 & 38.6 &
\end{align*}
\]
The timed automaton model
Timed automata
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.
Timed automata

The timed automaton model

safe

problem, \( x:=0 \)

alarm

repair

failsafe

done, \( 22 \leq y \leq 25 \)

repair, \( x \leq 15 \)

delayed, \( y:=0 \)

\( 2 \leq y \land x \leq 56 \)

\( y:=0 \)

\( 15 \leq x \leq 16 \)

\( y:=0 \)

\( 2 \leq y \leq 25 \)

\( 15 \leq y \leq 25 \)

\( 17.9 \)

\( 22.1 \)

\( 40 \)

\( 22.1 \)

\( 0 \)

\( 2.3 \)

\( 17.9 \)

\( 0 \)

\( 17.9 \)

\( 0 \)

\( 2.3 \)

\( 0 \)

\( 17.9 \)

\( 0 \)

\( 2.3 \)

\( 0 \)

\( 17.9 \)

\( 0 \)

\( 2.3 \)

\( 0 \)

\( 17.9 \)

\( 0 \)

\( 2.3 \)

\( 0 \)

\( 17.9 \)

\( 0 \)

\( 2.3 \)

\( 0 \)

\( 17.9 \)

\( 0 \)

\( 2.3 \)

\( 0 \)

\( 17.9 \)

\( 0 \)

\( 2.3 \)

\( 0 \)

\( 17.9 \)

\( 0 \)

\( 2.3 \)

\( 0 \)

\( 17.9 \)

\( 0 \)

\( 2.3 \)

\( 0 \)

\( 17.9 \)

\( 0 \)

\( 2.3 \)

\( 0 \)

\( 17.9 \)

\( 0 \)

\( 2.3 \)

\( 0 \)

\( 17.9 \)

\( 0 \)

\( 2.3 \)

\( 0 \)

\( 17.9 \)

\( 0 \)

\( 2.3 \)

\( 0 \)

\( 17.9 \)

\( 0 \)

\( 2.3 \)

\( 0 \)

\( 17.9 \)

\( 0 \)

\( 2.3 \)

\( 0 \)

\( 17.9 \)

\( 0 \)

\( 2.3 \)

\( 0 \)
Timed automata

Can be viewed:
- as the timed word $\langle \text{problem}, 23 \rangle \langle \text{delayed}, 38.6 \rangle \langle \text{repair}, 40.0 \rangle \langle \text{done}, 63.0 \rangle$
- as the signal $0 \to 10 \to 20 \to 30 \to 40 \to 50 \to 60 \to 70$

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

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 $\text{PSPACE}$-complete) for timed automata \[\text{[AD94]}\].

\[\text{[AD94]}\] Alur, Dill. A theory of timed automata (TCS, 1994).
Basic result on timed automata

**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 $\text{PSPACE}$-complete) for timed automata [AD94].

![Timed automaton](image1)

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:

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

model-checking algorithm

yes/no
Back to the model-checking problem

- System: Timed automaton
- Property: $G(request \rightarrow Fgrant)$
- 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>Logic</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!
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)

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

\[ s_1, s_2, s_3, s_4, s_5 \]

- \( s_1 \) to \( s_2 \) with !
- \( s_2 \) to \( s_3 \) with ?
- \( s_3 \) to \( s_4 \) with ?
- \( s_4 \) to \( s_5 \) with ?

- \( s_5 \) is reachable
- \( s_5 \) is not reachable
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
A short visit to channel machines (1)

A channel machine = a finite automaton + a FIFO channel

A channel machine

\[ s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \rightarrow s_5 \]

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

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

\[ s_1 \xrightarrow{!} s_2 \xrightarrow{!} s_3 \xrightarrow{?} s_4 \xrightarrow{?} s_5 \]

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

- !
- ?
- !

\$s_1\$ is not reachable

\$s_5\$ is reachable

- 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?

Schneebelen. Verifying lossy channel systems has non-primitive recursive complexity (IPL, 2002).]
A short visit to channel machines (2)

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 decidable but NPR for channel machines with insertion errors [Sch02].

[Sch02] Schnoebelen. Verifying lossy channel systems has non-primitive recursive complexity (IPL, 2002).
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:

\[(q_0, \varepsilon) \xrightarrow{a!} (q_1, a) \xrightarrow{b!} (q_2, ab) \xrightarrow{a?} (q_3, b) \xrightarrow{c!} (q_4, bc) \xrightarrow{b?} (q_5, c) \cdots\]
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 \( \mathcal{A}_{\text{univ}} \models \neg \varphi \)

* possibly with insertion errors
The model-checking problem

Constraints satisfied by the timed word

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

$$\mathbf{G}(a! \rightarrow \mathbf{F}_{=1} a?)$$

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!

*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 $a$,

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

This formula is not sufficient!

\[=1 \text{ t.u.}\]

only encodes a channel machine with insertion errors!

\[=1 \text{ t.u.}\]

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?

\[
G \left( (F_{=1} a?) \rightarrow a! \right)
\]
We need to express the property:

“Every \textit{a?}-event is preceded one time unit earlier by an \textit{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
The model-checking problem

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?
  \[
  G (a? \rightarrow F_{=1}^{-1} a!)
  \]
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?
  \[ \mathbf{G} \left( (\mathbf{F}_{-1} a?) \rightarrow a! \right) \]
  - correct in the continuous semantics
  - not correct in the pointwise semantics

- Why not look back in the past?
  \[ \mathbf{G} \left( a? \rightarrow \mathbf{F}^{-1} a! \right) \]
  - correct for $\text{MTL}+\text{Past}$ (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 \( \text{MTL+Past} \) (in the continuous and in the pointwise sem.)
  - no direct translation into \( \text{MTL} \)
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?
  \[ G ((F_{=1} a?) \rightarrow a!) \]
  - 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:
  \[ \neg \left( F x . X y . F (x > 1 \land y < 1 \land c?) \right) \]
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 \( \text{MTL}+\text{Past} \) (in the continuous and in the pointwise sem.)
  - no direct translation into \( \text{MTL} \)

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

- 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><strong>MTL</strong></td>
<td>NPR ([\text{OW07]})</td>
<td>undecidable ([\text{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 ([\text{AH94]})</td>
<td>undecidable ([\text{AH94]})</td>
</tr>
</tbody>
</table>
What remains to be proved

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 [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 alternating (Büchi) automata

\[
G (a \rightarrow F b)
\]
LTL formulas can be turned into alternating (Büchi) automata

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

LTL formulas can be turned into alternating (Büchi) automata

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

LTL formulas can be turned into 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

\[ G(a \to F_{[1,2]} b) \]
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. There is a well quasi-order on the set of abstract configurations (subword relation): Higman ⊑ Higman.
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):

\[
\text{higman} \sqsubseteq \text{highmountain}
\]
The model-checking problem

Summary

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


[OW06] Ouaknine, Worrell. On metric temporal logic and faulty Turing machines (FoSSaCS’06).
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 infinite 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</td>
<td>undecidable</td>
</tr>
<tr>
<td>MTL+Past</td>
<td>undecidable</td>
<td>undecidable</td>
</tr>
<tr>
<td>TPTL</td>
<td>undecidable</td>
<td>undecidable</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.

The fragment without punctuality

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

**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.
  
  **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?

The fragment without punctuality

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

\[
 MITL \ni \phi ::= a | \neg \phi | \phi \lor \phi | \phi \land \phi | \phi \mathbf{U} I \phi
\]

with \( I \) a non-punctual interval

---

The fragment without punctuality

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

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

- What if we forbid punctual constraints in MTL?

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

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

with \( / \) a non-punctual interval

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

---

The fragment without punctuality

- The undecidability/NPR proofs heavily rely on punctual constraints.
  
  **Claim:** “Any logic strong enough to express the property \( G (\bullet \to F_{\geq 1} \bullet) \) is undecidable”

- What if we forbid punctual constraints in MTL?

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

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

with \( / \) a non-punctual interval

- Examples:
  - \( G (\bullet \to F_{\geq 1} \bullet) \) is not in MITL
  - \( G (\bullet \to F_{[1,2]} \bullet) \) is in MITL

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}.

\cite{HR04} Hirshfeld, Rabinovich. Logics for real time: decidability and complexity (Fundamenta Informaticae, 2004).
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}.

we can bound the \textit{variability} of the signals

\cite{HR04} Hirshfeld, Rabinovich. Logics for real time: decidability and complexity (Fundamenta Informaticae, 2004).
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 $\varphi = G_{(0,1)} (\bullet \rightarrow F_{[1,2]} \bullet)$
Model-checking MITL is “easy”

**Theorem**

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

[Hirshfeld, Rabinovich. Logics for real time: decidability and complexity (Fundamenta Informaticae, 2004).]
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

\[\text{[HR04] Hirshfeld, Rabinovich. Logics for real time: decidability and complexity (Fundamenta Informaticae, 2004).}\]
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}.

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

\textbf{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).
\[ \varphi = G_{(0,1)}(a \rightarrow F_{[1,2]} b) \]
\[ \varphi = G_{(0,1)}(a \rightarrow F_{[1,2]} b) \]
\( \varphi = G_{(0,1)}(a \rightarrow F_{[1,2]} b) \)
\[ \varphi = \text{G}_{(0,1)} (a \rightarrow \text{F}_{[1,2]} b) \]
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?
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 $\text{coFlat-MTL}_{\text{LTL}}$:

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

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 $\text{coFlat-MTL}_{\text{LTL}}$: 

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

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

- Examples:
  - $G (\bullet \rightarrow F_{\leq 1} \bullet)$ is in $\text{coFlat-MTL}_{\text{LTL}}$
A co-flat fragment of MTL

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

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

\[
\text{coFlat-MTL}_{LTL} \ni \varphi ::= a \mid \neg a \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \varphi \mathcal{U} \psi \mid \psi \widetilde{\mathcal{U}} \varphi
\]

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

- Examples:
  - \( G (\bullet \to F_{=1} \bullet) \) is in \( \text{coFlat-MTL}_{LTL} \)
  - \( F G_{\leq 1} \bullet \) is not in \( \text{coFlat-MTL}_{LTL} \)

A co-flat fragment of MTL

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

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

$$
\text{coFlat-MTL}_{\text{LTL}} \ni \phi ::= a \mid \neg a \mid \phi \lor \phi \mid \phi \land \phi \mid \phi U I \psi \mid \psi \tilde{U} I \phi
$$

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

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

Model-checking coFlat-MTL\textsubscript{LTL} is “easy”

**Theorem**

The model-checking problem for coFlat-MTL\textsubscript{LTL} or Bounded-MTL is EXPSPACE-complete [BMOW07]. If constants are encoded in unary, the model-checking of Bounded-MTL is PSPACE-complete.
Model-checking coFlat-MTL_{\text{LTL}} is “easy”

**Theorem**

The model-checking problem for coFlat-MTL_{\text{LTL}} 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.):

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

**Theorem**

The model-checking problem for coFlat-MTL$_{\text{LTL}}$ 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.):

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

Theorem

The model-checking problem for coFlat-MTL$\text{LTL}$ 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.):

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

with

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

\[
\bigotimes 
\]
Model-checking coFlat-MTL$_{\text{LTL}}$ is “easy”

**Theorem**

The model-checking problem for coFlat-MTL$_{\text{LTL}}$ 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.):

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

### Theorem

The model-checking problem for coFlat-MTL\textsubscript{LTL} or Bounded-MTL is \textsc{ExpSpace}-complete [BMOW07]. If constants are encoded in unary, the model-checking of Bounded-MTL is \textsc{PSPACE}-complete.

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

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

\[ \]
Model-checking coFlat-MTL\textsubscript{LTL} is “easy”

Theorem

The model-checking problem for coFlat-MTL\textsubscript{LTL} 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.):

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

- 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 the formula

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

Assume one wants to verify the formula

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

Assume one wants to verify the formula

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

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 the 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 the 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 the formula

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

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 the formula

\[ G_{<2}(\mathcal{F}_1) \]

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 the 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 the 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 the 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 the 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_{LTL}

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

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

Only counter-examples of the following form need to be looked for:

where - the number of hard fragments is at most exponential
- the total duration of hard fragments is at most exponential
Algorithm for coFlat-MTL_{LTL}

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

Only counter-examples of the following form need to be looked for:

- the number of hard fragments is at most exponential
- the total duration of hard fragments is at most exponential

- hard fragment = sliding window algorithm
- LTL fragment = finite automaton computation
The ultimate fragment?

- MITL and coFlat-MTL$_{LTL}$ have ‘low’ complexities for pretty different reasons...
Some interesting fragments

The ultimate fragment?

- MITL and coFlat-MTL\textsubscript{LTL} have 'low' complexities for pretty different reasons...
- ... why not mix the two logics?
The ultimate fragment?

- MITL and coFlat-MTL_{LT} have ‘low’ complexities for pretty different reasons...
- ... why not mix the two logics?

We define coFlat-MTL_{MITL}:

\[
\text{coFlat-MTL}_{MITL} \ni \phi ::= a \mid \neg a \mid \phi \lor \phi \mid \phi \land \phi \mid \phi U I \psi \mid \psi \tilde{U} I \phi
\]

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

The ultimate fragment?

- MITL and coFlat-MTL\textsubscript{LTL} have ‘low’ complexities for pretty different reasons...
- ... why not mix the two logics?

We define coFlat-MTL\textsubscript{MITL}:

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

where $I$ unbounded $\Rightarrow \psi \in \text{MITL}$.

- Examples:
  - $F G_{\leq 1} \cdot$ is in coFlat-MTL\textsubscript{MITL}
  - $F G_{= 1} \cdot$ is not in coFlat-MTL\textsubscript{MITL}
  - coFlat-MTL\textsubscript{MITL} generalizes both logics MITL and coFlat-MTL\textsubscript{LTL}

Model-checking $\text{coFlat-MTL}_{\text{MITL}}$

**Theorem**

The model-checking problem for $\text{coFlat-MTL}_{\text{MITL}}$ is $\text{EXPSPACE}$-complete [BMOW08], regardless of the encoding of constants in unary or in binary.
Model-checking $\text{coFlat-MTL}_{\text{MITL}}$

Theorem

The model-checking problem for $\text{coFlat-MTL}_{\text{MITL}}$ is \textsc{ExpSpace}-complete [BMOW08], regardless of the encoding of constants in unary or in binary.

Only counter-examples of the following form need to be looked for:

![Diagram showing punctual and non-punctual segments with somewhat stretchable signals](image-url)
Model-checking $\text{coFlat-MTL}_{\text{MITL}}$

**Theorem**

The model-checking problem for $\text{coFlat-MTL}_{\text{MITL}}$ is $\text{EXPSPACE}$-complete [BMOW08], regardless of the encoding of constants in unary or in binary.

Only counter-examples of the following form need to be looked for:

![Diagram showing punctual and non-punctual segments with a stretchable signal.]

**Stretchable signal:**

Any model of an $\text{LTL}$ formula is punctual. Any model of an $\text{MITL}$ formula is somewhat stretchable.
Model-checking $\text{coFlat-MTL}_{\text{MITL}}$

Theorem

The model-checking problem for $\text{coFlat-MTL}_{\text{MITL}}$ is EXPSPACE-complete [BMOW08], regardless of the encoding of constants in unary or in binary.

Only counter-examples of the following form need to be looked for:

Stretchable signal:

- Any model of an LTL formula is stretchable.
Theorem

The model-checking problem for $\text{coFlat-MTL}_{\text{MITL}}$ is $\text{EXPSPACE}$-complete [BMOW08], regardless of the encoding of constants in unary or in binary.

Only counter-examples of the following form need to be looked for:

1. **Punctual**:
   - Any model of an LTL formula is stretchable.
   - Any model of an MITL formula is somewhat stretchable.

2. **Non-punctual**:
   - Somewhat stretchable
• non-punctual part: somewhat stretchable

→ transform into LTL constraints
• non-punctual part: somewhat stretchable
  → transform into LTL constraints

• punctual part:
• non-punctual part: somewhat stretchable
  → transform into LTL constraints

• punctual part: non-stretchable... 😞
• **non-punctual part:** somewhat stretchable
  → transform into **LTL** constraints

• **punctual part:** non-stretchable... 😞
  but not too long... ☺️
• non-punctual part: somewhat stretchable
  $\rightarrow$ transform into LTL constraints

• punctual part: non-stretchable...
  but not too long...
• **non-punctual part:** somewhat stretchable
  \[\rightarrow\] transform into **LTL** constraints

• **punctual part:** non-stretchable...
  \[\frownie\]
  but not too long...
  \[\smiley\]

\[
\begin{align*}
0 & \quad 1 & \quad 2 & \quad 3 & \quad 4 \\
\uparrow & \quad \mathbf{F}_{=1} & \quad \uparrow
\end{align*}
\]

\[
\begin{align*}
0 & \quad 1 & \quad 2 & \quad 3 & \quad 4 \\
\uparrow & \quad \mathbf{F}_{=1} & \quad \uparrow
\end{align*}
\]

\[
\begin{align*}
\mathbf{F}_{=1} & \quad \mathbf{F}_{=1}
\end{align*}
\]

\[
\begin{align*}
\mathbf{F}_{=1} & \quad \mathbf{F}_{=1}
\end{align*}
\]

track \(i\)

\[
\begin{align*}
2 & \quad 3 & \quad 4 \\
\uparrow & \quad \mathbf{F}_{=1} & \quad \uparrow
\end{align*}
\]

track \(i+1\)

\[
\begin{align*}
\mathbf{F}_{=1} & \quad \mathbf{F}_{=1}
\end{align*}
\]

\[
\begin{align*}
\mathbf{F}_{=1} & \quad \mathbf{F}_{=1}
\end{align*}
\]

\[
\begin{align*}
\mathbf{F}_{=1} & \quad \mathbf{F}_{=1}
\end{align*}
\]
A tableau satisfiability problem

check punctual formulas

check non-punctual formulas

1 time unit

track 1

track 2

track 3

track 4

track 5

(track k−3)

(track k−2)

(track k−1)

(track k)

bounded height

...
A tableau satisfiability problem

transform into sat. prob. for LTL+Past over $\mathbb{R}_+$ (PSPACE: [Rey04])

[Rey04] Reynolds. The complexity of the temporal logic over the reals (Subm.'04).
Some interesting fragments

The end of the quest for tractable fragments of MTL?

Parys, Walukiewicz. Weak Alternating Timed Automata (ICALP'09).

MTL

LTL

MITL

Safety-MTL

Bounded-MTL

coFlat-MTL_{LTL}

coFlat-MTL_{MITL}

undec./NPR

PSPACE-c.

EXPSPACE-c.
The end of the quest for tractable fragments of MTL?

- Bounded-MTL
- LTL
- MITL
- coFlat-MTL\(_{LTL}\)
- coFlat-MTL\(_{MITL}\)
- Safety-MTL
- MTL

\[\text{PSPACE-c.} \quad \text{EXPSPACE-c.} \quad \text{undec./NPR}\]

+ add Positive-MTL [PW09]

---

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
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

- No real data structures do exist for these logics.
Going further with quantities...

- System **resources** might be relevant and even crucial information
  - energy consumption, memory usage, price to pay, bandwidth, ...
Going further with quantities...

- **System resources** might be relevant and even crucial information
  - energy consumption, memory usage, price to pay, bandwidth, ...

- We need to integrate those aspects in models and in logics
  - *Models:* hybrid automata, timed automata with observer variables
  - *Logics:* extensions of classical temporal logics with quantitative constraints on the observer variables
Going further with quantities...

- System **resources** might be relevant and even crucial information
  - energy consumption, memory usage, price to pay, bandwidth, ...

- We need to integrate those aspects in models and in logics
  - **Models:** hybrid automata, timed automata with observer variables
  - **Logics:** extensions of classical temporal logics with quantitative constraints on the observer variables

- Very few decidability results, unless strong restrictions..
Going further with quantities...

- System **resources** might be relevant and even crucial information
  - energy consumption, memory usage, price to pay, bandwidth, ...

- We need to integrate those aspects in models and in logics
  - **Models:** hybrid automata, timed automata with observer variables
  - **Logics:** extensions of classical temporal logics with quantitative constraints on the observer variables

- Very few decidability results, unless strong restrictions..
  - ... but this is a **very active field of research!**
Weighted/priced timed automata [ALP01,BFH+01]

\[ \ell_0 \rightarrow \ell_1 \quad x \leq 2, c, y := 0 \]

\[ \ell_1 \rightarrow \ell_2: u \quad (y = 0) \quad +10 \]

\[ \ell_2 \rightarrow \ell_3: u \quad x = 2, c \quad +1 \]

\[ \ell_2 \rightarrow \text{smiley} \quad x = 2, c \quad +1 \]

\[ x = 2, c \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]

\[ \ell_3 \rightarrow \text{smiley} \quad +1 \]
Weighted/priced timed automata [ALP01,BFH+01]

Weighted/priced timed automata [ALP01,BFH+01]

\[ \ell_0 + 5 \xrightarrow{x \leq 2, c, y := 0} \ell_1 \xrightarrow{(y=0)} \ell_2 \xrightarrow{u} \ell_3 \xrightarrow{x = 2, c} +10 \rightarrow \ell_2 \xrightarrow{u} \ell_3 \xrightarrow{x = 2, c} +7 \rightarrow \ell_3 \xrightarrow{c} +1 \rightarrow \text{smiley} \]

\[
x \quad 0 \quad 1.3 \\
y \quad 0 \quad 1.3 \\
\]

\[ \text{cost :} \]

\[ \ell_0 \xrightarrow{1.3} \ell_0 \xrightarrow{c} \ell_1 \xrightarrow{u} \ell_3 \xrightarrow{0.7} \ell_3 \xrightarrow{c} \text{smiley} \]
Weighted/priced timed automata [ALP01,BFH+01]

\[
\begin{align*}
\ell_0 & \xrightarrow{+5} \ell_1 \quad (y=0) \quad \xrightarrow{u} \ell_2 \quad x=2, c \\
\ell_1 & \xrightarrow{u} \ell_3 \quad x=2, c \\
\ell_2 & \xrightarrow{+10} \\
\ell_3 & \xrightarrow{+1} \\
\end{align*}
\]

\[
\begin{array}{|c|c|c|c|c|c|c|}
\hline
\ell & 0 & 1.3 & 0 & 1.3 & 0.7 & 0 \hline
x & 0 & 1.3 & 1.3 & 1.3 & 2 & \smiley \hline
y & 0 & 1.3 & 0 & 0 & 0.7 & \smiley \hline
\end{array}
\]

cost : 6.5


Weighted/priced timed automata \[\text{[ALP01,BFH+01]}\]

\[
\begin{align*}
\ell_0 & \xrightarrow{5} \ell_0 \\
x \leq 2, c, y := 0 & \xrightarrow{c} \ell_1 \\
( y = 0 ) & \\
\ell_1 & \xrightarrow{u} \ell_3 \\
\ell_3 & \xrightarrow{1} \ell_3 \\
 & \xrightarrow{x = 2, c} \ell_2 \\
 & \xrightarrow{1} \ell_2 \\
 & \xrightarrow{+10} \ell_0 \\
\end{align*}
\]

\[
\begin{array}{c|ccc|ccc|c}
 & \ell_0 & \ell_1 & \ell_2 & \ell_3 & \ell_3 & \ell_0 \\
x & 0 & 1.3 & 1.3 & 1.3 & 2 & 0 \\
y & 0 & 1.3 & 0 & 0 & 0.7 & 0 \\
\end{array}
\]

\[
cost : \quad 6.5 + 0
\]

Weighted/priced timed automata \[\text{[ALP01,BFH+01]}\]

\[
\begin{align*}
\ell_0 & \xrightarrow{x \leq 2, c, y := 0} \ell_1 \\
\ell_1 & \xrightarrow{u} \ell_2 \\
\ell_2 & \xrightarrow{x = 2, c} \ell_3 \\
\ell_3 & \xrightarrow{x = 2, c} \ell_2 \\
\ell_3 & \xrightarrow{u} \ell_1 \\
\ell_1 & \xrightarrow{c} \ell_0
\end{align*}
\]

<table>
<thead>
<tr>
<th>State</th>
<th>$x$</th>
<th>$y$</th>
<th>$\ell_0$</th>
<th>$\ell_1$</th>
<th>$\ell_2$</th>
<th>$\ell_3$</th>
<th>$\ell_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\ell_0$</td>
<td>0</td>
<td>0</td>
<td>1.3</td>
<td>1.3</td>
<td>1.3</td>
<td>1.3</td>
<td>2</td>
</tr>
<tr>
<td>$\ell_1$</td>
<td>0</td>
<td>1.3</td>
<td>1.3</td>
<td>0</td>
<td>0</td>
<td>0.7</td>
<td></td>
</tr>
</tbody>
</table>

**Cost:**

\[
\begin{align*}
\text{cost} & : 6.5 + 0 + 0 \\
\end{align*}
\]


Weighted/priced timed automata [ALP01,BFH+01]

$\ell_0 \xrightarrow{+5} \ell_1 \xrightarrow{u} \ell_2 \xrightarrow{x=2,c} +1 \xrightarrow{+10} \ell_3 \xrightarrow{u} \ell_1 \xrightarrow{x=2,c} +1 \xrightarrow{+1} \ell_2 \xrightarrow{c} \ell_3 \xrightarrow{x=2,c} +1

\begin{align*}
\ell_0 & \xrightarrow{1.3} \ell_0 \\
x & \xrightarrow{c} 1.3 \\
y & \xrightarrow{c} 1.3 \\
& \xrightarrow{u} \ell_1 \\
& \xrightarrow{u} \ell_3 \\
& \xrightarrow{0.7} \ell_3 \\
& \xrightarrow{c} \ell_3 \\
& \xrightarrow{c} \ell_3
\end{align*}

cost : 6.5 + 0 + 0 + 0 + 0.7

Weighted/priced timed automata \[^{[ALP01,BFH+01]}\]

\[\ell_0 + 5 \quad \xrightarrow{x \leq 2, c, y:=0} \quad \ell_1 \quad (y=0) \quad \xrightarrow{u} \quad \ell_2 \quad \xrightarrow{x=2, c} \quad \ell_3 \quad \xrightarrow{c} \quad \text{smiley}\]

\[\ell_0 \xrightarrow{1.3} \ell_0 \xrightarrow{c} \ell_1 \xrightarrow{u} \ell_3 \xrightarrow{0.7} \ell_3 \xrightarrow{c} \text{smiley}\]

\[
x\quad \begin{array}{c|c|c|c|c|c|c|c}
0 & 1.3 & 1.3 & 1.3 & 2 \\
y\quad \begin{array}{c|c|c|c|c|c|c|c}
0 & 1.3 & 0 & 0 & 0.7 & 7 \\
\end{array}
\end{array}
\]

\[
\text{cost :} \quad 6.5 + 0 + 0 + 0.7 + 7
\]

[^{[ALP01]} Alur, La Torre, Pappas. Optimal paths in weighted timed automata (HSCC’01).
^{[BFH+01]} Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata (HSCC’01).\]
Weighted/priced timed automata [ALP01,BFH+01]

\[
\ell_0 \xrightarrow{x \leq 2, c, y := 0} \ell_1 \xrightarrow{(y = 0)} \ell_2 \xrightarrow{x = 2, c} +10 \xrightarrow{u} \ell_2 \xrightarrow{x = 2, c} +1 \xrightarrow{+1} \text{smiley} \\
\ell_0 \xrightarrow{+5} \ell_1 \xrightarrow{+\ell_0} \ell_3 \xrightarrow{+1} \\
\ell_2 \xrightarrow{u} \ell_3 \xrightarrow{x = 2, c} +1 \\
\ell_1 \xrightarrow{\ell_2} \ell_3 \xrightarrow{\ell_1} \\
\ell_1 \xrightarrow{\ell_3} \text{smiley}
\]

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c}
\ell & l_0 & l_0 & c & l_1 & u & l_3 & 0.7 & l_3 & c \\
x & 0 & 1.3 & 1.3 & 1.3 & 2 & 0.7 & 7 & 14.2 \\
y & 0 & 1.3 & 0 & 0 & 0.7 & 7 & 14.2 \\
\hline
cost & 6.5 & + & 0 & + & 0 & + & 0.7 & + & 7 = 14.2
\end{array}
\]

**Weighted/priced timed automata** [ALP01,BFH+01]

**Question:** what is the optimal cost for reaching 🌸?
Weighted/priced timed automata \([\text{ALP01,BFH+01}]\)

\[\ell_0 + 5 \xrightarrow{x \leq 2, c, y := 0} \ell_1 (y=0) \xrightarrow{u} \ell_2 + 10 \xrightarrow{x=2, c} \ell_3 + 1 \xrightarrow{x=2, c} \text{smiley} \]

**Question:** what is the optimal cost for reaching \(\text{smiley} \)?

\[5t + 10(2 - t) + 1\]

[\text{ALP01}] Alur, La Torre, Pappas. Optimal paths in weighted timed automata \((HSCC'01)\).

[\text{BFH+01}] Behrmann, Fehnk, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata \((HSCC'01)\).
Weighted/priced timed automata \cite{ALP01,BFH+01}

\[ \ell_0 + 5 \xrightarrow{x \leq 2, c, y := 0} \ell_1 \xrightarrow{u} \ell_2 \xrightarrow{x = 2, c} \ell_3 \xrightarrow{u} \ell_1 \xrightarrow{\ell_3} +1 \xrightarrow{\ell_2} +10 \xrightarrow{\ell_0} \]

**Question:** what is the optimal cost for reaching \( \smiley \)?

\[
5t + 10(2 - t) + 1 , \quad 5t + (2 - t) + 7
\]

\cite{ALP01} Alur, La Torre, Pappas. Optimal paths in weighted timed automata (HSCC'01).
\cite{BFH+01} Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata (HSCC'01).
Weighted/priced timed automata [ALP01,BFH+01]

**Question:** what is the optimal cost for reaching 😊?

\[
\min \left( 5t + 10(2 - t) + 1, \ 5t + (2 - t) + 7 \right)
\]

[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata (HSCC'01).

Weighted/priced timed automata [ALP01,BFH+01]

Question: what is the optimal cost for reaching 😊?

\[
\inf \min_{0 \leq t \leq 2} \left( 5t + 10(2 - t) + 1, 5t + (2 - t) + 7 \right) = 9
\]

[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata (HSCC'01).
Weighted/priced timed automata [ALP01,BFH+01]

**Question:** What is the optimal cost for reaching \( \smiley \)?

\[
\inf_{0 \leq t \leq 2} \min \left( 5t + 10(2 - t) + 1 , 5t + (2 - t) + 7 \right) = 9
\]

\(~ \leadsto \text{strategy: leave immediately } \ell_0, \text{ go to } \ell_3, \text{ and wait there } 2 \text{ t.u.}\)