# Model-Checking Timed Temporal Logics

Patricia Bouyer

LSV – CNRS & ENS de Cachan – France

Oxford University Computing Laboratory – UK

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

# Outline

# Model-checking

system:

property:

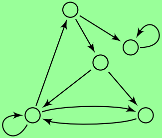

$\Rightarrow$

# Model-checking

system:

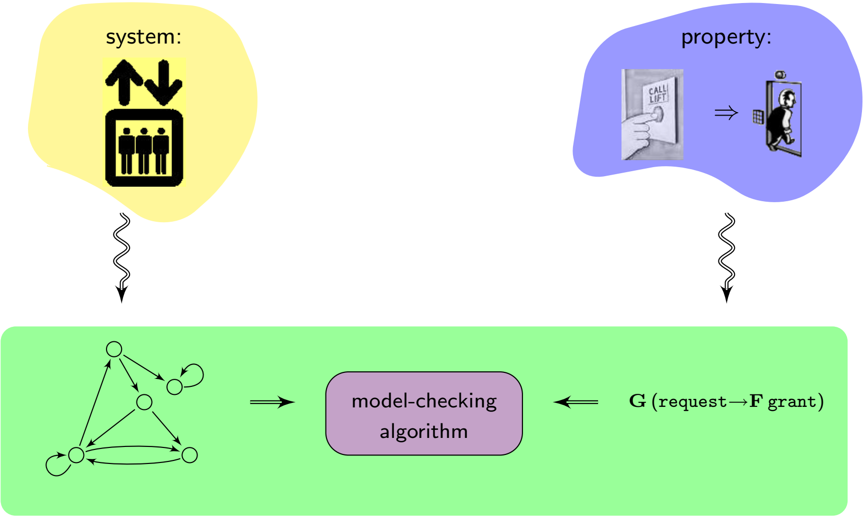
property:


$\Rightarrow$

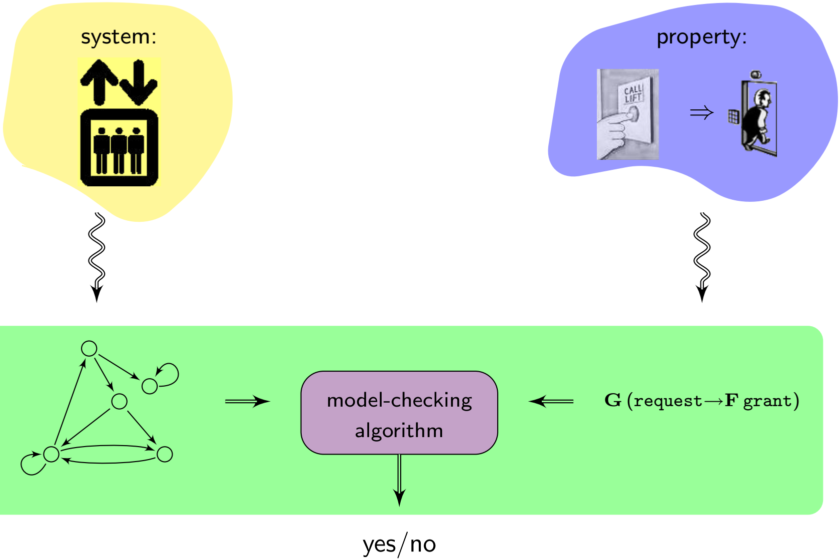


$\mathbf{G}\,(\texttt{request}\rightarrow\mathbf{F}\,\texttt{grant})$

# Model-checking



# Model-checking



# The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

$$\mathsf{LTL} \ni \varphi ::= p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi$$

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

# The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

$$\mathsf{LTL} \ni \varphi ::= p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi$$

$\models \quad \mathbf{X}$ 

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

# The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

$$\mathsf{LTL} \ni \varphi ::= p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi$$

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

# The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

$$\text{LTL} \ni \varphi ::= p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi$$

$\models$ $\mathbf{X}$ •

$\models$ • $\mathbf{U}$ •

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

# The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

$$\mathsf{LTL} \ni \varphi ::= p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi$$

$\models \quad \mathbf{X}$ •

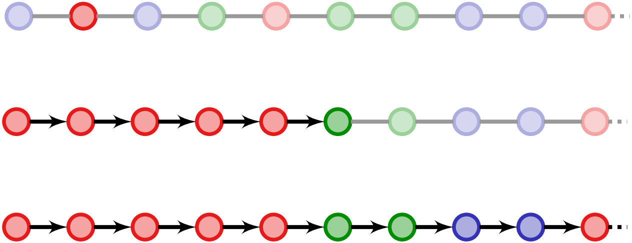
$\models \quad$ • $\mathbf{U}$ •

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

# The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

$$\mathsf{LTL} \ni \varphi ::= p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi$$



$\models \quad \mathbf{X}\,\bullet$

$\models \quad \bullet\,\mathbf{U}\,\bullet$

$\models \quad \mathbf{F}\,\bullet \;\equiv\; \mathtt{tt}\,\mathbf{U}\,\bullet$

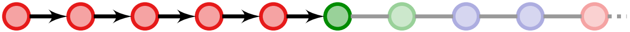
[Pnu77] Pnueli. The temporal logic of programs (FOCS'77).

# The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

$$\mathsf{LTL} \ni \varphi ::= p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi$$

 $\models \quad \mathbf{X}\,\bullet$

 $\models \quad \bullet\,\mathbf{U}\,\bullet$

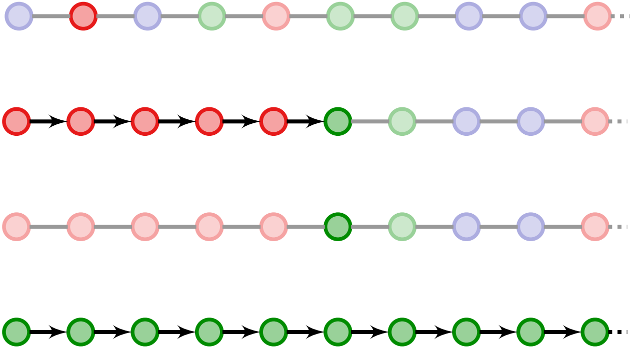
 $\models \quad \mathbf{F}\,\bullet \equiv \mathtt{tt}\,\mathbf{U}\,\bullet$

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

# The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

$$\mathsf{LTL} \ni \varphi \ ::= \ p \ \mid \ \varphi \wedge \varphi \ \mid \ \varphi \vee \varphi \ \mid \ \neg\varphi \ \mid \ \mathbf{X}\,\varphi \ \mid \ \varphi\,\mathbf{U}\,\varphi$$



$\models \quad \mathbf{X}\,\bullet$

$\models \quad \bullet\,\mathbf{U}\,\bullet$

$\models \quad \mathbf{F}\,\bullet \ \equiv \ \mathtt{tt}\,\mathbf{U}\,\bullet$

$\models \quad \mathbf{G}\,\bullet \ \equiv \ \neg\mathbf{F}\,\neg\bullet$

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

# The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

$$\mathsf{LTL} \ni \varphi ::= p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi$$

- response property:

$$\mathbf{G}\,(\bullet \to \mathbf{F}\,\bullet)$$

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

# The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

$$\mathsf{LTL} \ni \varphi ::= p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi$$

- response property:

$$\mathbf{G}\,(\bullet \rightarrow \mathbf{F}\,\bullet)$$

- liveness property:

$$\mathbf{G}\,\mathbf{F}\,\bullet$$

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

# The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

$$\text{LTL} \ni \varphi ::= p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi$$

- response property:

$$\mathbf{G}\,(\bullet \rightarrow \mathbf{F}\,\bullet)$$

- liveness property:

$$\mathbf{G}\,\mathbf{F}\,\bullet$$

- safety property:

$$\mathbf{G}\,\neg\bullet$$

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

# The untimed (linear-time) framework

Linear-time temporal logic [Pnu77]

$$\mathsf{LTL} \ni \varphi ::= p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi$$

- response property:

$$\mathbf{G}\,({\color{red}\bullet} \to \mathbf{F}\,{\color{green}\bullet})$$

- liveness property:

$$\mathbf{G}\,\mathbf{F}\,{\color{green}\bullet}$$

- safety property:

$$\mathbf{G}\,\neg{\color{red}\bullet}$$

- a more complex property:

$$({\color{green}\bullet} \wedge (\mathbf{F}\,{\color{blue}\bullet} \vee \mathbf{G}\,{\color{orange}\bullet}))\,\mathbf{U}\,{\color{red}\bullet}$$

[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

# Metric Temporal Logic (MTL)

[Koy90]

$$\mathsf{MTL} \ni \varphi ::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \, \mathbf{U}_I \, \varphi$$

where $I$ is an interval with integral bounds.

[Koy90] Koymans. Specifying real-time properties with metric temporal logic (Real-time systems, 1990).

# Metric Temporal Logic (MTL)

[Koy90]

$$\mathsf{MTL} \ni \varphi ::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi\, \mathbf{U}_I\, \varphi$$

where $I$ is an interval with integral bounds.

- This is a timed extension of LTL

[Koy90] Koymans. Specifying real-time properties with metric temporal logic (Real-time systems, 1990).

# Metric Temporal Logic (MTL)

[Koy90]

$$\mathsf{MTL} \ni \varphi ::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \, \mathbf{U}_I \, \varphi$$

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

[Koy90] Koymans. Specifying real-time properties with metric temporal logic (Real-time systems, 1990).

# Metric Temporal Logic (MTL)

[Koy90]

$$\text{MTL} \ni \varphi ::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \, \mathbf{U}_I \, \varphi$$

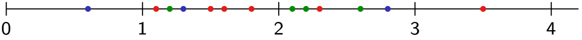
where $I$ is an interval with integral bounds.

- 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

[Koy90] Koymans. Specifying real-time properties with metric temporal logic (Real-time systems, 1990).

# The pointwise semantics

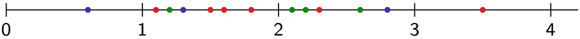
MTL formulas are interpreted over timed words:




$$(\bullet, .6)(\bullet, 1.1)(\bullet, 1.2)(\bullet, 1.3)\ldots$$

# The pointwise semantics

MTL formulas are interpreted over timed words:

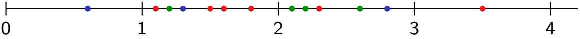



$$({\color{blue}\bullet}, .6)({\color{red}\bullet}, 1.1)({\color{green}\bullet}, 1.2)({\color{blue}\bullet}, 1.3)\dots$$

☞ the system is observed only when actions happen

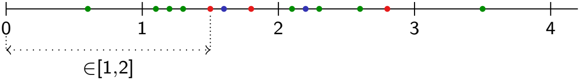
# The pointwise semantics

MTL formulas are interpreted over timed words:




$$(\bullet, .6)(\bullet, 1.1)(\bullet, 1.2)(\bullet, 1.3)\ldots$$

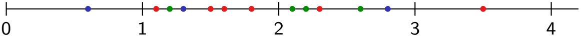
☞ the system is observed only when actions happen




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

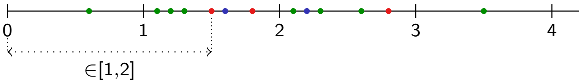
# The pointwise semantics

MTL formulas are interpreted over timed words:

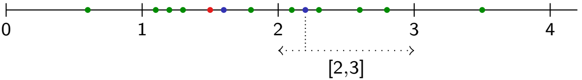



$$(\bullet, .6)(\bullet, 1.1)(\bullet, 1.2)(\bullet, 1.3)\dots$$

☞ the system is observed only when actions happen



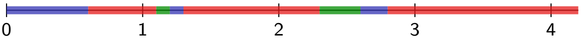

$\models \bullet\ \mathbf{U}_{[1,2]}\ \bullet$




$\not\models\ \mathbf{G}_{[2,3]}\ \bullet$

# The continuous semantics

MTL formulas are interpreted over (finitely variable) signals:

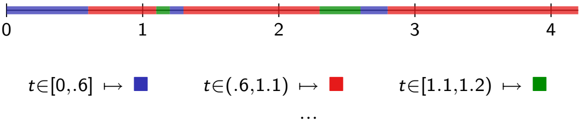



$t \in [0,.6] \mapsto$ ■ $t \in (.6,1.1) \mapsto$ ■ $t \in [1.1,1.2) \mapsto$ ■

…

# The continuous semantics

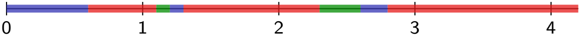
MTL formulas are interpreted over (finitely variable) signals:




☞ the system is observed continuously

# The continuous semantics

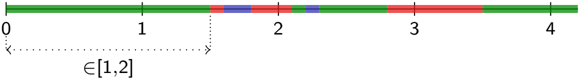
MTL formulas are interpreted over (finitely variable) signals:




$t\in[0,.6] \mapsto$ ■ $\quad t\in(.6,1.1) \mapsto$ ■ $\quad t\in[1.1,1.2) \mapsto$ ■

...

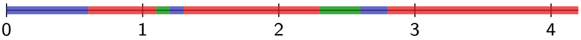
☞ the system is observed continuously




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

# The continuous semantics

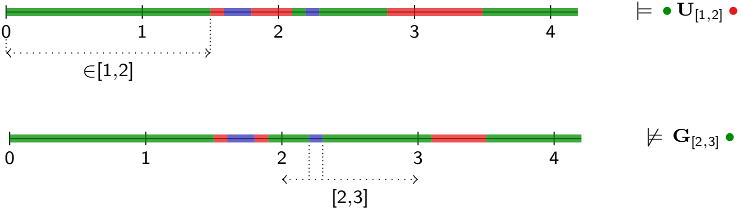
MTL formulas are interpreted over (finitely variable) signals:




$t\in[0,.6] \mapsto$ ■ $\quad t\in(.6,1.1) \mapsto$ ■ $\quad t\in[1.1,1.2) \mapsto$ ■

...

☞ the system is observed continuously



# Some examples

- "Every problem is followed within 56 time units by an alarm"

$$\mathbf{G}\,(\texttt{problem} \rightarrow \mathbf{F}_{\leqslant 56}\,\texttt{alarm})$$

# Some examples

- "Every problem is followed within 56 time units by an alarm"

$$\mathbf{G}\,(\texttt{problem} \rightarrow \mathbf{F}_{\leqslant 56}\,\texttt{alarm})$$

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

$$\mathbf{G}\,(\texttt{problem} \rightarrow (\mathbf{F}_{\leqslant 15}\,\texttt{repair} \vee \mathbf{G}_{[12,15)}\,\texttt{alarm}))$$

# Some examples

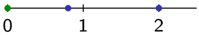
- "Every problem is followed within 56 time units by an alarm"

$$\mathbf{G}\left(\texttt{problem} \rightarrow \mathbf{F}_{\leqslant 56}\, \texttt{alarm}\right)$$

- "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"

$$\mathbf{G}\left(\texttt{problem} \rightarrow \left(\mathbf{F}_{\leqslant 15}\, \texttt{repair} \vee \mathbf{G}_{[12,15)}\, \texttt{alarm}\right)\right)$$

- $\mathbf{F}_{=2}\, \texttt{repair}$ *vs* $\mathbf{F}_{=1}\left(\mathbf{F}_{=1}\, \texttt{repair}\right)$




# Some examples

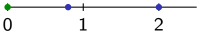
- "Every problem is followed within 56 time units by an alarm"

$$\mathbf{G}\,(\texttt{problem} \rightarrow \mathbf{F}_{\leqslant 56}\,\texttt{alarm})$$

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

$$\mathbf{G}\,(\texttt{problem} \rightarrow (\mathbf{F}_{\leqslant 15}\,\texttt{repair} \vee \mathbf{G}_{[12,15)}\,\texttt{alarm}))$$

- $\mathbf{F}_{=2}\,\texttt{repair}$ *vs* $\mathbf{F}_{=1}\,(\mathbf{F}_{=1}\,\texttt{repair})$




$\models \mathbf{F}_{=2}\,\bullet \quad \not\models \mathbf{F}_{=1}\,(\mathbf{F}_{=1}\,\bullet)$

# Some examples

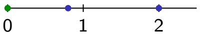
- "Every problem is followed within 56 time units by an alarm"

$$\mathbf{G}\,(\texttt{problem} \rightarrow \mathbf{F}_{\leqslant 56}\,\texttt{alarm})$$

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

$$\mathbf{G}\,(\texttt{problem} \rightarrow (\mathbf{F}_{\leqslant 15}\,\texttt{repair} \vee \mathbf{G}_{[12,15)}\,\texttt{alarm}))$$

- $\mathbf{F}_{=2}\,\texttt{repair}$ *vs* $\mathbf{F}_{=1}\,(\mathbf{F}_{=1}\,\texttt{repair})$




$\models \mathbf{F}_{=2}\,\bullet$ $\not\models \mathbf{F}_{=1}\,(\mathbf{F}_{=1}\,\bullet)$


$\models \mathbf{F}_{=2}\,\bullet$ $\models \mathbf{F}_{=1}\,(\mathbf{F}_{=1}\,\bullet)$

# Some examples

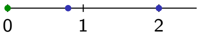
- "Every problem is followed within 56 time units by an alarm"

$$\mathbf{G}\,(\texttt{problem} \to \mathbf{F}_{\leqslant 56}\,\texttt{alarm})$$

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

$$\mathbf{G}\,(\texttt{problem} \to (\mathbf{F}_{\leqslant 15}\,\texttt{repair} \vee \mathbf{G}_{[12,15)}\,\texttt{alarm}))$$

- $\mathbf{F}_{=2}\,\texttt{repair}$ *vs* $\mathbf{F}_{=1}\,(\mathbf{F}_{=1}\,\texttt{repair})$




$\models \mathbf{F}_{=2}\,\bullet$ $\not\models \mathbf{F}_{=1}\,(\mathbf{F}_{=1}\,\bullet)$


$\models \mathbf{F}_{=2}\,\bullet$ $\models \mathbf{F}_{=1}\,(\mathbf{F}_{=1}\,\bullet)$

  - in the pointwise semantics, $\mathbf{F}_{=2}\,\bullet \not\equiv \mathbf{F}_{=1}\,\mathbf{F}_{=1}\,\bullet$
  - in the continuous semantics, $\mathbf{F}_{=2}\,\bullet \equiv \mathbf{F}_{=1}\,\mathbf{F}_{=1}\,\bullet$

# Some further extensions

- Timed Propositional Temporal Logic (TPTL) [AH89]

TPTL = LTL + clock variables + clock constraints

[AH89] Alur, Henzinger. A really temporal logic (FOCS'89).

# Some further extensions

- Timed Propositional Temporal Logic (TPTL) [AH89]

TPTL = LTL + clock variables + clock constraints

$$\mathbf{G}\,(\texttt{problem} \rightarrow \mathbf{F}_{\leqslant 56}\,\texttt{alarm}) \quad \equiv \quad \mathbf{G}\,(\texttt{problem} \rightarrow x.\mathbf{F}\,(\texttt{alarm} \wedge x \leqslant 56))$$

[AH89] Alur, Henzinger. A really temporal logic (FOCS'89).

# Some further extensions

- Timed Propositional Temporal Logic (TPTL) [AH89]

TPTL = LTL + clock variables + clock constraints

$$\mathbf{G}\,(\texttt{problem} \to \mathbf{F}_{\leqslant 56}\,\texttt{alarm}) \quad \equiv \quad \mathbf{G}\,(\texttt{problem} \to x.\mathbf{F}\,(\texttt{alarm} \wedge x \leqslant 56))$$

$$\mathbf{G}\,(\texttt{problem} \to x.\mathbf{F}\,(\texttt{alarm} \wedge \mathbf{F}\,(\texttt{failsafe} \wedge x \leqslant 56)))$$

[AH89] Alur, Henzinger. A really temporal logic (FOCS'89).

# Some further extensions

- Timed Propositional Temporal Logic (TPTL) [AH89]

TPTL = LTL + clock variables + clock constraints

$$\mathbf{G}\,(\texttt{problem} \to \mathbf{F}_{\leqslant 56}\,\texttt{alarm}) \quad \equiv \quad \mathbf{G}\,(\texttt{problem} \to x.\mathbf{F}\,(\texttt{alarm} \wedge x \leqslant 56))$$

$$\mathbf{G}\,(\texttt{problem} \to x.\mathbf{F}\,(\texttt{alarm} \wedge \mathbf{F}\,(\texttt{failsafe} \wedge x \leqslant 56)))$$

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

[AH89] Alur, Henzinger. A really temporal logic (FOCS'89).
[AH92] Alur, Henzinger. Back to the future: towards a theory of timed regular languages (FOCS'92).

# Some further extensions

- Timed Propositional Temporal Logic (TPTL) [AH89]

TPTL = LTL + clock variables + clock constraints

$$\mathbf{G}\,(\mathtt{problem} \rightarrow \mathbf{F}_{\leqslant 56}\,\mathtt{alarm}) \quad \equiv \quad \mathbf{G}\,(\mathtt{problem} \rightarrow x.\mathbf{F}\,(\mathtt{alarm} \wedge x \leqslant 56))$$

$$\mathbf{G}\,(\mathtt{problem} \rightarrow x.\mathbf{F}\,(\mathtt{alarm} \wedge \mathbf{F}\,(\mathtt{failsafe} \wedge x \leqslant 56)))$$

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

$$\mathbf{G}\,(\mathtt{alarm} \rightarrow \mathbf{F}^{-1}_{\leqslant 56}\,\mathtt{problem})$$

[AH89] Alur, Henzinger. A really temporal logic (FOCS'89).
[AH92] Alur, Henzinger. Back to the future: towards a theory of timed regular languages (FOCS'92).

# A note on the expressiveness

**Theorem**

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

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

# 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].

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

$$\mathbf{G}\,(\bullet \to x.\mathbf{F}\,(\bullet \wedge \mathbf{F}\,(\bullet \wedge x \leqslant 2)))$$

cannot be expressed in MTL.

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

$$\mathbf{G}\,(\bullet \rightarrow x.\mathbf{F}\,(\bullet \wedge \mathbf{F}\,(\bullet \wedge x \leqslant 2)))$$

cannot be expressed in MTL.

- This is true in the pointwise semantics.

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

$$\mathbf{G}\,(\bullet \to x.\mathbf{F}\,(\bullet \wedge \mathbf{F}\,(\bullet \wedge x \leqslant 2)))$$

cannot be expressed in MTL.

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

[Kam68] Kamp. Tense logic and the theory of linear order (PhD Thesis UCLA 1968).
[GPSS80] Gabbay, Pnueli, Shelah, Stavi. On the temporal analysis of fairness (POPL'80).
[BCM05] Bouyer, Chevalier, Markey. On the expressiveness of MTL and TPTL (FSTTCS'05).

The TPTL formula

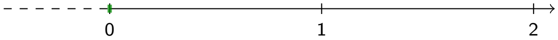
$$\mathbf{G}\left(\bullet \rightarrow x.\mathbf{F}\left(\bullet \wedge \mathbf{F}\left(\bullet \wedge x \leqslant 2\right)\right)\right)$$

can be expressed in MTL in the continuous semantics

The TPTL formula

$$\mathbf{G}\,(\bullet \rightarrow x.\mathbf{F}\,(\bullet \wedge \mathbf{F}\,(\bullet \wedge x \leqslant 2)))$$

can be expressed in MTL in the continuous semantics

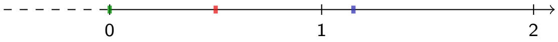



$$\mathbf{G}\ \bullet \ \rightarrow \ \left\{ \right.$$

The TPTL formula

$$\mathbf{G}\left({\color{green}\bullet} \to x.\mathbf{F}\left({\color{red}\bullet} \wedge \mathbf{F}\left({\color{blue}\bullet} \wedge x \leqslant 2\right)\right)\right)$$

can be expressed in MTL in the continuous semantics

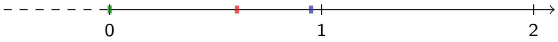



$$\mathbf{G}\ {\color{green}\bullet} \to \left\{ \begin{array}{l} \mathbf{F}_{\leqslant 1}\ {\color{red}\bullet} \ \wedge\ \mathbf{F}_{[1,2]}\ {\color{blue}\bullet} \\ \\ \\ \end{array} \right.$$

The TPTL formula

$$\mathbf{G}\,(\textcolor{green}{\bullet} \rightarrow x.\mathbf{F}\,(\textcolor{red}{\bullet} \wedge \mathbf{F}\,(\textcolor{blue}{\bullet} \wedge x \leqslant 2)))$$

can be expressed in MTL in the continuous semantics

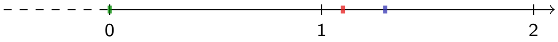



$$\mathbf{G}\ \textcolor{green}{\bullet} \ \rightarrow \ \left\{ \begin{array}{ll} & \mathbf{F}_{\leqslant 1}\ \textcolor{red}{\bullet} \ \wedge \ \mathbf{F}_{[1,2]}\ \textcolor{blue}{\bullet} \\ \vee & \\ & \mathbf{F}_{\leqslant 1}\left(\ \textcolor{red}{\bullet} \ \wedge \ \mathbf{F}_{\leqslant 1}\,\textcolor{blue}{\bullet}\ \right) \\ & \\ & \end{array} \right.$$

The TPTL formula

$$\mathbf{G}\left({\color{green}\bullet} \rightarrow x.\mathbf{F}\left({\color{red}\bullet} \wedge \mathbf{F}\left({\color{blue}\bullet} \wedge x \leqslant 2\right)\right)\right)$$

can be expressed in MTL in the continuous semantics

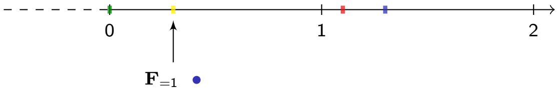



$$\mathbf{G}\ {\color{green}\bullet} \rightarrow \left\{\begin{array}{ll} & \mathbf{F}_{\leqslant 1}\ {\color{red}\bullet} \ \wedge\ \mathbf{F}_{[1,2]}\ {\color{blue}\bullet} \\ \vee & \\ & \mathbf{F}_{\leqslant 1}\left(\ {\color{red}\bullet} \ \wedge\ \mathbf{F}_{\leqslant 1}\,{\color{blue}\bullet}\ \right) \\ \vee & \\ & \end{array}\right.$$

The TPTL formula

$$\mathbf{G}\,(\textcolor{green}{\bullet} \rightarrow x.\mathbf{F}\,(\textcolor{red}{\bullet} \wedge \mathbf{F}\,(\textcolor{blue}{\bullet} \wedge x \leqslant 2)))$$

can be expressed in MTL in the continuous semantics

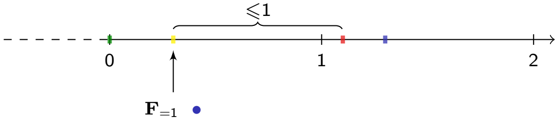



$$\mathbf{G}\ \textcolor{green}{\bullet}\ \rightarrow \begin{cases} \quad \mathbf{F}_{\leqslant 1}\ \textcolor{red}{\bullet}\ \wedge\ \mathbf{F}_{[1,2]}\ \textcolor{blue}{\bullet} \\ \vee \\ \quad \mathbf{F}_{\leqslant 1}\,(\ \textcolor{red}{\bullet}\ \wedge\ \mathbf{F}_{\leqslant 1}\,\textcolor{blue}{\bullet}\ ) \\ \vee \\ \end{cases}$$

The TPTL formula

$$\mathbf{G}\left(\textcolor{green}{\bullet} \rightarrow x.\mathbf{F}\left(\textcolor{red}{\bullet} \wedge \mathbf{F}\left(\textcolor{blue}{\bullet} \wedge x \leqslant 2\right)\right)\right)$$

can be expressed in MTL in the continuous semantics

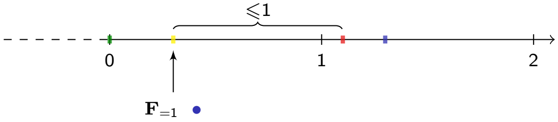



$$\mathbf{G}\ \textcolor{green}{\bullet} \rightarrow \left\{ \begin{array}{l} \quad \mathbf{F}_{\leqslant 1}\ \textcolor{red}{\bullet} \ \wedge\ \mathbf{F}_{[1,2]}\ \textcolor{blue}{\bullet} \\ \vee \\ \quad \mathbf{F}_{\leqslant 1}\left(\ \textcolor{red}{\bullet}\ \wedge\ \mathbf{F}_{\leqslant 1}\textcolor{blue}{\bullet}\ \right) \\ \vee \\ \\ \end{array} \right.$$

The TPTL formula

$$\mathbf{G}\left(\textcolor{green}{\bullet} \rightarrow x.\mathbf{F}\left(\textcolor{red}{\bullet} \wedge \mathbf{F}\left(\textcolor{blue}{\bullet} \wedge x \leqslant 2\right)\right)\right)$$

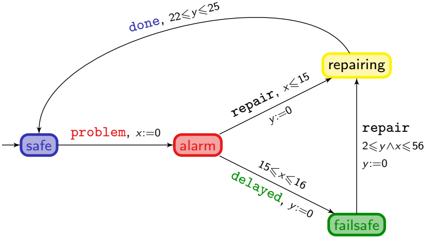
can be expressed in MTL in the continuous semantics



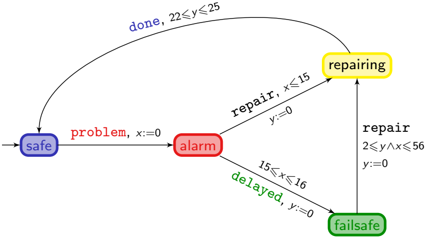

$$\mathbf{G}\ \textcolor{green}{\bullet} \rightarrow \begin{cases} \mathbf{F}_{\leqslant 1}\ \textcolor{red}{\bullet} \wedge \mathbf{F}_{[1,2]}\ \textcolor{blue}{\bullet} \\ \vee \\ \mathbf{F}_{\leqslant 1}\left(\textcolor{red}{\bullet} \wedge \mathbf{F}_{\leqslant 1} \textcolor{blue}{\bullet}\right) \\ \vee \\ \mathbf{F}_{\leqslant 1}\left(\mathbf{F}_{\leqslant 1}\ \textcolor{red}{\bullet} \wedge \mathbf{F}_{=1}\ \textcolor{blue}{\bullet}\right) \end{cases}$$

# Outline

# Timed automata

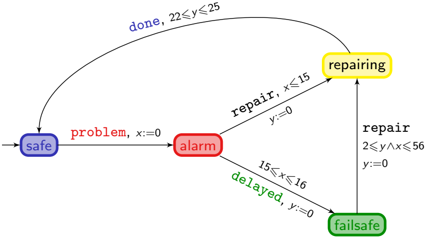


# Timed automata



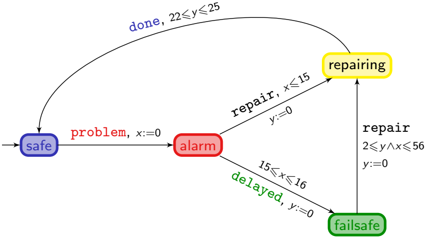

| | safe |
|---|---|
| x | 0 |
| y | 0 |

# Timed automata



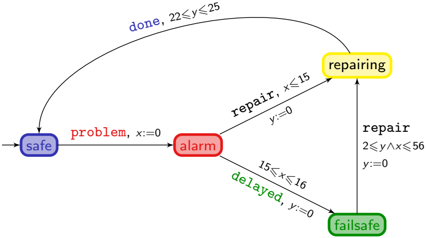

| | safe | $\xrightarrow{23}$ | safe |
|---|---|---|---|
| x | 0 | | 23 |
| y | 0 | | 23 |

# Timed automata



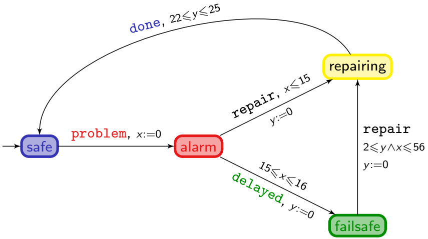

| | safe | $\xrightarrow{23}$ | safe | $\xrightarrow{\texttt{problem}}$ | alarm |
|---|---|---|---|---|---|
| x | 0 | | 23 | | 0 |
| y | 0 | | 23 | | 23 |

# Timed automata



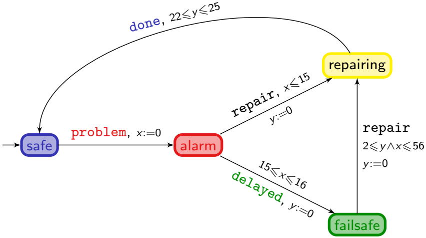

| | safe | $\xrightarrow{23}$ | safe | $\xrightarrow{\texttt{problem}}$ | alarm | $\xrightarrow{15.6}$ | alarm |
|---|---|---|---|---|---|---|---|
| x | 0 | | 23 | | 0 | | 15.6 |
| y | 0 | | 23 | | 23 | | 38.6 |

# Timed automata



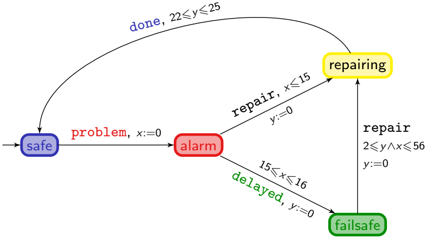

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | safe | $\xrightarrow{23}$ | safe | $\xrightarrow{\texttt{problem}}$ | alarm | $\xrightarrow{15.6}$ | alarm | $\xrightarrow{\texttt{delayed}}$ | failsafe |
| x | 0 | | 23 | | 0 | | 15.6 | | 15.6 |
| y | 0 | | 23 | | 23 | | 38.6 | | 0 |

# Timed automata



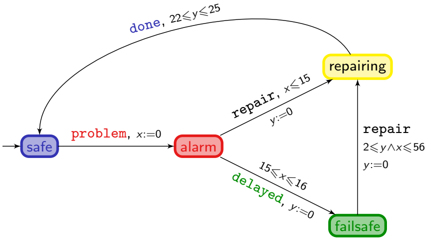

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | safe | $\xrightarrow{23}$ | safe | $\xrightarrow{\texttt{problem}}$ | alarm | $\xrightarrow{15.6}$ | alarm | $\xrightarrow{\texttt{delayed}}$ | failsafe | $\xrightarrow{2.3}$ | failsafe |
| x | 0 | | 23 | | 0 | | 15.6 | | 15.6 | | 17.9 |
| y | 0 | | 23 | | 23 | | 38.6 | | 0 | | 2.3 |

# Timed automata



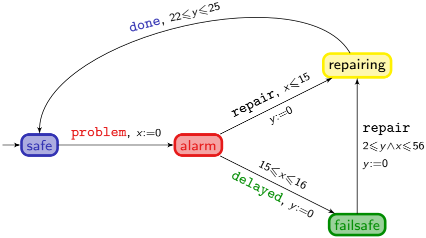

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | safe | $\xrightarrow{23}$ | safe | $\xrightarrow{\texttt{problem}}$ | alarm | $\xrightarrow{15.6}$ | alarm | $\xrightarrow{\texttt{delayed}}$ | failsafe | $\xrightarrow{2.3}$ | failsafe | $\xrightarrow{\texttt{repair}}$ | reparation |
| x | 0 | | 23 | | 0 | | 15.6 | | 15.6 | | 17.9 | | 17.9 |
| y | 0 | | 23 | | 23 | | 38.6 | | 0 | | 2.3 | | 0 |

# Timed automata



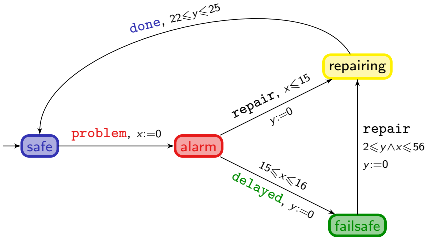

| | safe | $\xrightarrow{23}$ | safe | $\xrightarrow{\texttt{problem}}$ | alarm | $\xrightarrow{15.6}$ | alarm | $\xrightarrow{\texttt{delayed}}$ | failsafe | $\xrightarrow{2.3}$ | failsafe | $\xrightarrow{\texttt{repair}}$ | reparation | $\xrightarrow{22.1}$ | reparation |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| x | 0 | | 23 | | 0 | | 15.6 | | 15.6 | | 17.9 | | 17.9 | | 40 |
| y | 0 | | 23 | | 23 | | 38.6 | | 0 | | 2.3 | | 0 | | 22.1 |

# Timed automata




| | safe | $\xrightarrow{23}$ | safe | $\xrightarrow{\texttt{problem}}$ | alarm | $\xrightarrow{15.6}$ | alarm | $\xrightarrow{\texttt{delayed}}$ | failsafe | $\xrightarrow{2.3}$ | failsafe | $\xrightarrow{\texttt{repair}}$ | reparation | $\xrightarrow{22.1}$ | reparation | $\xrightarrow{\texttt{done}}$ | safe |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| x | 0 | | 23 | | 0 | | 15.6 | | 15.6 | | 17.9 | | 17.9 | | 40 | | 40 |
| y | 0 | | 23 | | 23 | | 38.6 | | 0 | | 2.3 | | 0 | | 22.1 | | 22.1 |

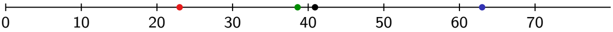
# Timed automata



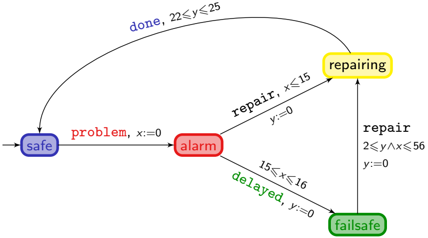

| | safe | $\xrightarrow{23}$ | safe | $\xrightarrow{\texttt{problem}}$ | alarm | $\xrightarrow{15.6}$ | alarm | $\xrightarrow{\texttt{delayed}}$ | failsafe | $\xrightarrow{2.3}$ | failsafe | $\xrightarrow{\texttt{repair}}$ | reparation | $\xrightarrow{22.1}$ | reparation | $\xrightarrow{\texttt{done}}$ | safe |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| x | 0 | | 23 | | 0 | | 15.6 | | 15.6 | | 17.9 | | 17.9 | | 40 | | 40 |
| y | 0 | | 23 | | 23 | | 38.6 | | 0 | | 2.3 | | 0 | | 22.1 | | 22.1 |

Can be viewed:

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



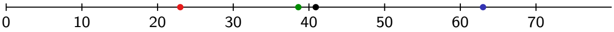
# Timed automata



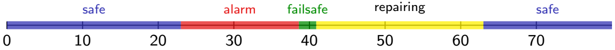

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | safe | $\xrightarrow{23}$ | safe | $\xrightarrow{\texttt{problem}}$ | alarm | $\xrightarrow{15.6}$ | alarm | $\xrightarrow{\texttt{delayed}}$ | failsafe | $\xrightarrow{2.3}$ | failsafe | $\xrightarrow{\texttt{repair}}$ | reparation | $\xrightarrow{22.1}$ | reparation | $\xrightarrow{\texttt{done}}$ safe |
| x | 0 | | 23 | | 0 | | 15.6 | | 15.6 | | 17.9 | | 17.9 | | 40 | 40 |
| y | 0 | | 23 | | 23 | | 38.6 | | 0 | | 2.3 | | 0 | | 22.1 | 22.1 |

Can be viewed:

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




- as the signal



# Basic result on timed automata

**Theorem**

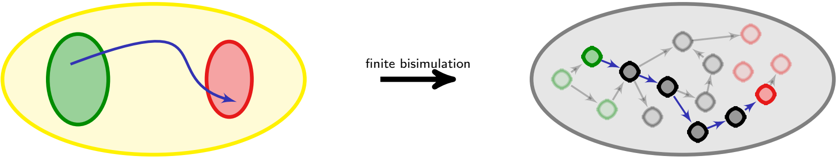
The reachability problem is decidable (and PSPACE-complete) for timed automata [AD94].

[AD94] Alur, Dill. A theory of timed automata (TCS, 1994).

# Basic result on timed automata

**Theorem**

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




timed automaton

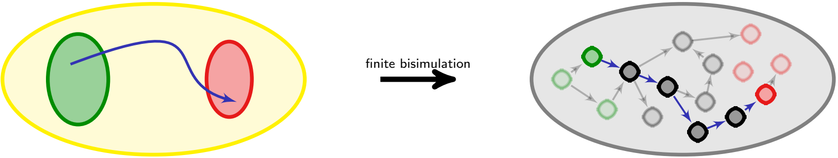
large (but finite) automaton
(region automaton)

[AD94] Alur, Dill. A theory of timed automata (TCS, 1994).

# Basic result on timed automata

**Theorem**

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




timed automaton

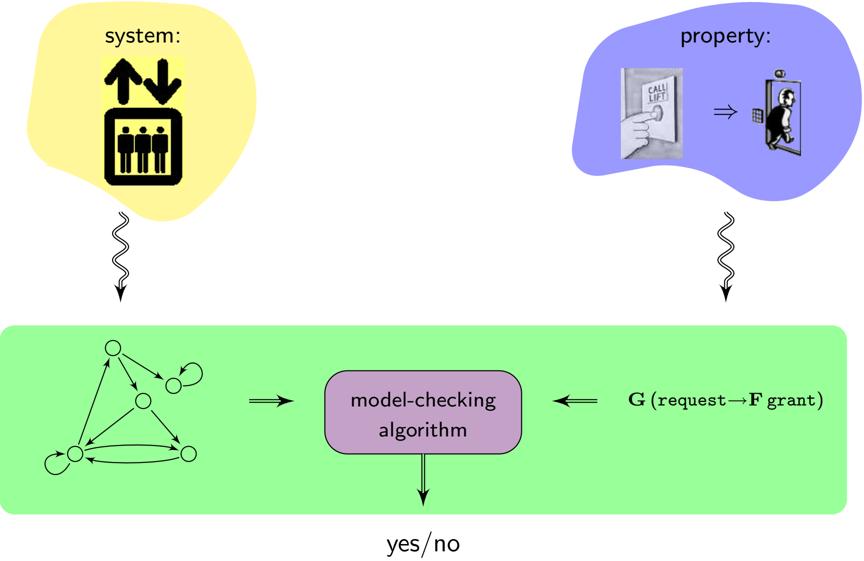
large (but finite) automaton
(region automaton)

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

[AD94] Alur, Dill. A theory of timed automata (TCS, 1994).
[ACD93] Alur, Courcoubetis, Dill. Model-checking in dense real-time (I&C, 1993).

# Outline

# Back to the model-checking problem

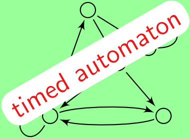


# Back to the model-checking problem

system:

property:

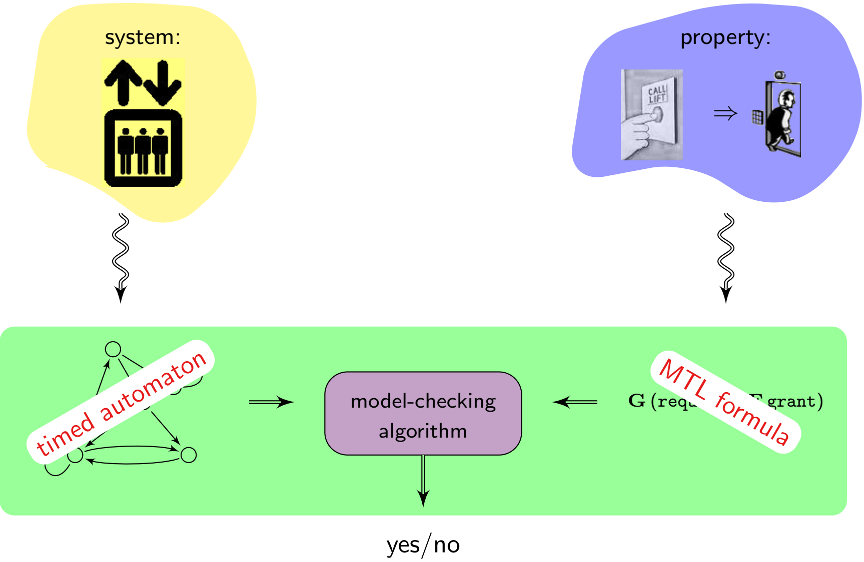

$\Rightarrow$




$\Longrightarrow$ model-checking algorithm $\Longleftarrow$ $\mathbf{G}\,(\texttt{request} \rightarrow \mathbf{F}\,\texttt{grant})$

yes/no

# Back to the model-checking problem



# Results

**Theorem**

Over finite runs, the model-checking problem is:

| | pointwise sem. | continuous sem. |
|---|---|---|
| MTL | decidable, NPR [OW05] | undecidable [AFH96] |
| MTL+Past | undecidable | undecidable |
| TPTL | undecidable [AH94] | undecidable [AH94] |

[OW05] Ouaknine, Worrell. On the decidability of metric temporal logic (LICS'05).
[AFH96] Alur, Feder, Henzinger. The benefits of relaxing punctuality (Journal of the ACM, 1996).
[AH94] Alur, Henzinger. A really temporal logic (Journal of the ACM, 1994).

# Results

**Theorem**

Over finite runs, the model-checking problem is:

| | pointwise sem. | continuous sem. |
|---|---|---|
| MTL | decidable, NPR [OW05] | undecidable [AFH96] |
| MTL+Past | undecidable | undecidable |
| TPTL | undecidable [AH94] | undecidable [AH94] |

- Model-checking linear-time timed temporal logics is hard!

[OW05] Ouaknine, Worrell. On the decidability of metric temporal logic (LICS'05).
[AFH96] Alur, Feder, Henzinger. The benefits of relaxing punctuality (Journal of the ACM, 1996).
[AH94] Alur, Henzinger. A really temporal logic (Journal of the ACM, 1994).

# Results

**Theorem**

Over finite runs, the model-checking problem is:

| | pointwise sem. | continuous sem. |
|---|---|---|
| MTL | decidable, NPR [OW05] | undecidable [AFH96] |
| MTL+Past | undecidable | undecidable |
| TPTL | undecidable [AH94] | undecidable [AH94] |

- 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).
[AFH96] Alur, Feder, Henzinger. The benefits of relaxing punctuality (Journal of the ACM, 1996).
[AH94] Alur, Henzinger. A really temporal logic (Journal of the ACM, 1994).

# Results

**Theorem**

Over finite runs, the model-checking problem is:

| | pointwise sem. | continuous sem. |
|---|---|---|
| MTL | decidable, NPR [OW05] | undecidable [AFH96] |
| MTL+Past | undecidable | undecidable |
| TPTL | undecidable [AH94] | undecidable [AH94] |

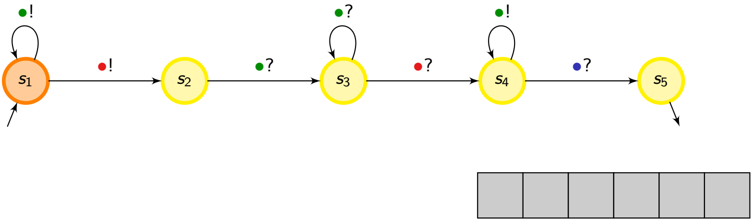
- Model-checking linear-time timed temporal logics is hard!
- The gap between branching-time and linear-time dramatically increases in the timed framework...
  (reminder: model-checking TCTL is PSPACE-complete)

☞ we will explain this high complexity, following [Che07]

[OW05] Ouaknine, Worrell. On the decidability of metric temporal logic (LICS'05).
[AFH96] Alur, Feder, Henzinger. The benefits of relaxing punctuality (Journal of the ACM, 1996).
[AH94] Alur, Henzinger. A really temporal logic (Journal of the ACM, 1994).
[Che07] Chevalier. Logiques pour les systèmes temporisés : contrôle et expressivité (PhD Thesis ENS Cachan, June 2007).

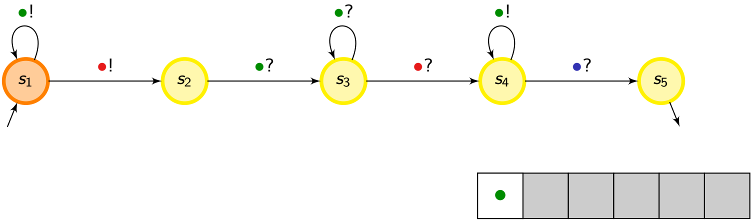
# A short visit to channel machines (1)

A channel machine = a finite automaton + a FIFO channel



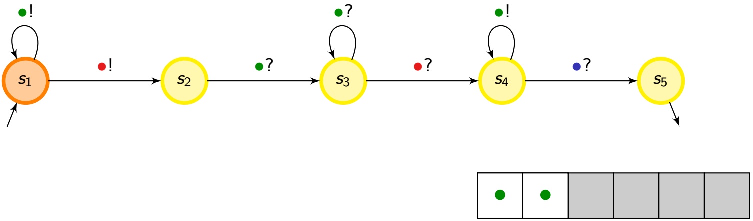
# A short visit to channel machines (1)

A channel machine $=$ a finite automaton $+$ a FIFO channel



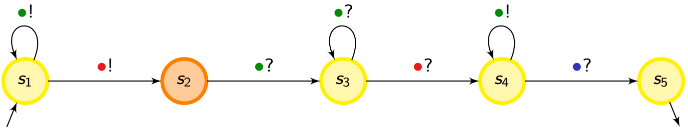
# A short visit to channel machines (1)

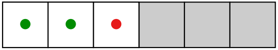
A channel machine $=$ a finite automaton $+$ a FIFO channel



# A short visit to channel machines (1)

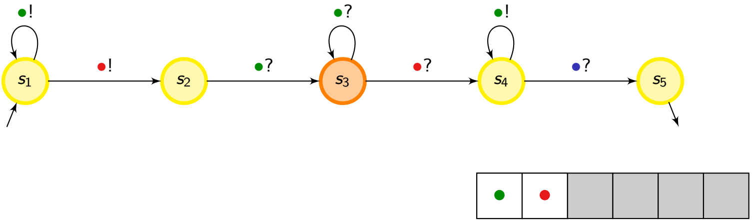
A channel machine $=$ a finite automaton $+$ a FIFO channel



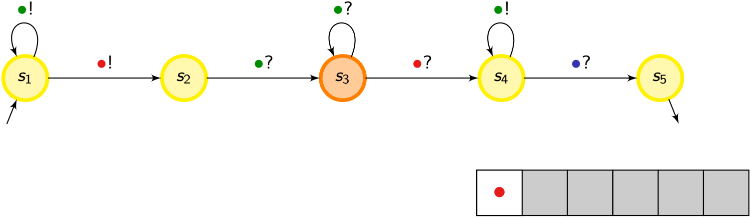
# A short visit to channel machines (1)

A channel machine = a finite automaton + a FIFO channel



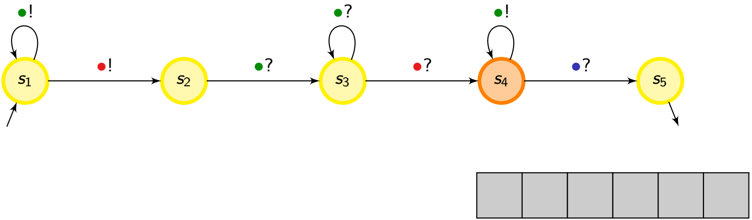
# A short visit to channel machines (1)

A channel machine = a finite automaton + a FIFO channel



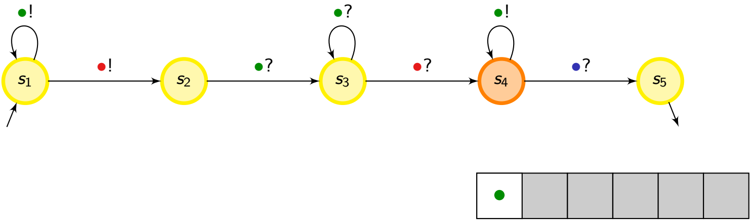
# A short visit to channel machines (1)

A channel machine = a finite automaton + a FIFO channel



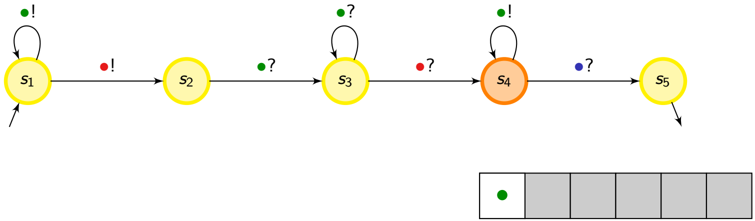
# A short visit to channel machines (1)

A channel machine $=$ a finite automaton $+$ a FIFO channel



# A short visit to channel machines (1)

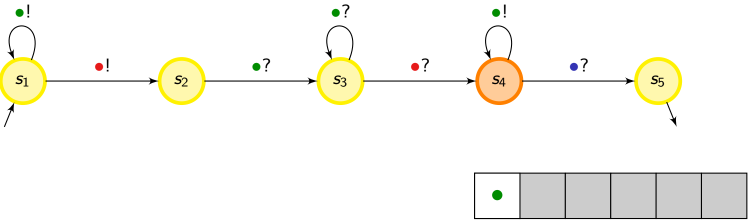
A channel machine = a finite automaton + a FIFO channel




$s_5$ is not reachable

# A short visit to channel machines (1)

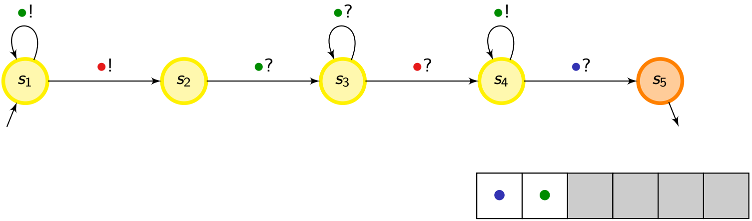
A channel machine $=$ a finite automaton $+$ a FIFO channel




- insertion errors: any letter can appear on the channel at any time

# A short visit to channel machines (1)

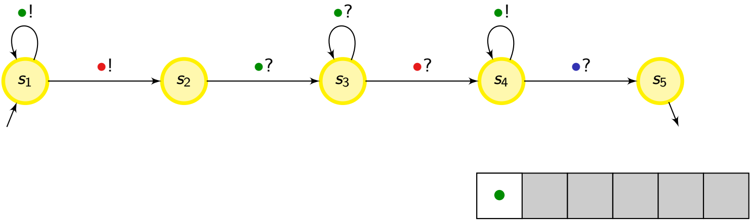
A channel machine $=$ a finite automaton $+$ a FIFO channel




- insertion errors: any letter can appear on the channel at any time

# A short visit to channel machines (1)

A channel machine $=$ a finite automaton $+$ a FIFO channel




- insertion errors: any letter can appear on the channel at any time

$s_5$ is reachable

# A short visit to channel machines (2)

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

# A short visit to channel machines (2)

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

**Proposition**

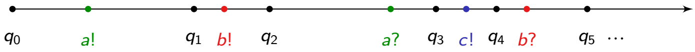
- The halting problem is undecidable for channel machines [BZ83].
- The halting problem is NPR for channel machines with insertion errors [Sch02].

[BZ83] Brand, Zafiropulo. On communicating finite-state machines (Journal of the ACM, 1983).
[Sch02] Schnoebelen. Verifying lossy channel systems has non-primitive recursive complexity (IPL, 2002).

# Channel machines and timed words

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

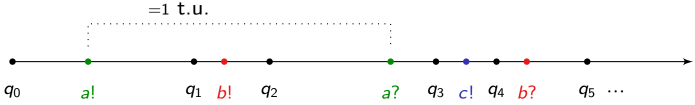
$$(q_0, \varepsilon) \xrightarrow{a!} (q_1, a) \xrightarrow{b!} (q_2, ab) \xrightarrow{a?} (q_3, b) \xrightarrow{c!} (q_4, bc) \xrightarrow{b?} (q_5, c) \cdots$$



# Channel machines and timed words

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

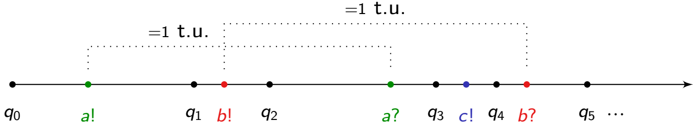
$$(q_0, \varepsilon) \xrightarrow{a!} (q_1, a) \xrightarrow{b!} (q_2, ab) \xrightarrow{a?} (q_3, b) \xrightarrow{c!} (q_4, bc) \xrightarrow{b?} (q_5, c) \cdots$$



# Channel machines and timed words

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

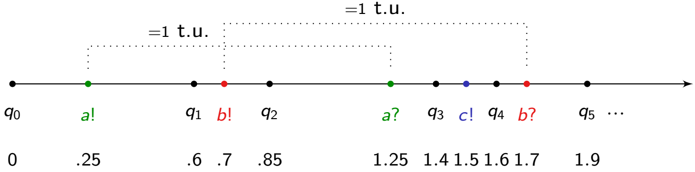
$$(q_0, \varepsilon) \xrightarrow{a!} (q_1, a) \xrightarrow{b!} (q_2, ab) \xrightarrow{a?} (q_3, b) \xrightarrow{c!} (q_4, bc) \xrightarrow{b?} (q_5, c) \cdots$$



# 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

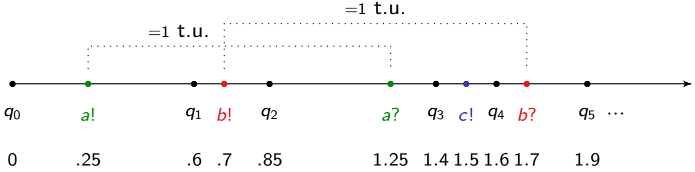
the channel machine* halts iff the formula $\varphi$ is satisfiable

* 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 $\mathcal{A}_{\text{univ}} \not\models \neg\varphi$

* possibly with insertion errors

# Constraints satisfied by the timed word

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

# 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 $a$,

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

# 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 $a$,

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

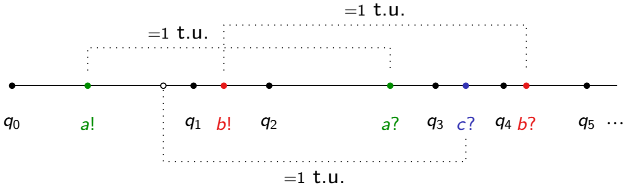
This formula is not sufficient!

# 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!

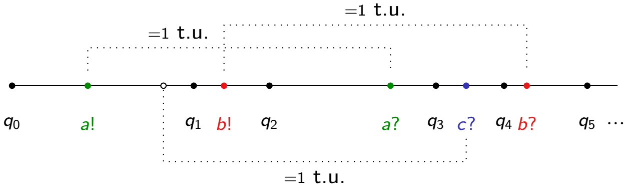


# Constraints satisfied by the timed word

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

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

This formula is not sufficient!



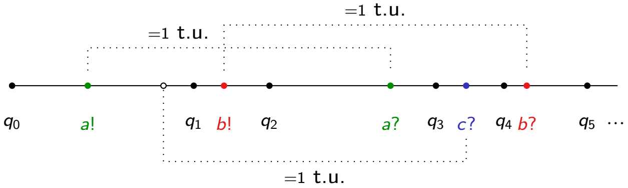

☞ only encodes a channel machine *with* insertion errors!

# Constraints satisfied by the timed word

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

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

This formula is not sufficient!




☞ only encodes a channel machine *with* insertion errors!

☞ 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?

$$\mathbf{G}\left((\mathbf{F}_{=1}\, a?) \longrightarrow a!\right)$$

We need to express the property:

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

- Why not reverse the previous implication?

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

  - 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?

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

  - correct in the continuous semantics
  - not correct in the pointwise semantics

We need to express the property:

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

- Why not reverse the previous implication?

$$\mathbf{G}\left((\mathbf{F}_{=1}\, a?) \longrightarrow a!\right)$$

  - correct in the continuous semantics
  - not correct in the pointwise semantics
- Why not look back in the past?

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

We need to express the property:

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

- Why not reverse the previous implication?

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

  - correct in the continuous semantics
  - not correct in the pointwise semantics
- Why not look back in the past?

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

  - correct for MTL+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?

$$\mathbf{G}\left((\mathbf{F}_{=1}\, a?) \longrightarrow a!\right)$$

  - correct in the continuous semantics
  - not correct in the pointwise semantics
- Why not look back in the past?

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

  - correct for MTL+Past (in the continuous and in the pointwise sem.)
  - no direct translation into MTL

We need to express the property:

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

- Why not reverse the previous implication?

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

  - correct in the continuous semantics
  - not correct in the pointwise semantics

- Why not look back in the past?

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

  - correct for MTL+Past (in the continuous and in the pointwise sem.)
  - no direct translation into MTL

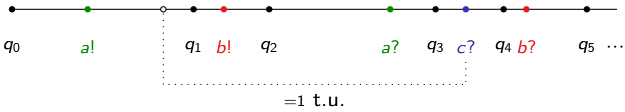
- A more tricky way:

$$\neg\Big(\mathbf{F}\, x.\mathbf{X}\, y.\mathbf{F}\left(x > 1 \wedge y < 1 \wedge c?\right)\Big)$$

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?
$$\mathbf{G}\left((\mathbf{F}_{=1}\, a?) \longrightarrow a!\right)$$
  - correct in the continuous semantics
  - not correct in the pointwise semantics
- Why not look back in the past?
$$\mathbf{G}\left(a? \longrightarrow \mathbf{F}^{-1}_{=1}\, a!\right)$$
  - correct for MTL+Past (in the continuous and in the pointwise sem.)
  - no direct translation into MTL
- A more tricky way:
$$\neg\Big(\mathbf{F}\, x.\mathbf{X}\, y.\mathbf{F}\,(x > 1 \wedge y < 1 \wedge c?)\Big)$$



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?

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

  - correct in the continuous semantics
  - not correct in the pointwise semantics

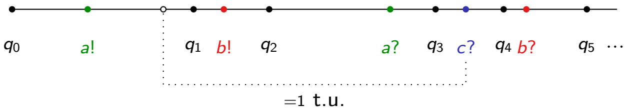
- Why not look back in the past?

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

  - correct for MTL+Past (in the continuous and in the pointwise sem.)
  - no direct translation into MTL

- A more tricky way:

$$\neg\Big(\mathbf{F}\, x.\mathbf{X}\, y.\mathbf{F}\,(x > 1 \wedge y < 1 \wedge c?)\Big)$$




  - 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:

| | pointwise sem. | continuous sem. |
|---|---|---|
| MTL | NPR [OW07] | undecidable [AFH96] |
| MTL+Past | undecidable | undecidable |
| TPTL | undecidable [AH94] | undecidable [AH94] |

# What remains to be proved

**Theorem**

Over finite runs, the model-checking problem is:

| | pointwise sem. | continuous sem. |
|---|---|---|
| MTL | decidable, NPR [OW07] | undecidable [AFH96] |
| MTL+Past | undecidable | undecidable |
| TPTL | undecidable [AH94] | undecidable [AH94] |

# From LTL to alternating automata

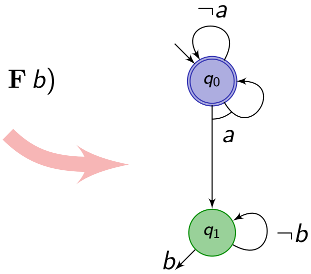
LTL formulas can be turned into linear alternating (Büchi) automata

$\mathbf{G}\,(a \rightarrow \mathbf{F}\,b)$

# From LTL to alternating automata

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

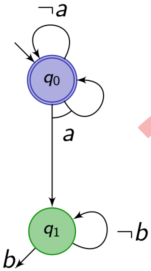
$\mathbf{G}\,(a \to \mathbf{F}\,b)$

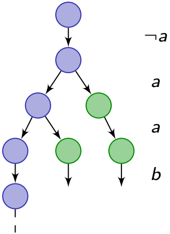


# From LTL to alternating automata

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

$\mathbf{G}\,(a \rightarrow \mathbf{F}\,b)$

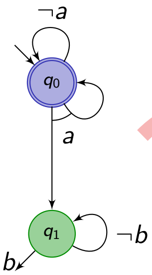


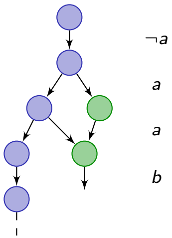



# From LTL to alternating automata

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

$\mathbf{G}\,(a \rightarrow \mathbf{F}\,b)$



# From MTL to alternating timed automata

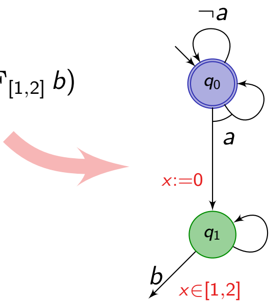
MTL formulas can be turned into linear alternating timed automata

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

# From MTL to alternating timed automata

MTL formulas can be turned into linear alternating timed automata

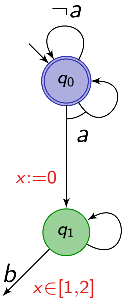
$\mathbf{G}\,(a \rightarrow \mathbf{F}_{[1,2]}\, b)$

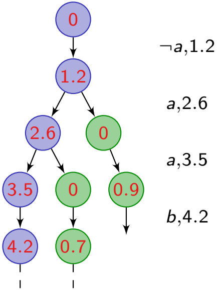


# From MTL to alternating timed automata

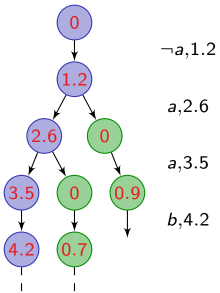
MTL formulas can be turned into linear alternating timed automata

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

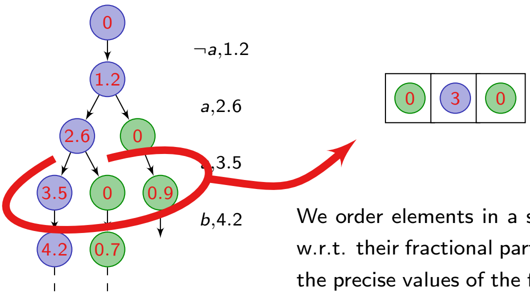


# An abstract transition system

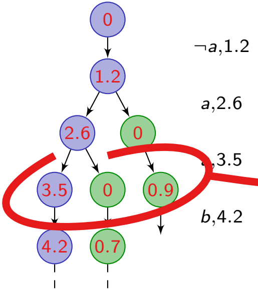


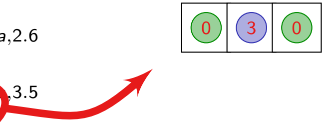
# An abstract transition system




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.

# An abstract transition system

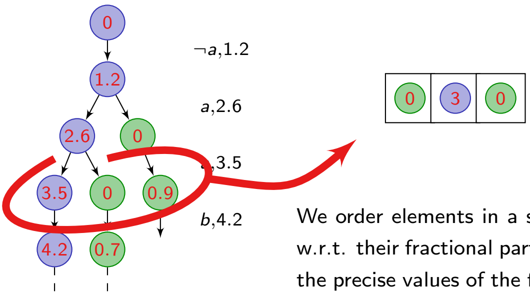



We order elements in a slice of the tree w.r.t. their fractional part, and we forget the precise values of the fractional parts.

☞ this defines an abstract (infinite) transition system

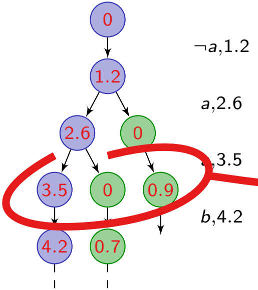
# An abstract transition system

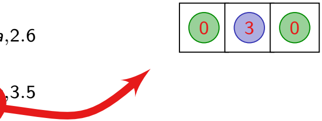



We order elements in a slice of the tree w.r.t. their fractional part, and we forget the precise values of the fractional parts.

- ☞ 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:

| | pointwise sem. | continuous sem. |
|---|---|---|
| MTL | decidable, NPR [OW05] | undecidable [AFH96] |
| MTL+Past | undecidable | undecidable |
| TPTL | undecidable [AH94] | undecidable [AH94] |

# 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:

| | pointwise sem. | continuous sem. |
|---|---|---|
| MTL | undecidable [OW06]$^\star$ | undecidable [AFH96] |
| MTL+Past | undecidable | undecidable |
| TPTL | undecidable [AH94] | undecidable [AH94] |

$^\star$ by reduction of the recurrence problem for channel machines

[MH84] Miyano, Hayashi. Alternating finite automata on $\omega$-words (TCS, 1984).
[OW06] Ouaknine, Worrell. On metric temporal logic and faulty Turing machines (FoSSaCS'06).

# Outline

# 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.

  **Old claim:** "Any logic strong enough to express the property $\mathbf{G}\,(\bullet \rightarrow \mathbf{F}_{=1}\,\bullet)$ 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 $\mathbf{G}\,(\bullet \to \mathbf{F}_{=1}\,\bullet)$ is undecidable"
- What if we forbid punctual constraints in MTL?

# The fragment without punctuality

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

  **Old claim:** "Any logic strong enough to express the property $\mathbf{G}\,(\bullet \rightarrow \mathbf{F}_{=1}\,\bullet)$ is undecidable"
- What if we forbid punctual constraints in MTL?

Metric Interval Temporal Logic (MITL): [AFH96]

$$\mathsf{MITL} \ni \varphi \ ::= \ a \ \mid \ \neg\varphi \ \mid \ \varphi \vee \varphi \ \mid \ \varphi \wedge \varphi \ \mid \ \varphi \, \mathbf{U}_I \, \varphi$$

with $I$ a non-punctual interval

[AFH96] Alur, Feder, Henzinger. The benefits of relaxing punctuality (Journal of the ACM, 1996).

# The fragment without punctuality

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

  **Old claim:** "Any logic strong enough to express the property $\mathbf{G}\,(\bullet \to \mathbf{F}_{=1}\,\bullet)$ is undecidable"
- What if we forbid punctual constraints in MTL?

Metric Interval Temporal Logic (MITL): [AFH96]

$$\mathsf{MITL} \ni \varphi \;::=\; a \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi\,\mathbf{U}_I\,\varphi$$

with $I$ a non-punctual interval

- Examples:
  - $\mathbf{G}\,(\bullet \to \mathbf{F}_{=1}\,\bullet)$ is not in MITL

[AFH96] Alur, Feder, Henzinger. The benefits of relaxing punctuality (Journal of the ACM, 1996).

# The fragment without punctuality

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

  **Old claim:** "Any logic strong enough to express the property $\mathbf{G}\,(\bullet \rightarrow \mathbf{F}_{=1}\,\bullet)$ is undecidable"
- What if we forbid punctual constraints in MTL?

Metric Interval Temporal Logic (MITL): [AFH96]

$$\mathsf{MITL} \ni \varphi \ ::= \ a \ \mid \ \neg\varphi \ \mid \ \varphi \vee \varphi \ \mid \ \varphi \wedge \varphi \ \mid \ \varphi\,\mathbf{U}_{I}\,\varphi$$

with $I$ a non-punctual interval

- Examples:
  - $\mathbf{G}\,(\bullet \rightarrow \mathbf{F}_{=1}\,\bullet)$ is not in MITL
  - $\mathbf{G}\,(\bullet \rightarrow \mathbf{F}_{[1,2]}\,\bullet)$ is in MITL

[AFH96] Alur, Feder, Henzinger. The benefits of relaxing punctuality (Journal of the ACM, 1996).

# Model-checking MITL is “easy”

**Theorem**

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

# Model-checking MITL is “easy”

**Theorem**

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

☞ we can bound the variability of the signals

# Model-checking MITL is "easy"

**Theorem**

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

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

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

# Model-checking MITL is "easy"

**Theorem**

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

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

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

- each time an • occurs within the first time unit, start a new clock, and check that a • occurs between 1 and 2 time units afterwards

# Model-checking MITL is “easy”

**Theorem**

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

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

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

- each time an • occurs within the first time unit, start a new clock, and check that a • 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 EXPSPACE-complete [AFH96].
If constants are encoded in unary, it is even PSPACE-complete [HR04].

☞ we can bound the variability of the signals

☞ an MITL formula defines a timed regular language

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

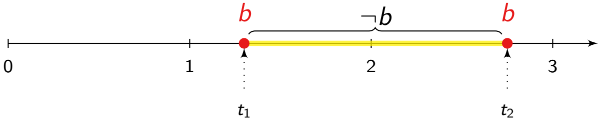
- each time an • occurs within the first time unit, start a new clock, and check that a • occurs between 1 and 2 time units afterwards
- this requires an unbounded number of clocks

☞ something more clever needs to be done

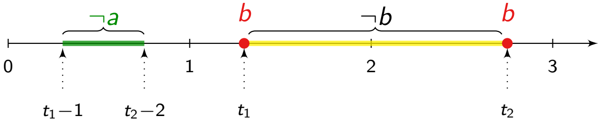
[HR04] Hirshfeld, Rabinovich. Logics for real time: decidability and complexity (Fundamenta Informaticae, 2004).

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

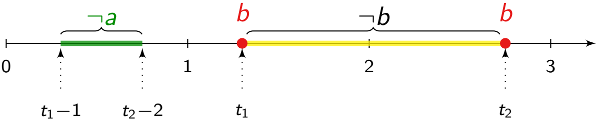
$$\varphi \;=\; \mathbf{G}_{(0,1)}\,(a \rightarrow \mathbf{F}_{[1,2]}\, b)$$

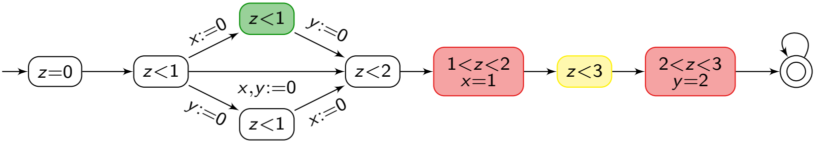


$$\varphi \;=\; \mathbf{G}_{(0,1)}\,(a \rightarrow \mathbf{F}_{[1,2]}\, b)$$

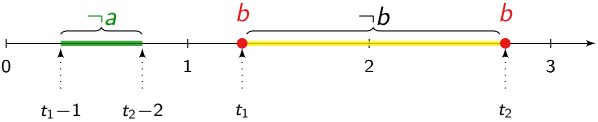


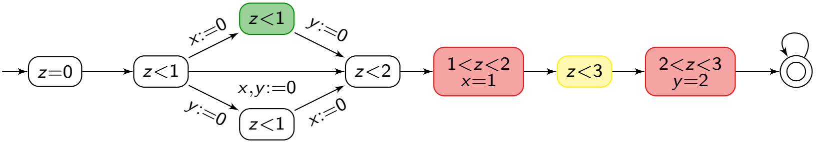
$$\varphi \;=\; \mathbf{G}_{(0,1)}\,(a \to \mathbf{F}_{[1,2]}\, b)$$



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




☞ 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?

# A co-flat fragment of MTL

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

We define coFlat-MTL: [BMOW07]

$$\text{coFlat-MTL} \ni \varphi \;::=\; a \mid \neg a \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \, \mathbf{U}_I \, \psi \mid \psi \, \widetilde{\mathbf{U}}_I \, \varphi$$

where $I$ unbounded $\Rightarrow \psi \in$ LTL

[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 coFlat-MTL: [BMOW07]

$$\text{coFlat-MTL} \ni \varphi \ ::= \ a \ \mid \ \neg a \ \mid \ \varphi \vee \varphi \ \mid \ \varphi \wedge \varphi \ \mid \ \varphi \, \mathbf{U}_I \, \psi \ \mid \ \psi \, \widetilde{\mathbf{U}}_I \, \varphi$$

where $I$ unbounded $\Rightarrow \psi \in$ LTL

- Examples:
  - $\mathbf{G}\,(\bullet \rightarrow \mathbf{F}_{=1}\,\bullet)$ is in coFlat-MTL

[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 coFlat-MTL: [BMOW07]

$$\text{coFlat-MTL} \ni \varphi \ ::= \ a \mid \neg a \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \,\mathbf{U}_I\, \psi \mid \psi \,\widetilde{\mathbf{U}}_I\, \varphi$$

where $I$ unbounded $\Rightarrow \psi \in$ LTL

- Examples:
  - $\mathbf{G}\,(\bullet \rightarrow \mathbf{F}_{=1}\,\bullet)$ is in coFlat-MTL
  - $\mathbf{F}\,\mathbf{G}_{\leqslant 1}\,\bullet$ is not in coFlat-MTL

[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 coFlat-MTL: [BMOW07]

$$\text{coFlat-MTL} \ni \varphi \ ::= \ a \mid \neg a \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \, \mathbf{U}_I \, \psi \mid \psi \, \widetilde{\mathbf{U}}_I \, \varphi$$

where $I$ unbounded $\Rightarrow \psi \in$ LTL

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

[BMOW07] Bouyer, Markey, Ouaknine, Worrell. The cost of punctuality (LICS'07).

# Model-checking coFlat-MTL is "easy"

**Theorem**

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

# Model-checking coFlat-MTL is "easy"

**Theorem**

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

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

$$\varphi_n \equiv {\color{red}\bullet} \wedge \mathbf{G}_{[0,2^n]}\, \varphi_D \qquad \text{with} \qquad \varphi_D = \begin{array}{rl} & ({\color{red}\bullet} \rightarrow \mathbf{F}_{=1}\,({\color{red}\bullet} \wedge \mathbf{F}_{\leqslant 1}\,{\color{green}\bullet})) \\ \wedge & ({\color{green}\bullet} \rightarrow \mathbf{F}_{=1}\,({\color{red}\bullet} \wedge \mathbf{F}_{\leqslant 1}\,{\color{green}\bullet})) \end{array}$$

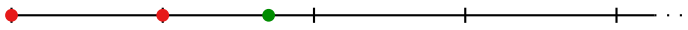
# Model-checking coFlat-MTL is "easy"

**Theorem**

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

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

$$\varphi_n \equiv {\color{red}\bullet} \wedge \mathbf{G}_{[0,2^n]}\, \varphi_D \qquad \text{with} \qquad \varphi_D = \begin{array}{rl} & ({\color{red}\bullet} \rightarrow \mathbf{F}_{=1}\,({\color{red}\bullet} \wedge \mathbf{F}_{\leqslant 1}\, {\color{green}\bullet})) \\ \wedge & ({\color{green}\bullet} \rightarrow \mathbf{F}_{=1}\,({\color{red}\bullet} \wedge \mathbf{F}_{\leqslant 1}\, {\color{green}\bullet})) \end{array}$$



# Model-checking coFlat-MTL is "easy"

**Theorem**

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

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

$$\varphi_n \equiv {\color{red}\bullet} \wedge \mathbf{G}_{[0,2^n]}\, \varphi_D \qquad \text{with} \qquad \varphi_D = \begin{array}{ll} & ({\color{red}\bullet} \rightarrow \mathbf{F}_{=1}\,({\color{red}\bullet} \wedge \mathbf{F}_{\leqslant 1}\, {\color{green}\bullet})) \\ \wedge & ({\color{green}\bullet} \rightarrow \mathbf{F}_{=1}\,({\color{red}\bullet} \wedge \mathbf{F}_{\leqslant 1}\, {\color{green}\bullet})) \end{array}$$

# Model-checking coFlat-MTL is "easy"

**Theorem**

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

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

$$\varphi_n \equiv \textcolor{red}{\bullet} \wedge \mathbf{G}_{[0,2^n]}\, \varphi_D \qquad \text{with} \qquad \varphi_D = \begin{array}{rl} & (\textcolor{red}{\bullet} \rightarrow \mathbf{F}_{=1}\,(\textcolor{red}{\bullet} \wedge \mathbf{F}_{\leqslant 1}\, \textcolor{green}{\bullet})) \\ \wedge & (\textcolor{green}{\bullet} \rightarrow \mathbf{F}_{=1}\,(\textcolor{red}{\bullet} \wedge \mathbf{F}_{\leqslant 1}\, \textcolor{green}{\bullet})) \end{array}$$

# Model-checking coFlat-MTL is “easy”

**Theorem**

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

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

$$\varphi_n \equiv \color{red}{\bullet} \color{black}{\wedge \mathbf{G}_{[0,2^n]}\, \varphi_D} \qquad \text{with} \qquad \varphi_D = \begin{array}{ll} & (\color{red}{\bullet}\color{black}{} \rightarrow \mathbf{F}_{=1}\,(\color{red}{\bullet}\color{black}{} \wedge \mathbf{F}_{\leqslant 1}\,\color{green}{\bullet}\color{black}{})) \\ \wedge & (\color{green}{\bullet}\color{black}{} \rightarrow \mathbf{F}_{=1}\,(\color{red}{\bullet}\color{black}{} \wedge \mathbf{F}_{\leqslant 1}\,\color{green}{\bullet}\color{black}{})) \end{array}$$

# Model-checking coFlat-MTL is “easy”

**Theorem**

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

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

$$\varphi_n \equiv \textcolor{red}{\bullet} \wedge \mathbf{G}_{[0,2^n]}\, \varphi_D \qquad \text{with} \qquad \varphi_D = \begin{array}{rl} & (\textcolor{red}{\bullet} \rightarrow \mathbf{F}_{=1}\,(\textcolor{red}{\bullet} \wedge \mathbf{F}_{\leqslant 1}\,\textcolor{green}{\bullet})) \\ \wedge & (\textcolor{green}{\bullet} \rightarrow \mathbf{F}_{=1}\,(\textcolor{red}{\bullet} \wedge \mathbf{F}_{\leqslant 1}\,\textcolor{green}{\bullet})) \end{array}$$

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

$$\mathbf{G}_{\leqslant 1}\,(\textcolor{red}{\bullet} \rightarrow \mathbf{F}_{=1}\,\textcolor{green}{\bullet}) \wedge \mathbf{G}_{\leqslant 1}\,\textcolor{red}{\bullet} \wedge \mathbf{G}_{(1,2]}\,\textcolor{green}{\bullet}$$

defines the context-free language $\{\textcolor{red}{\bullet}^n \textcolor{green}{\bullet}^m \mid n \leqslant m\}$.

# Algorithm for Bounded-MTL

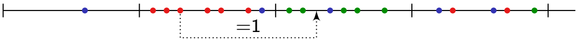
Assume one wants to verify formula

$$\mathbf{G}_{<2}\left(\textcolor{red}{\bullet} \rightarrow \mathbf{F}_{=1}\,\textcolor{green}{\bullet}\right)$$

# Algorithm for Bounded-MTL

Assume one wants to verify formula

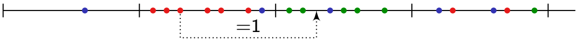
$$\mathbf{G}_{<2}\left({\color{red}\bullet} \rightarrow \mathbf{F}_{=1}\,{\color{green}\bullet}\right)$$



# Algorithm for Bounded-MTL

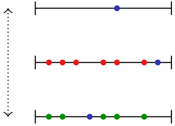
Assume one wants to verify formula

$$\mathbf{G}_{<2}\Big(\textcolor{red}{\bullet} \rightarrow \mathbf{F}_{=1}\,\textcolor{green}{\bullet}\Big)$$




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

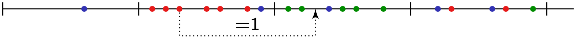
3 t.u. = useful duration



# Algorithm for Bounded-MTL

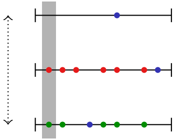
Assume one wants to verify formula

$$\mathbf{G}_{<2}\left(\bullet \rightarrow \mathbf{F}_{=1}\, \bullet\right)$$




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

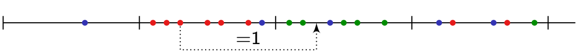
3 t.u. = useful duration



# Algorithm for Bounded-MTL

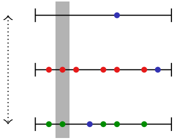
Assume one wants to verify formula

$$\mathbf{G}_{<2}\Big(\bullet \to \mathbf{F}_{=1}\,\bullet\Big)$$




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

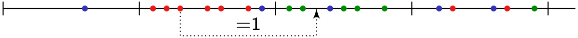
3 t.u. = useful duration



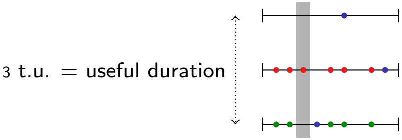
# Algorithm for Bounded-MTL

Assume one wants to verify formula

$$\mathbf{G}_{<2}\left({\color{red}\bullet} \rightarrow \mathbf{F}_{=1}\, {\color{green}\bullet}\right)$$



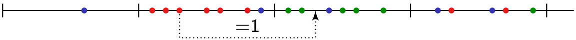

Offline, we stack all 'relevant' time units and use a sliding window:



# Algorithm for Bounded-MTL

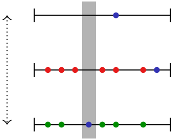
Assume one wants to verify formula

$$\mathbf{G}_{<2}\Big(\bullet \longrightarrow \mathbf{F}_{=1}\,\bullet\Big)$$




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

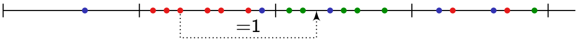
3 t.u. = useful duration



# Algorithm for Bounded-MTL

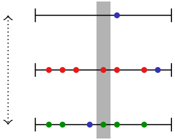
Assume one wants to verify formula

$$\mathbf{G}_{<2}\Big(\bullet \rightarrow \mathbf{F}_{=1} \bullet\Big)$$




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

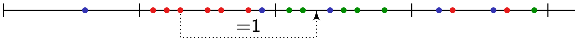
3 t.u. = useful duration



# Algorithm for Bounded-MTL

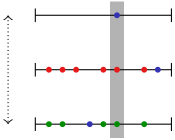
Assume one wants to verify formula

$$\mathbf{G}_{<2}\Big( {\color{red}\bullet} \to \mathbf{F}_{=1}\, {\color{green}\bullet} \Big)$$




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

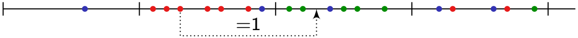
3 t.u. = useful duration



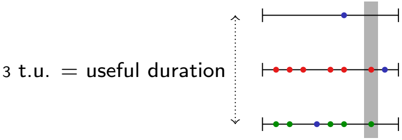
# Algorithm for Bounded-MTL

Assume one wants to verify formula

$$\mathbf{G}_{<2}\left(\bullet \to \mathbf{F}_{=1}\,\bullet\right)$$



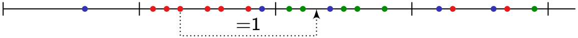

Offline, we stack all ‘relevant’ time units and use a sliding window:



# Algorithm for Bounded-MTL

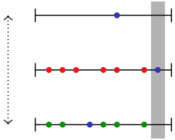
Assume one wants to verify formula

$$\mathbf{G}_{<2}\Big(\bullet \rightarrow \mathbf{F}_{=1}\,\bullet\Big)$$




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

3 t.u. = useful duration

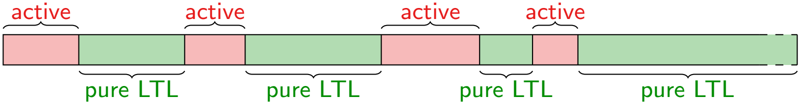


# Algorithm for coFlat-MTL

$\varphi \rightsquigarrow$ alternating timed automata $\mathcal{B}_{\neg\varphi}$ for $\neg\varphi$ with a 'flatness' property

# Algorithm for coFlat-MTL

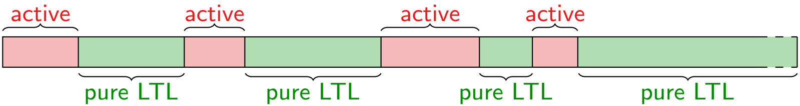
$\varphi \rightsquigarrow$ alternating timed automata $\mathcal{B}_{\neg\varphi}$ for $\neg\varphi$ with a 'flatness' property




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

# Algorithm for coFlat-MTL

$\varphi \rightsquigarrow$ alternating timed automata $\mathcal{B}_{\neg\varphi}$ for $\neg\varphi$ with a 'flatness' property




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

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

# Outline

# 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
- A recent result: coFlat-$\mathsf{MTL}_{\mathsf{MITL}}$ unifies coFlat-MTL and MITL, and is 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-$\mathsf{MTL}_{\mathsf{MITL}}$ unifies coFlat-MTL and MITL, and is EXPSPACE-complete [BMOW08]!
- No real data structures do exist for these logics.

# Conclusion

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

| | TCTL | MTL | Bounded-TCTL | Bounded-MTL |
|---|---|---|---|---|
| m.-c. | PSPACE | NPR/undec. | PSPACE | PSPACE |
| sat. | undec. | NPR/undec. | non-elem.⋆ | PSPACE |

⋆ ongoing work with Jenkins, Ouaknine, Worrell.

[BMOW08] Bouyer, Markey, Ouaknine, Worrell. A small-model theorem for real-time logics.