Monitoring Real-Time Systems Under Parametric Delay

Martin Zimmermann

Aalborg University

SynCoP 2025, Aarhus, Denmark

Joint work with

Martin Fränzle

Thomas M. Grosen

Sean Kauffman

Kim G. Larsen

Joint work with

Martin Fränzle

Thomas M. Grosen

Sean Kauffman

Kim G. Larsen

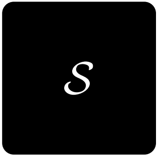
standing on the shoulders of giants:

Andreas Bauer

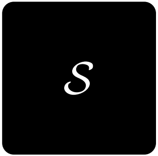
Martin Leucker

Christian Schallhart

Motivation

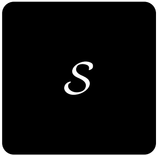


Motivation



$\{\texttt{init}\}$

Motivation



$\{\texttt{init}\}$ $\emptyset$

Motivation

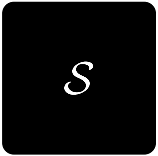
$\mathcal{S}$

$\{\texttt{init}\}$ $\emptyset$ $\{\texttt{req}\}$

Motivation

$\mathcal{S}$ $\{\texttt{init}\}$ $\emptyset$ $\{\texttt{req}\}$ $\emptyset$

Motivation



$\{\mathtt{init}\}$ $\emptyset$ $\{\mathtt{req}\}$ $\emptyset$ $\emptyset$

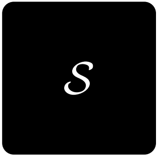
Motivation

$\mathcal{S}$ $\{\texttt{init}\}$ $\emptyset$ $\{\texttt{req}\}$ $\emptyset$ $\emptyset$ $\{\texttt{req}\}$

Motivation

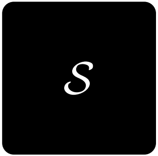
$\mathcal{S}$ $\{\texttt{init}\}$ $\emptyset$ $\{\texttt{req}\}$ $\emptyset$ $\emptyset$ $\{\texttt{req}\}$ $\emptyset$

Motivation



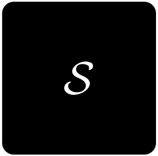
$\{\texttt{init}\}$ $\emptyset$ $\{\texttt{req}\}$ $\emptyset$ $\emptyset$ $\{\texttt{req}\}$ $\emptyset$ $\{\texttt{grnt}\}$

Motivation

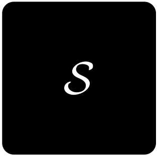


$\{\texttt{init}\}\ \emptyset\ \{\texttt{req}\}\ \emptyset\ \ \emptyset\ \{\texttt{req}\}\ \emptyset\ \{\texttt{grnt}\}\ \emptyset$

Motivation



No `grnt` before the first `req`.



$\{\texttt{init}\}$

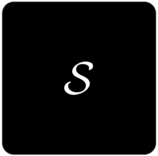
No $\texttt{grnt}$ before the first $\texttt{req}$.

Motivation

$\mathcal{S}$ $\{\texttt{init}\}$ $\emptyset$

No `grnt` before the first `req`.

Motivation



$\{\texttt{init}\}\ \emptyset\ \{\texttt{req}\}$

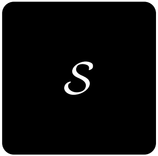
No grnt before the first req.

Motivation

$\mathcal{S}$ $\{\texttt{init}\}$ $\emptyset$ $\{\texttt{req}\}$

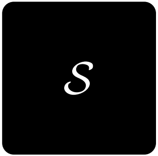
No `grnt` before the first `req`. ✔

Motivation



Every `req` is followed by a `grnt` within 3 steps.

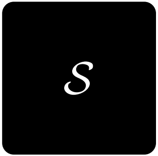
Motivation



{init}

Every req is followed by a grnt within 3 steps.

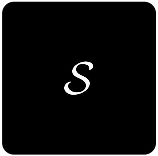
Motivation



$\{\texttt{init}\}$ $\emptyset$

Every `req` is followed by a `grnt` within 3 steps.

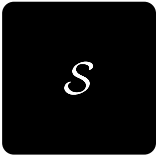
Motivation



$\{\texttt{init}\}$ $\emptyset$ $\{\texttt{req}\}$

Every `req` is followed by a `grnt` within 3 steps.

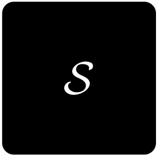
Motivation



$\{\texttt{init}\}$ $\emptyset$ $\{\texttt{req}\}$ $\emptyset$

Every `req` is followed by a `grnt` within 3 steps.

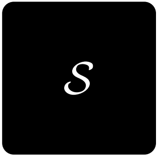
Motivation



$\{\texttt{init}\}$ $\emptyset$ $\{\texttt{req}\}$ $\emptyset$ $\emptyset$

Every `req` is followed by a `grnt` within 3 steps.

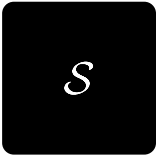
Motivation



{init} $\emptyset$ {req} $\emptyset$ $\emptyset$ {req}

Every req is followed by a grnt within 3 steps.

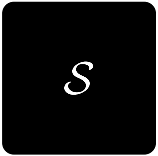
Motivation



$\{\texttt{init}\}$ $\emptyset$ $\{\texttt{req}\}$ $\emptyset$ $\emptyset$ $\{\texttt{req}\}$

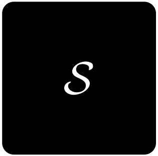
Every `req` is followed by a `grnt` within 3 steps. ✘

Motivation



Every `req` is eventually followed by a `grnt`.

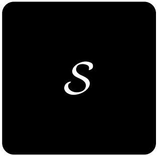
Motivation



$\{\texttt{init}\}$

Every `req` is eventually followed by a `grnt`.

Motivation



$\{\texttt{init}\}$ $\emptyset$

Every `req` is eventually followed by a `grnt`.

Motivation

$\mathcal{S}$ $\{\texttt{init}\}$ $\emptyset$ $\{\texttt{req}\}$

Every `req` is eventually followed by a `grnt`.

Motivation

$\mathcal{S}$

$\{\texttt{init}\}$ $\emptyset$ $\{\texttt{req}\}$ $\emptyset$

Every `req` is eventually followed by a `grnt`.

Motivation

$\mathcal{S}$ $\{\texttt{init}\}$ $\emptyset$ $\{\texttt{req}\}$ $\emptyset$ $\emptyset$

Every `req` is eventually followed by a `grnt`.

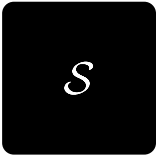
Motivation

$\mathcal{S}$

$\{\texttt{init}\}$ $\emptyset$ $\{\texttt{req}\}$ $\emptyset$ $\emptyset$ $\{\texttt{req}\}$

Every `req` is eventually followed by a `grnt`.

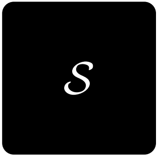
Motivation



$\{\texttt{init}\}$ $\emptyset$ $\{\texttt{req}\}$ $\emptyset$ $\emptyset$ $\{\texttt{req}\}$ $\emptyset$

Every `req` is eventually followed by a `grnt`.

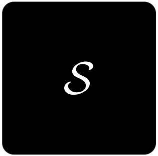
Motivation



$\{\texttt{init}\}$ $\emptyset$ $\{\texttt{req}\}$ $\emptyset$ $\emptyset$ $\{\texttt{req}\}$ $\emptyset$ $\{\texttt{grnt}\}$

Every `req` is eventually followed by a `grnt`.

Motivation



$\{\texttt{init}\}$ $\emptyset$ $\{\texttt{req}\}$ $\emptyset$ $\emptyset$ $\{\texttt{req}\}$ $\emptyset$ $\{\texttt{grnt}\}$ $\emptyset$

Every `req` is eventually followed by a `grnt`.

Monitoring in a Nutshell

1. Observe (finite) executions of a system..
2. and make verdicts about the satisfaction or violation of a property about infinite executions:
   - ✔ if the property is already satisfied.
   - ✘ if the property is already violated.
   - **?** otherwise.

Monitoring in a Nutshell

1. Observe (finite) executions of a system..
2. and make verdicts about the satisfaction or violation of a property about infinite executions:
   - ✔ if the property is already satisfied.
   - ✘ if the property is already violated.
   - **?** otherwise.

Advantages

- Lightweight
- Blackbox
- Online
- Violations detected as soon as possible (enables quick mitigation)

Outline

1. Monitoring in Discrete Time
2. Monitoring in Real-Time
3. Monitoring under Delay in Real-Time
4. Conclusion

LTL in One Slide

Syntax

$$\varphi ::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{X}\,\varphi \mid \varphi\,\mathsf{U}\,\varphi \qquad \text{where } a \in \mathrm{AP}$$

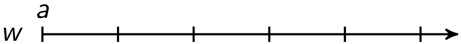
LTL in One Slide

Syntax

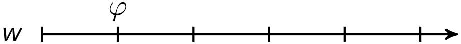
$$\varphi ::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{X}\,\varphi \mid \varphi \,\mathsf{U}\, \varphi \qquad \text{where } a \in \mathrm{AP}$$

Semantics

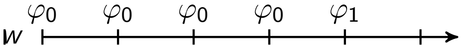
- $w \models a$:



- $w \models \mathsf{X}\,\varphi$:



- $w \models \varphi_0 \,\mathsf{U}\, \varphi_1$:



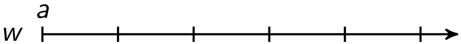
LTL in One Slide

Syntax

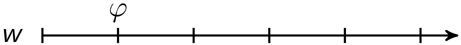
$$\varphi ::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{X}\,\varphi \mid \varphi\,\mathsf{U}\,\varphi \qquad \text{where } a \in \mathrm{AP}$$

Semantics

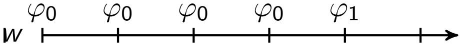
- $w \models a$:



- $w \models \mathsf{X}\,\varphi$:



- $w \models \varphi_0\,\mathsf{U}\,\varphi_1$:



Syntactic Sugar

- $\mathsf{F}\,\psi = \mathsf{tt}\,\mathsf{U}\,\psi$
- $\mathsf{G}\,\psi = \neg\,\mathsf{F}\,\neg\psi$

Examples

- No grnt before the first req.

$$((\neg \mathtt{grnt})\,\mathsf{U}\,\mathtt{req}) \vee \mathsf{G}\,\neg\mathtt{req}$$

Examples

- No $\texttt{grnt}$ before the first $\texttt{req}$.

$$((\neg \texttt{grnt}) \, \mathsf{U} \, \texttt{req}) \vee \mathsf{G} \, \neg \texttt{req}$$

- Every $\texttt{req}$ is followed by a $\texttt{grnt}$ within 3 steps.

$$\mathsf{G}(\texttt{req} \rightarrow \texttt{grnt} \vee \mathsf{X} \, \texttt{grnt} \vee \mathsf{X} \, \mathsf{X} \, \texttt{grnt} \vee \mathsf{X} \, \mathsf{X} \, \mathsf{X} \, \texttt{grnt})$$

Examples

- No $\texttt{grnt}$ before the first $\texttt{req}$.

$$((\neg\texttt{grnt})\,\mathsf{U}\,\texttt{req}) \vee \mathsf{G}\,\neg\texttt{req}$$

- Every $\texttt{req}$ is followed by a $\texttt{grnt}$ within 3 steps.

$$\mathsf{G}(\texttt{req} \rightarrow \texttt{grnt} \vee \mathsf{X}\,\texttt{grnt} \vee \mathsf{X}\,\mathsf{X}\,\texttt{grnt} \vee \mathsf{X}\,\mathsf{X}\,\mathsf{X}\,\texttt{grnt})$$

- Every $\texttt{req}$ is eventually followed by a $\texttt{grnt}$.

$$\mathsf{G}(\texttt{req} \rightarrow \mathsf{F}\,\texttt{grnt})$$

Monitoring in Discrete Time

Let $\varphi \subseteq \Sigma^\omega$ be a property. The monitoring function

$$\mathcal{M}_\varphi \colon \Sigma^* \to \{✔, ✘, \textbf{?}\}$$

is defined as

$$\mathcal{M}_\varphi(w) = \begin{cases} ✔ & \text{if } w \cdot \mu \in \varphi \text{ for all } \mu \in \Sigma^\omega, \\ ✘ & \text{if } w \cdot \mu \notin \varphi \text{ for all } \mu \in \Sigma^\omega, \\ \textbf{?} & \text{otherwise.} \end{cases}$$

Examples

- Let $\varphi_1 = (\neg \mathtt{grnt}) \, \mathsf{U} \, \mathtt{req}((\neg \mathtt{grnt}) \, \mathsf{U} \, \mathtt{req}) \vee \mathsf{G} \, \neg \mathtt{req}$
  - $\mathcal{M}_{\varphi_1}(\emptyset) =$ **?**

Examples

- Let $\varphi_1 = (\neg \mathtt{grnt})\,\mathsf{U}\,\mathtt{req}((\neg \mathtt{grnt})\,\mathsf{U}\,\mathtt{req}) \vee \mathsf{G}\,\neg \mathtt{req}$
  - $\mathcal{M}_{\varphi_1}(\emptyset) =$ **?**
  - $\mathcal{M}_{\varphi_1}(\emptyset\{\mathtt{req}\}) =$ ✔

Examples

- Let $\varphi_1 = (\neg \mathtt{grnt})\,\mathsf{U}\,\mathtt{req}((\neg \mathtt{grnt})\,\mathsf{U}\,\mathtt{req}) \vee \mathsf{G}\,\neg \mathtt{req}$
  - $\mathcal{M}_{\varphi_1}(\emptyset) =$ **?**
  - $\mathcal{M}_{\varphi_1}(\emptyset\{\mathtt{req}\}) =$ ✔
  - $\mathcal{M}_{\varphi_1}(\emptyset\{\mathtt{grnt}\}) =$ ✘

Examples

- Let $\varphi_1 = (\neg \mathtt{grnt}) \mathsf{U} \mathtt{req}((\neg \mathtt{grnt}) \mathsf{U} \mathtt{req}) \vee \mathsf{G} \neg \mathtt{req}$
  - $\mathcal{M}_{\varphi_1}(\emptyset) =$ **?**
  - $\mathcal{M}_{\varphi_1}(\emptyset\{\mathtt{req}\}) =$ ✔
  - $\mathcal{M}_{\varphi_1}(\emptyset\{\mathtt{grnt}\}) =$ ✘
- Let $\varphi_2 = \mathsf{G}(\mathtt{req} \rightarrow \mathtt{grnt} \vee \mathsf{X}\, \mathtt{grnt} \vee \mathsf{X}\,\mathsf{X}\, \mathtt{grnt} \vee \mathsf{X}\,\mathsf{X}\,\mathsf{X}\, \mathtt{grnt})$
  - $\mathcal{M}_{\varphi_2}(\{\mathtt{req}\}) =$ **?**

Examples

- Let $\varphi_1 = (\neg\texttt{grnt})\,\mathsf{U}\,\texttt{req}((\neg\texttt{grnt})\,\mathsf{U}\,\texttt{req}) \vee \mathsf{G}\,\neg\texttt{req}$
  - $\mathcal{M}_{\varphi_1}(\emptyset) =$ **?**
  - $\mathcal{M}_{\varphi_1}(\emptyset\{\texttt{req}\}) =$ ✔
  - $\mathcal{M}_{\varphi_1}(\emptyset\{\texttt{grnt}\}) =$ ✘
- Let $\varphi_2 = \mathsf{G}(\texttt{req} \rightarrow \texttt{grnt} \vee \mathsf{X}\,\texttt{grnt} \vee \mathsf{X}\,\mathsf{X}\,\texttt{grnt} \vee \mathsf{X}\,\mathsf{X}\,\mathsf{X}\,\texttt{grnt})$
  - $\mathcal{M}_{\varphi_2}(\{\texttt{req}\}) =$ **?**
  - $\mathcal{M}_{\varphi_2}(\{\texttt{req}\}\emptyset) =$ **?**

Examples

- Let $\varphi_1 = (\neg\texttt{grnt})\,\mathsf{U}\,\texttt{req}((\neg\texttt{grnt})\,\mathsf{U}\,\texttt{req}) \vee \mathsf{G}\,\neg\texttt{req}$
  - $\mathcal{M}_{\varphi_1}(\emptyset) = $ **?**
  - $\mathcal{M}_{\varphi_1}(\emptyset\{\texttt{req}\}) = $ ✔
  - $\mathcal{M}_{\varphi_1}(\emptyset\{\texttt{grnt}\}) = $ ✘
- Let $\varphi_2 = \mathsf{G}(\texttt{req} \rightarrow \texttt{grnt} \vee \mathsf{X}\,\texttt{grnt} \vee \mathsf{X}\,\mathsf{X}\,\texttt{grnt} \vee \mathsf{X}\,\mathsf{X}\,\mathsf{X}\,\texttt{grnt})$
  - $\mathcal{M}_{\varphi_2}(\{\texttt{req}\}) = $ **?**
  - $\mathcal{M}_{\varphi_2}(\{\texttt{req}\}\emptyset) = $ **?**
  - $\mathcal{M}_{\varphi_2}(\{\texttt{req}\}\emptyset\{\texttt{req}\}) = $ **?**

Examples

- Let $\varphi_1 = (\neg \mathtt{grnt}) \mathsf{U} \mathtt{req}((\neg \mathtt{grnt}) \mathsf{U} \mathtt{req}) \vee \mathsf{G} \neg \mathtt{req}$
  - $\mathcal{M}_{\varphi_1}(\emptyset) =$ **?**
  - $\mathcal{M}_{\varphi_1}(\emptyset\{\mathtt{req}\}) =$ ✔
  - $\mathcal{M}_{\varphi_1}(\emptyset\{\mathtt{grnt}\}) =$ ✘
- Let $\varphi_2 = \mathsf{G}(\mathtt{req} \rightarrow \mathtt{grnt} \vee \mathsf{X}\, \mathtt{grnt} \vee \mathsf{X}\,\mathsf{X}\, \mathtt{grnt} \vee \mathsf{X}\,\mathsf{X}\,\mathsf{X}\, \mathtt{grnt})$
  - $\mathcal{M}_{\varphi_2}(\{\mathtt{req}\}) =$ **?**
  - $\mathcal{M}_{\varphi_2}(\{\mathtt{req}\}\emptyset) =$ **?**
  - $\mathcal{M}_{\varphi_2}(\{\mathtt{req}\}\emptyset\{\mathtt{req}\}) =$ **?**
  - $\mathcal{M}_{\varphi_2}(\{\mathtt{req}\}\emptyset\{\mathtt{req}\}\emptyset) =$ ✘

Examples

- Let $\varphi_1 = (\neg \mathtt{grnt}) \,\mathsf{U}\, \mathtt{req}((\neg \mathtt{grnt}) \,\mathsf{U}\, \mathtt{req}) \vee \mathsf{G} \neg \mathtt{req}$
  - $\mathcal{M}_{\varphi_1}(\emptyset) = $ **?**
  - $\mathcal{M}_{\varphi_1}(\emptyset\{\mathtt{req}\}) = $ ✔
  - $\mathcal{M}_{\varphi_1}(\emptyset\{\mathtt{grnt}\}) = $ ✘
- Let $\varphi_2 = \mathsf{G}(\mathtt{req} \rightarrow \mathtt{grnt} \vee \mathsf{X}\,\mathtt{grnt} \vee \mathsf{X}\,\mathsf{X}\,\mathtt{grnt} \vee \mathsf{X}\,\mathsf{X}\,\mathsf{X}\,\mathtt{grnt})$
  - $\mathcal{M}_{\varphi_2}(\{\mathtt{req}\}) = $ **?**
  - $\mathcal{M}_{\varphi_2}(\{\mathtt{req}\}\emptyset) = $ **?**
  - $\mathcal{M}_{\varphi_2}(\{\mathtt{req}\}\emptyset\{\mathtt{req}\}) = $ **?**
  - $\mathcal{M}_{\varphi_2}(\{\mathtt{req}\}\emptyset\{\mathtt{req}\}\emptyset) = $ ✘
  - $\mathcal{M}_{\varphi_2}(\{\mathtt{req}\}\emptyset\{\mathtt{req}\}\{\mathtt{grnt}\}) = $ **?**

Examples

- Let $\varphi_1 = (\neg \mathtt{grnt})\,\mathsf{U}\,\mathtt{req}((\neg \mathtt{grnt})\,\mathsf{U}\,\mathtt{req}) \vee \mathsf{G}\,\neg \mathtt{req}$
  - $\mathcal{M}_{\varphi_1}(\emptyset) =$ **?**
  - $\mathcal{M}_{\varphi_1}(\emptyset\{\mathtt{req}\}) =$ ✔
  - $\mathcal{M}_{\varphi_1}(\emptyset\{\mathtt{grnt}\}) =$ ✘
- Let $\varphi_2 = \mathsf{G}(\mathtt{req} \rightarrow \mathtt{grnt} \vee \mathsf{X}\,\mathtt{grnt} \vee \mathsf{X}\,\mathsf{X}\,\mathtt{grnt} \vee \mathsf{X}\,\mathsf{X}\,\mathsf{X}\,\mathtt{grnt})$
  - $\mathcal{M}_{\varphi_2}(\{\mathtt{req}\}) =$ **?**
  - $\mathcal{M}_{\varphi_2}(\{\mathtt{req}\}\emptyset) =$ **?**
  - $\mathcal{M}_{\varphi_2}(\{\mathtt{req}\}\emptyset\{\mathtt{req}\}) =$ **?**
  - $\mathcal{M}_{\varphi_2}(\{\mathtt{req}\}\emptyset\{\mathtt{req}\}\emptyset) =$ ✘
  - $\mathcal{M}_{\varphi_2}(\{\mathtt{req}\}\emptyset\{\mathtt{req}\}\{\mathtt{grnt}\}) =$ **?**
- Let $\varphi_3 = \mathsf{G}(\mathtt{req} \rightarrow \mathsf{F}\,\mathtt{grnt})$
  - $\mathcal{M}_{\varphi_3}(w) =$ **?** for all w

Büchi Automata

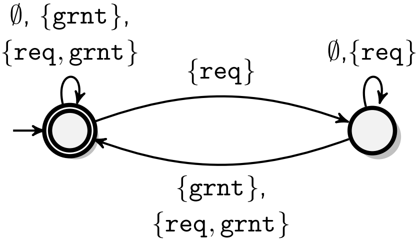
For algorithmic purposes, we compile LTL-defined properties into Büchi automata processing infinite words.

Büchi Automata

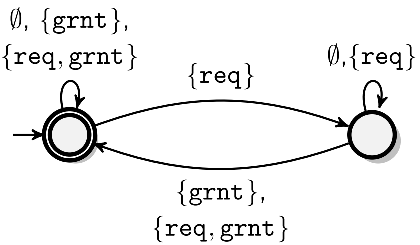
For algorithmic purposes, we compile LTL-defined properties into Büchi automata processing infinite words.

Example

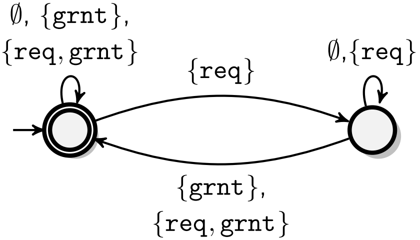
A Büchi automaton $\mathcal{B}_\varphi$ for $\varphi = \mathsf{G}(\mathtt{req} \rightarrow \mathsf{F}\,\mathtt{grnt})$:



Towards an Algorithm

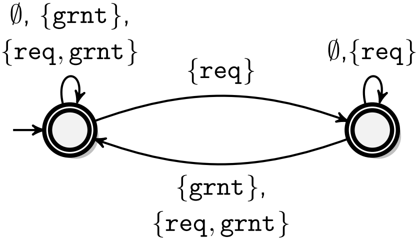


Towards an Algorithm



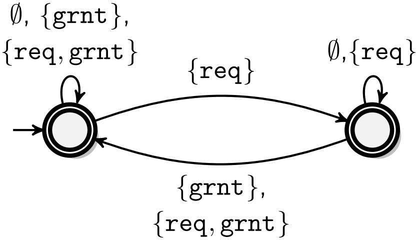
- A state q of $\mathcal{A}_\varphi$ has a nonempty language, if an accepting runs starts in q.

Towards an Algorithm



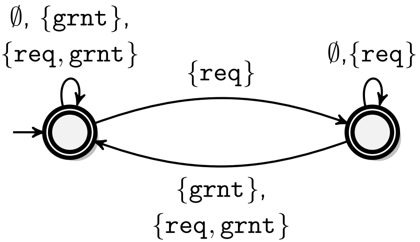
- A state q of $\mathcal{A}_\varphi$ has a nonempty language, if an accepting runs starts in q.
- Make nonempty language states of $\mathcal{A}_\varphi$ accepting and interpret the resulting automaton as an NFA $\mathcal{N}_\varphi$ on finite words.

Towards an Algorithm



- A state q of $\mathcal{A}_\varphi$ has a nonempty language, if an accepting runs starts in q.
- Make nonempty language states of $\mathcal{A}_\varphi$ accepting and interpret the resulting automaton as an NFA $\mathcal{N}_\varphi$ on finite words.
- Then: $w \in L(\mathcal{N}_\varphi)$ iff there exists $\mu \in \Sigma^\omega$ such that $w \cdot \mu \in \varphi$.

Towards an Algorithm



- A state q of $\mathcal{A}_\varphi$ has a nonempty language, if an accepting runs starts in q.
- Make nonempty language states of $\mathcal{A}_\varphi$ accepting and interpret the resulting automaton as an NFA $\mathcal{N}_\varphi$ on finite words.
- Then: $w \in L(\mathcal{N}_\varphi)$ iff there exists $\mu \in \Sigma^\omega$ such that $w \cdot \mu \in \varphi$.
- Equivalently: $w \notin L(\mathcal{N}_\varphi)$ iff $\mathcal{M}_\varphi(w) = $ ✘.

Complete Construction

φ

Complete Construction

$$\varphi \xrightarrow{\text{exp}} \mathcal{B}_{\varphi}$$

Complete Construction

$$\varphi \xrightarrow{\text{exp}} \mathcal{B}_\varphi \xrightarrow{\text{cons}} \mathcal{N}_\varphi$$

Complete Construction

$$\varphi \xrightarrow{\mathsf{exp}} \mathcal{B}_\varphi \xrightarrow{\mathsf{cons}} \mathcal{N}_\varphi \xrightarrow{\mathsf{exp}} \mathcal{D}_\varphi$$

Complete Construction

$$\varphi \xrightarrow{\text{exp}} \mathcal{B}_\varphi \xrightarrow{\text{cons}} \mathcal{N}_\varphi \xrightarrow{\text{exp}} \mathcal{D}_\varphi$$

$$\neg\varphi \xrightarrow[\text{exp}]{} \mathcal{B}_{\neg\varphi} \xrightarrow[\text{cons}]{} \mathcal{N}_{\neg\varphi} \xrightarrow[\text{exp}]{} \mathcal{D}_{\neg\varphi}$$

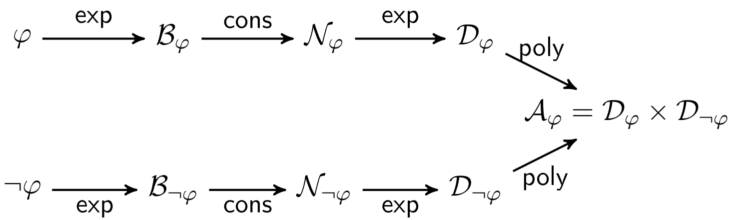
Complete Construction

$$\varphi \xrightarrow{\text{exp}} \mathcal{B}_\varphi \xrightarrow{\text{cons}} \mathcal{N}_\varphi \xrightarrow{\text{exp}} \mathcal{D}_\varphi$$

$$\neg\varphi \xrightarrow[\text{exp}]{} \mathcal{B}_{\neg\varphi} \xrightarrow[\text{cons}]{} \mathcal{N}_{\neg\varphi} \xrightarrow[\text{exp}]{} \mathcal{D}_{\neg\varphi}$$

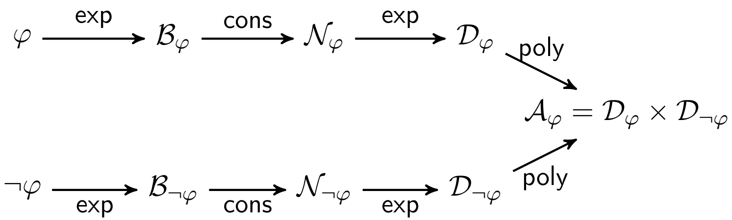
- $w \notin L(\mathcal{D}_\varphi)$ iff $\mathcal{M}_\varphi(w) =$ ✘.
- $w \notin L(\mathcal{D}_{\neg\varphi})$ iff $\mathcal{M}_\varphi(w) =$ ✔.
- $w \in L(\mathcal{D}_\varphi) \cap L(\mathcal{D}_{\neg\varphi})$ iff $\mathcal{M}_\varphi =$ **?**.

Complete Construction



- $w \notin L(\mathcal{D}_\varphi)$ iff $\mathcal{M}_\varphi(w) = $ ✘.
- $w \notin L(\mathcal{D}_{\neg\varphi})$ iff $\mathcal{M}_\varphi(w) = $ ✔.
- $w \in L(\mathcal{D}_\varphi) \cap L(\mathcal{D}_{\neg\varphi})$ iff $\mathcal{M}_\varphi = $ **?**.

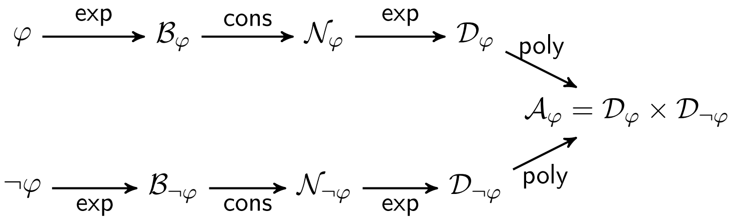
Complete Construction



Theorem (Bauer, Leucker, Schallhart '06)

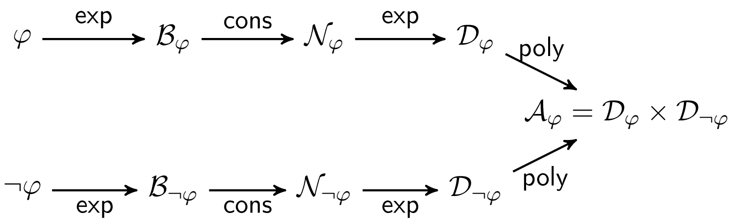
Given an LTL property φ, one can effectively construct a deterministic automaton with output implementing $\mathcal{M}_\varphi$, which has doubly-exponential size.

On-the-fly Monitoring



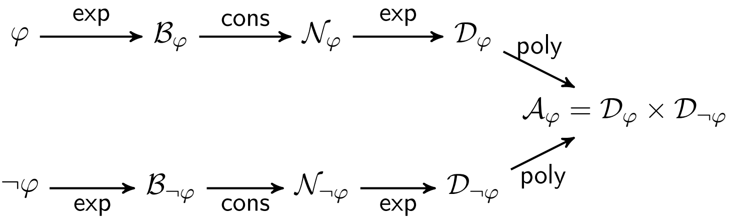
- Alternatively, given w, we can compute the sets $\mathcal{R}_{\mathcal{B}_\varphi}(w)$ and $\mathcal{R}_{\mathcal{B}_{\neg\varphi}}(w)$ of reachable states.

On-the-fly Monitoring



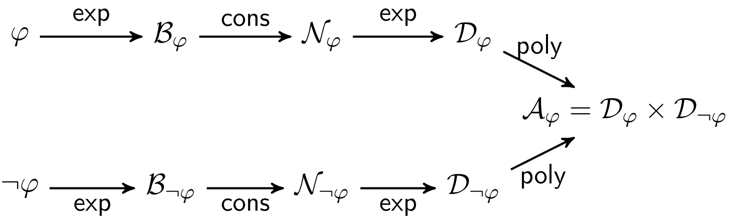
- Alternatively, given w, we can compute the sets $\mathcal{R}_{\mathcal{B}_\varphi}(w)$ and $\mathcal{R}_{\mathcal{B}_{\neg\varphi}}(w)$ of reachable states.
  - If $\mathcal{R}_{\mathcal{B}_{\neg\varphi}}(w)$ does not contain a state with nonempty language, then $\mathcal{M}_\varphi(w) = ✔$.

On-the-fly Monitoring



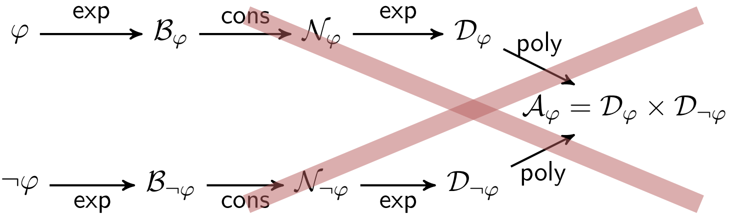
- Alternatively, given w, we can compute the sets $\mathcal{R}_{\mathcal{B}_\varphi}(w)$ and $\mathcal{R}_{\mathcal{B}_{\neg\varphi}}(w)$ of reachable states.
  - If $\mathcal{R}_{\mathcal{B}_{\neg\varphi}}(w)$ does not contain a state with nonempty language, then $\mathcal{M}_\varphi(w) = $ ✔.
  - If $\mathcal{R}_{\mathcal{B}_\varphi}(w)$ does not contain a state with nonempty language, then $\mathcal{M}_\varphi(w) = $ ✘.

On-the-fly Monitoring



- Alternatively, given w, we can compute the sets $\mathcal{R}_{\mathcal{B}_\varphi}(w)$ and $\mathcal{R}_{\mathcal{B}_{\neg\varphi}}(w)$ of reachable states.
  - If $\mathcal{R}_{\mathcal{B}_{\neg\varphi}}(w)$ does not contain a state with nonempty language, then $\mathcal{M}_\varphi(w) =$ ✔.
  - If $\mathcal{R}_{\mathcal{B}_\varphi}(w)$ does not contain a state with nonempty language, then $\mathcal{M}_\varphi(w) =$ ✘.
  - Otherwise, $\mathcal{M}_\varphi(w) =$ **?**.

On-the-fly Monitoring



- Alternatively, given w, we can compute the sets $\mathcal{R}_{\mathcal{B}_\varphi}(w)$ and $\mathcal{R}_{\mathcal{B}_{\neg\varphi}}(w)$ of reachable states.
  - If $\mathcal{R}_{\mathcal{B}_{\neg\varphi}}(w)$ does not contain a state with nonempty language, then $\mathcal{M}_\varphi(w) =$ ✔.
  - If $\mathcal{R}_{\mathcal{B}_\varphi}(w)$ does not contain a state with nonempty language, then $\mathcal{M}_\varphi(w) =$ ✘.
  - Otherwise, $\mathcal{M}_\varphi(w) =$ **?**.

Outline

1. Monitoring in Discrete Time

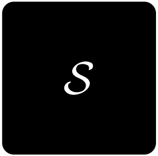
2. Monitoring in Real-Time

3. Monitoring under Delay in Real-Time

4. Conclusion

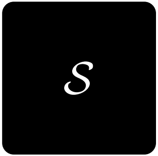
Motivation

Many systems and their specifications are real-time, not discrete-time!



Motivation

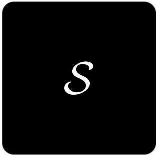
Many systems and their specifications are real-time, not discrete-time!



$(\mathtt{init}, 0)$

Motivation

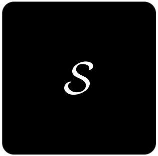
Many systems and their specifications are real-time, not discrete-time!



$(\texttt{init}, 0)$ $(\texttt{req}, 2)$

Motivation

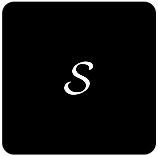
Many systems and their specifications are real-time, not discrete-time!



$(\texttt{init}, 0)$ $(\texttt{req}, 2)$ $(\texttt{req}, 3.7)$

Motivation

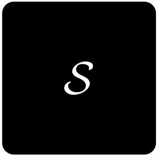
Many systems and their specifications are real-time, not discrete-time!



$(\mathtt{init}, 0) \quad (\mathtt{req}, 2) \quad (\mathtt{req}, 3.7) \quad (\mathtt{grnt}, 13)$

Motivation

Many systems and their specifications are real-time, not discrete-time!

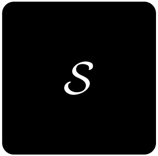


$(\texttt{init}, 0)$ $(\texttt{req}, 2)$ $(\texttt{req}, 3.7)$ $(\texttt{grnt}, 13)$

$(\texttt{req}, 13)$

Motivation

Many systems and their specifications are real-time, not discrete-time!



$(\texttt{init}, 0)$ $(\texttt{req}, 2)$ $(\texttt{req}, 3.7)$ $(\texttt{grnt}, 13)$

$(\texttt{req}, 13)$ $(\texttt{grnt}, 19.11)$

Motivation

Many systems and their specifications are real-time, not discrete-time!

$\mathcal{S}$

$(\mathtt{init}, 0)$ $(\mathtt{req}, 2)$ $(\mathtt{req}, 3.7)$ $(\mathtt{grnt}, 13)$

$(\mathtt{req}, 13)$ $(\mathtt{grnt}, 19.11)$ $(\mathtt{req}, 20)$

Motivation

Many systems and their specifications are real-time, not discrete-time!

$\mathcal{S}$

$(\texttt{init}, 0)$ $(\texttt{req}, 2)$ $(\texttt{req}, 3.7)$ $(\texttt{grnt}, 13)$

$(\texttt{req}, 13)$ $(\texttt{grnt}, 19.11)$ $(\texttt{req}, 20)$ $(\texttt{req}, 25)$

Motivation

Many systems and their specifications are real-time, not discrete-time!

$\mathcal{S}$

$(\texttt{init}, 0)$ $(\texttt{req}, 2)$ $(\texttt{req}, 3.7)$ $(\texttt{grnt}, 13)$ $(\texttt{req}, 13)$ $(\texttt{grnt}, 19.11)$ $(\texttt{req}, 20)$ $(\texttt{req}, 25)$

Bauer, Leucker, Schallhart '06:
Monitoring of TLTL properties using event-clock automata and regions.

Motivation

Many systems and their specifications are real-time, not discrete-time!

$\mathcal{S}$ $(\mathtt{init}, 0)$ $(\mathtt{req}, 2)$ $(\mathtt{req}, 3.7)$ $(\mathtt{grnt}, 13)$

$(\mathtt{req}, 13)$ $(\mathtt{grnt}, 19.11)$ $(\mathtt{req}, 20)$ $(\mathtt{req}, 25)$

Bauer, Leucker, Schallhart '06:
Monitoring of TLTL properties using event-clock automata and regions.

Grosen, Kauffman, Larsen, Z. '22:
Monitoring of MITL properties using timed Büchi automata and zones.

Timed Words

- A (point-based) timed word over an alphabet Σ is a sequence

$$(\sigma_0, t_0)(\sigma_1, t_1)(\sigma_2, t_2)\cdots$$

  such that

  - $\sigma_i \in \Sigma$ for all i,
  - $t_i \in \mathbb{R}_{\geq 0}$ for all i,
  - $t_0 \leq t_1 \leq t_2 \leq \cdots$, and
  - $t_0, t_1, t_2, \ldots$ diverges.

Timed Words

- A (point-based) timed word over an alphabet Σ is a sequence

$$(\sigma_0, t_0)(\sigma_1, t_1)(\sigma_2, t_2) \cdots$$

  such that
  - $\sigma_i \in \Sigma$ for all i,
  - $t_i \in \mathbb{R}_{\geq 0}$ for all i,
  - $t_0 \leq t_1 \leq t_2 \leq \cdots$, and
  - $t_0, t_1, t_2, \ldots$ diverges.

- The duration $\mathrm{dur}(w)$ of a finite timed word $w = (\sigma_0, t_0) \cdots (\sigma_n, t_n)$ is t_n.

Timed Words

- A (point-based) timed word over an alphabet Σ is a sequence

$$(\sigma_0, t_0)(\sigma_1, t_1)(\sigma_2, t_2)\cdots$$

  such that
  - $\sigma_i \in \Sigma$ for all i,
  - $t_i \in \mathbb{R}_{\geq 0}$ for all i,
  - $t_0 \leq t_1 \leq t_2 \leq \cdots$, and
  - $t_0, t_1, t_2, \ldots$ diverges.

- The duration $\mathrm{dur}(w)$ of a finite timed word $w = (\sigma_0, t_0)\cdots(\sigma_n, t_n)$ is t_n.

- Let $w = (\sigma_0, t_0)\cdots(\sigma_n, t_n)$, $w' = (\sigma_0, t_0)(\sigma_1, t_1)(\sigma_2, t_2)\cdots$, and $t \geq \mathrm{dur}(w)$. Then,

$$w \cdot_t w' = (\sigma_0, t_0)\cdots(\sigma_n, t_n)(\sigma_0, t_0 + t)(\sigma_1, t_1 + t)(\sigma_2, t_2 + t)\cdots$$

MITL in One Slide

Syntax

$$\varphi ::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{X}_I\, \varphi \mid \varphi\, \mathsf{U}_I\, \varphi$$

where

- $a \in \mathrm{AP}$ and
- $I \subseteq \mathbb{R}_{\geq 0}$ is a nonempty interval with endpoints in $\mathbb{N} \cup \{\infty\}$.

MITL in One Slide

Syntax

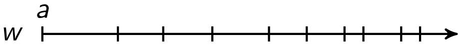
$$\varphi ::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{X}_I\, \varphi \mid \varphi\, \mathsf{U}_I\, \varphi$$

where

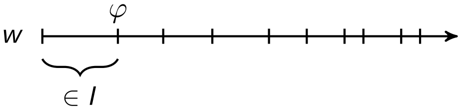
- $a \in \mathrm{AP}$ and
- $I \subseteq \mathbb{R}_{\geq 0}$ is a nonempty interval with endpoints in $\mathbb{N} \cup \{\infty\}$.

Semantics

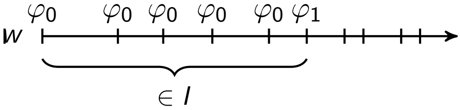
- $w \models a$:



- $w \models \mathsf{X}_I\, \varphi$:



- $w \models \varphi_0\, \mathsf{U}_I\, \varphi_1$:



Monitoring in Real Time

- Every $\mathtt{req}$ is followed by a $\mathtt{grnt}$ within 3 units of time.

$$\mathsf{G}_{[0,\infty)}(\mathtt{req} \rightarrow \mathsf{F}_{[0,3]}\,\mathtt{grnt})$$

Monitoring in Real Time

- Every `req` is followed by a `grnt` within 3 units of time.

$$G_{[0,\infty)}(\mathtt{req} \rightarrow F_{[0,3]}\,\mathtt{grnt})$$

- There is an a in the first 10 units of time and no b in the first 20 units of time.

$$F_{[0,10]}\, a \wedge G_{[0,20]}\, \neg b$$

Monitoring in Real-Time

Let $\varphi \subseteq T\Sigma^\omega$ be a property. The monitoring function

$$\mathcal{M}_\varphi \colon T\Sigma^* \to \{✔, ✘, ?\}$$

is defined as

$$\mathcal{M}_\varphi(w) = \begin{cases} ✔ & \text{if } w \cdot_{\mathrm{dur}(w)} \mu \in \varphi \text{ for all } \mu \in T\Sigma^\omega, \\ ✘ & \text{if } w \cdot_{\mathrm{dur}(w)} \mu \notin \varphi \text{ for all } \mu \in T\Sigma^\omega, \\ ? & \text{otherwise.} \end{cases}$$

We Can Do Better

Consider the property

$$\varphi = \mathsf{F}_{[0,10]}\, a \wedge \mathsf{G}_{[0,20]}\, \neg b.$$

- $\mathcal{M}_\varphi((a,4)) =$ **?**

We Can Do Better

Consider the property

$$\varphi = \mathsf{F}_{[0,10]}\, a \land \mathsf{G}_{[0,20]}\, \neg b.$$

- $\mathcal{M}_\varphi((a,4)) =$ **?**
- $\mathcal{M}_\varphi((a,4)(a,100)) =$ ✔

We Can Do Better

Consider the property

$$\varphi = \mathsf{F}_{[0,10]}\, a \wedge \mathsf{G}_{[0,20]}\, \neg b.$$

- $\mathcal{M}_\varphi((a,4)) =$ **?**
- $\mathcal{M}_\varphi((a,4)(a,100)) =$ ✔
- But we can give the verdict ✔ much earlier, i.e., when no further event has been observed for more than 16 units of time after the initial a at time 4.

Monitoring in Real-Time, Take 2

Let $\varphi \subseteq T\Sigma^\omega$ be a property. The monitoring function

$$\mathcal{M}_\varphi \colon T\Sigma^* \times \mathbb{R}_{\geq 0} \to \{✔, ✘, ?\}$$

is defined as

$$\mathcal{M}_\varphi(w, t) = \begin{cases} ✔ & \text{if } w \cdot_t \mu \in \varphi \text{ for all } \mu \in T\Sigma^\omega, \\ ✘ & \text{if } w \cdot_t \mu \notin \varphi \text{ for all } \mu \in T\Sigma^\omega, \\ ? & \text{otherwise,} \end{cases}$$

if $t \geq \mathrm{dur}(w)$, otherwise it is undefined.

Examples

Again, consider the property

$$\varphi = \mathsf{F}_{[0,10]}\, a \wedge \mathsf{G}_{[0,20]}\, \neg b.$$

- $\mathcal{M}_\varphi((a,4)) =$ **?**

Examples

Again, consider the property

$$\varphi = \mathsf{F}_{[0,10]}\, a \wedge \mathsf{G}_{[0,20]}\, \neg b.$$

- $\mathcal{M}_\varphi((a,4)) =$ **?**
- $\mathcal{M}_\varphi((a,4),4) =$ **?**

Examples

Again, consider the property

$$\varphi = \mathsf{F}_{[0,10]}\, a \wedge \mathsf{G}_{[0,20]}\, \neg b.$$

- $\mathcal{M}_\varphi((a,4)) =$ **?**
- $\mathcal{M}_\varphi((a,4),4) =$ **?**
- $\mathcal{M}_\varphi((a,4),20) =$ **?**

Examples

Again, consider the property

$$\varphi = \mathsf{F}_{[0,10]}\, a \wedge \mathsf{G}_{[0,20]}\, \neg b.$$

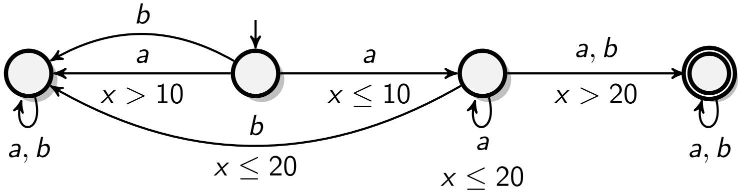
- $\mathcal{M}_\varphi((a,4)) =$ **?**
- $\mathcal{M}_\varphi((a,4),4) =$ **?**
- $\mathcal{M}_\varphi((a,4),20) =$ **?**
- $\mathcal{M}_\varphi((a,4),20+\varepsilon) =$ ✔ for all $\varepsilon > 0$

Timed Büchi Automata

Again, we compile MITL formulas into automata, here timed Büchi automata.

Example

A timed Büchi automaton for $F_{[0,10]}\, a \wedge G_{[0,20]}\, \neg b$:

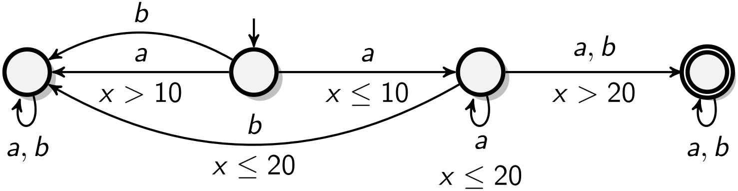


Timed Büchi Automata

Again, we compile MITL formulas into automata, here timed Büchi automata.

Example

A timed Büchi automaton for $F_{[0,10]}\, a \wedge G_{[0,20]}\, \neg b$:



Note

States of a timed Büchi automaton are pairs of locations and clock valuations, functions mapping the clocks to $\mathbb{R}_{\geq 0}$.

On-the-fly Monitoring

Recall that a state q has a nonempty language if there is an infinite run starting in q.

On-the-fly Monitoring

Recall that a state q has a nonempty language if there is an infinite run starting in q.

Lemma

Let $\mathcal{R}_{\mathcal{B}_\varphi}(w, t)$ and $\mathcal{R}_{\mathcal{B}_{\neg\varphi}}(w, t)$ be the sets of states in $\mathcal{B}_\varphi$ and $\mathcal{B}_{\neg\varphi}$ reachable by processing w and then delaying $t - \mathrm{dur}(w)$.

On-the-fly Monitoring

Recall that a state q has a nonempty language if there is an infinite run starting in q.

Lemma

Let $\mathcal{R}_{\mathcal{B}_\varphi}(w,t)$ and $\mathcal{R}_{\mathcal{B}_{\neg\varphi}}(w,t)$ be the sets of states in $\mathcal{B}_\varphi$ and $\mathcal{B}_{\neg\varphi}$ reachable by processing w and then delaying $t - \mathrm{dur}(w)$.

- *$\mathcal{R}_{\mathcal{B}_{\neg\varphi}}(w,t)$ does not contain a state with nonempty language iff $\mathcal{M}_\varphi(w,t) =$ ✔.*

On-the-fly Monitoring

Recall that a state q has a nonempty language if there is an infinite run starting in q.

Lemma

Let $\mathcal{R}_{\mathcal{B}_\varphi}(w, t)$ and $\mathcal{R}_{\mathcal{B}_{\neg\varphi}}(w, t)$ be the sets of states in $\mathcal{B}_\varphi$ and $\mathcal{B}_{\neg\varphi}$ reachable by processing w and then delaying $t - \mathrm{dur}(w)$.

- *$\mathcal{R}_{\mathcal{B}_{\neg\varphi}}(w, t)$ does not contain a state with nonempty language iff $\mathcal{M}_\varphi(w, t) =$ ✔.*
- *$\mathcal{R}_{\mathcal{B}_\varphi}(w, t)$ does not contain a state with nonempty language iff $\mathcal{M}_\varphi(w, t) =$ ✘.*

On-the-fly Monitoring

Recall that a state q has a nonempty language if there is an infinite run starting in q.

Lemma

Let $\mathcal{R}_{\mathcal{B}_\varphi}(w, t)$ and $\mathcal{R}_{\mathcal{B}_{\neg\varphi}}(w, t)$ be the sets of states in $\mathcal{B}_\varphi$ and $\mathcal{B}_{\neg\varphi}$ reachable by processing w and then delaying $t - \mathrm{dur}(w)$.

- *$\mathcal{R}_{\mathcal{B}_{\neg\varphi}}(w, t)$ does not contain a state with nonempty language iff $\mathcal{M}_\varphi(w, t) =$ ✔.*
- *$\mathcal{R}_{\mathcal{B}_\varphi}(w, t)$ does not contain a state with nonempty language iff $\mathcal{M}_\varphi(w, t) =$ ✘.*
- *Otherwise, $\mathcal{M}_\varphi(w, t) =$ **?**.*

Zones

Timed automata may have uncountably many states, i.e., we need symbolic algorithms to represent sets of states.

Zones

Timed automata may have uncountably many states, i.e., we need symbolic algorithms to represent sets of states.

- A zone is a finite conjunction of constraints on the form
  - $x_1 \sim n$, or
  - $x_1 - x_2 \sim n$,

  where x_1, x_2 are clocks, $\sim \in \{<, \leq, =, \geq, >\}$, and $n \in \mathbb{Q}_{\geq 0}$.

Zones

Timed automata may have uncountably many states, i.e., we need symbolic algorithms to represent sets of states.

- A zone is a finite conjunction of constraints on the form
  - $x_1 \sim n$, or
  - $x_1 - x_2 \sim n$,

  where x_1, x_2 are clocks, $\sim \in \{<, \leq, =, \geq, >\}$, and $n \in \mathbb{Q}_{\geq 0}$.
- A zone describes a convex set of clock valuations.

Zones

Timed automata may have uncountably many states, i.e., we need symbolic algorithms to represent sets of states.

- A zone is a finite conjunction of constraints on the form
  - $x_1 \sim n$, or
  - $x_1 - x_2 \sim n$,

  where x_1, x_2 are clocks, $\sim \in \{<, \leq, =, \geq, >\}$, and $n \in \mathbb{Q}_{\geq 0}$.
- A zone describes a convex set of clock valuations.
- A symbolic state is a pair of a location and a zone.

Zone-based Algorithms

There are zone-based algorithms computing

- the set of nonempty language states of a timed Büchi automaton, and
- the set $\mathcal{R}_{\mathcal{B}}(w, t)$ for a given timed Büchi automaton $\mathcal{B}$, given finite timed word w, and $t \geq \mathrm{dur}(w)$.

Zone-based Algorithms

There are zone-based algorithms computing

- the set of nonempty language states of a timed Büchi automaton, and
- the set $\mathcal{R}_{\mathcal{B}}(w, t)$ for a given timed Büchi automaton $\mathcal{B}$, given finite timed word w, and $t \geq \mathrm{dur}(w)$.

Thus, $\mathcal{M}_{\varphi}$ can be effectively computed for MITL properties φ.

MoniTAal

```
tmgr@tmgr-pc:~/repos/MoniTAal/build-bundle/src/monitaal-bin$ ./MoniTAal-bin -p p
ositive ~/monitoring/simple-gearcontroller.xml -n negative ~/monitoring/simple-g
earcontroller.xml
Interactive monitor (respond with "q" to quit)
Next event: @3000 GearReq
Next event: @3900 GearChanged
Next event: @4200 GearReq
Next event: @5200 GearChanged
Monitoring ended, verdict is: NEGATIVE
Monitored 4 events
```

https://github.com/DEIS-Tools/MoniTAal

Outline

1. Monitoring in Discrete Time

2. Monitoring in Real-Time

3. **Monitoring under Delay in Real-Time**

4. Conclusion

Motivation

So far we have assumed that the monitor has direct access to the observations. This is not always realistic!

Motivation

So far we have assumed that the monitor has direct access to the observations. This is not always realistic!

$\mathcal{S}$

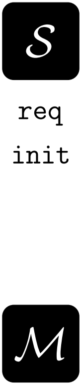
$\mathcal{M}$

Motivation

So far we have assumed that the monitor has direct access to the observations. This is not always realistic!

Motivation

So far we have assumed that the monitor has direct access to the observations. This is not always realistic!

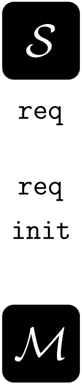


Motivation

So far we have assumed that the monitor has direct access to the observations. This is not always realistic!

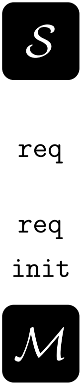
Motivation

So far we have assumed that the monitor has direct access to the observations. This is not always realistic!



Motivation

So far we have assumed that the monitor has direct access to the observations. This is not always realistic!



Motivation

So far we have assumed that the monitor has direct access to the observations. This is not always realistic!

$\mathcal{S}$

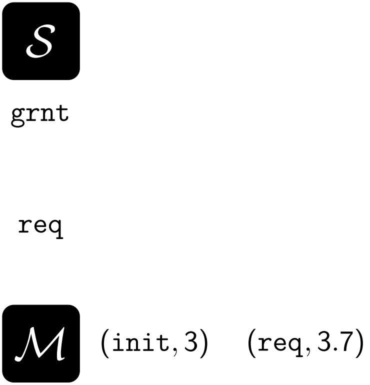
`req`

`req`

$\mathcal{M}$ $(\texttt{init}, 3)$

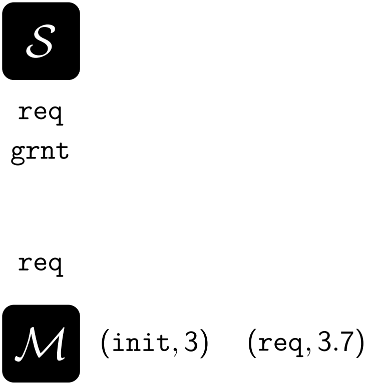
Motivation

So far we have assumed that the monitor has direct access to the observations. This is not always realistic!



Motivation

So far we have assumed that the monitor has direct access to the observations. This is not always realistic!



Motivation

So far we have assumed that the monitor has direct access to the observations. This is not always realistic!

$\mathcal{S}$

req

grnt

$\mathcal{M}$ $(\texttt{init}, 3)$ $(\texttt{req}, 3.7)$ $(\texttt{req}, 4)$

Motivation

So far we have assumed that the monitor has direct access to the observations. This is not always realistic!

$\mathcal{S}$

`req`
`grnt`

$\mathcal{M}$ $(\texttt{init}, 3)$ $(\texttt{req}, 3.7)$ $(\texttt{req}, 4)$

Motivation

So far we have assumed that the monitor has direct access to the observations. This is not always realistic!

$\mathcal{S}$

req

grnt

$\mathcal{M}$ (init, 3) (req, 3.7) (req, 4)

Motivation

So far we have assumed that the monitor has direct access to the observations. This is not always realistic!

$\mathcal{S}$

grnt

req

$\mathcal{M}$ $(\texttt{init}, 3)$ $(\texttt{req}, 3.7)$ $(\texttt{req}, 4)$ $(\texttt{grnt}, 6.12)$

Motivation

So far we have assumed that the monitor has direct access to the observations. This is not always realistic!

$\mathcal{S}$

grnt

$\mathcal{M}$ $(\texttt{init}, 3)$ $(\texttt{req}, 3.7)$ $(\texttt{req}, 4)$ $(\texttt{grnt}, 6.12)$ $(\texttt{req}, 8)$

Delay = Latency + Jitter

Consider a setting where the observations arrive at the monitor via a channel with some delay, and timestamps are assigned at arrival at the monitor.

Delay = Latency + Jitter

Consider a setting where the observations arrive at the monitor via a channel with some delay, and timestamps are assigned at arrival at the monitor.

Delay is modelled as the sum of

- a fixed latency δ plus
- a variable jitter $\leq \varepsilon$.

Delay = Latency + Jitter

Consider a setting where the observations arrive at the monitor via a channel with some delay, and timestamps are assigned at arrival at the monitor.

Delay is modelled as the sum of

- a fixed latency δ plus
- a variable jitter $\leq \varepsilon$.

Often, bounds are known on these quantities: A delay set has the form

$$\mathcal{D} = \{(\delta, \varepsilon) \mid \delta, \varepsilon \in \mathbb{R}_{\geq 0}\}$$

Intuition

Consider the property

$$\mathsf{F}_{[0,10]}\, a \wedge \mathsf{G}_{[0,20]}\, \neg b$$

and the assume we observe

$$(a, 17.3)(b, 27.1)$$

at the monitor, knowing that the jitter is strictly smaller than 0.2:

Intuition

Consider the property

$$\mathsf{F}_{[0,10]}\, a \wedge \mathsf{G}_{[0,20]}\, \neg b$$

and the assume we observe

$$(a, 17.3)(b, 27.1)$$

at the monitor, knowing that the jitter is strictly smaller than 0.2:

- The delay must be at least $17.3 - 10 = 7.3$ to satisfy the constraint $\mathsf{F}_{[0,10]}\, a$.

Intuition

Consider the property

$$F_{[0,10]}\, a \wedge G_{[0,20]}\, \neg b$$

and the assume we observe

$$(a, 17.3)(b, 27.1)$$

at the monitor, knowing that the jitter is strictly smaller than 0.2:

- The delay must be at least $17.3 - 10 = 7.3$ to satisfy the constraint $F_{[0,10]}\, a$.
- The delay must be at most $27.1 - 20 = 7.1$ to satisfy the constraint $G_{[0,20]}\, \neg b$.

Intuition

Consider the property

$$\mathsf{F}_{[0,10]}\, a \wedge \mathsf{G}_{[0,20]}\, \neg b$$

and the assume we observe

$$(a, 17.3)(b, 27.1)$$

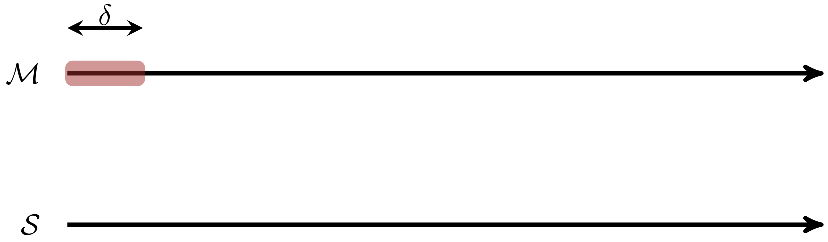
at the monitor, knowing that the jitter is strictly smaller than 0.2:

- The delay must be at least $17.3 - 10 = 7.3$ to satisfy the constraint $\mathsf{F}_{[0,10]}\, a$.
- The delay must be at most $27.1 - 20 = 7.1$ to satisfy the constraint $\mathsf{G}_{[0,20]}\, \neg b$.

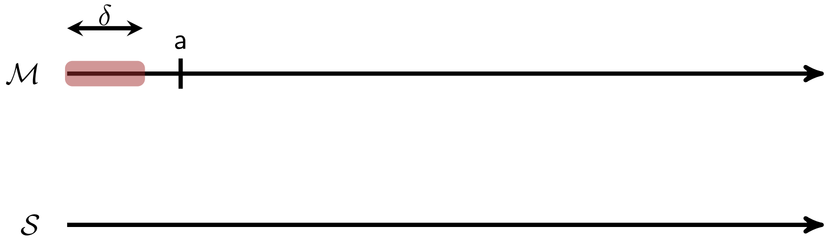
Thus, we can conclude, even under unknown latency, that the property is violated.

Ground Truths

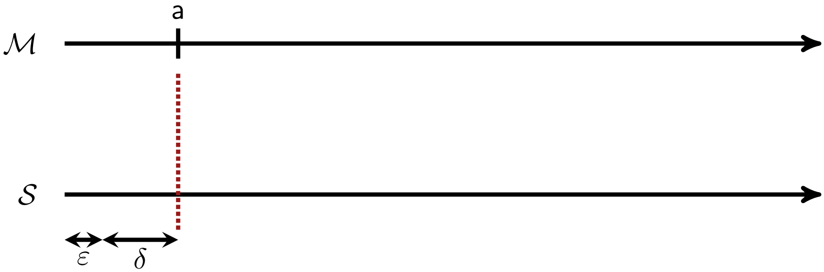
Ground Truths



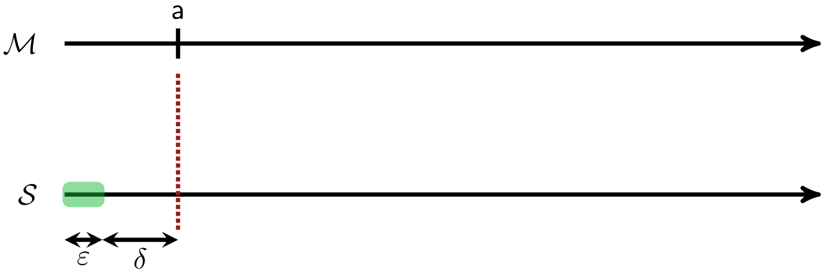
Ground Truths



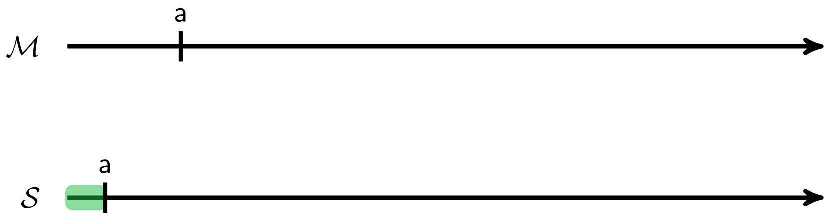
Ground Truths



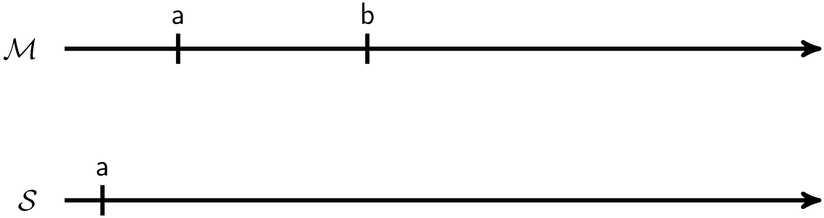
Ground Truths



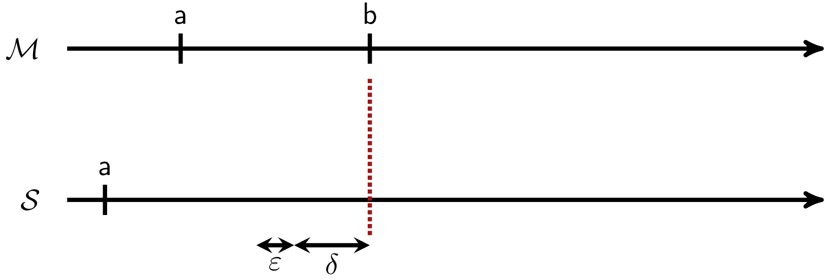
Ground Truths



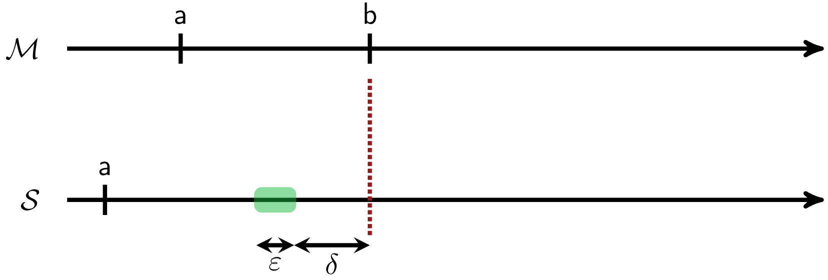
Ground Truths



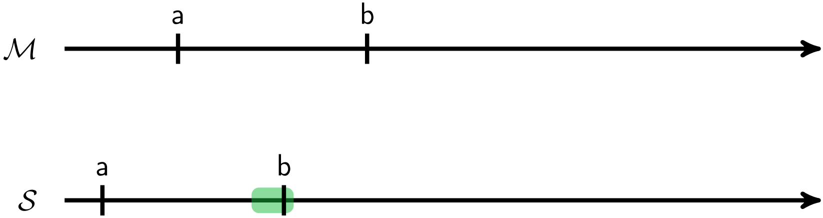
Ground Truths



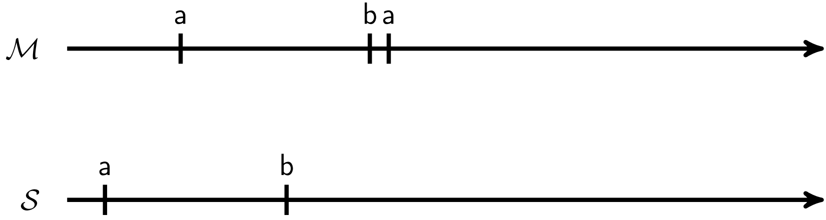
Ground Truths



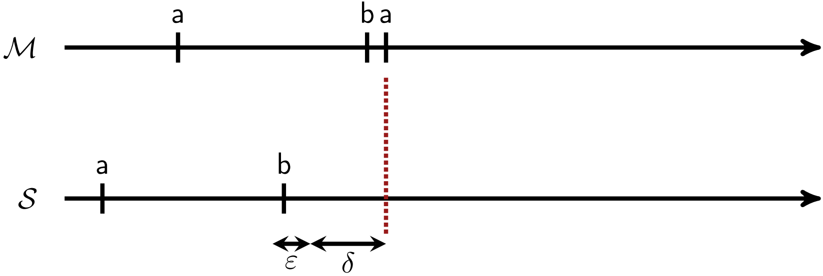
Ground Truths



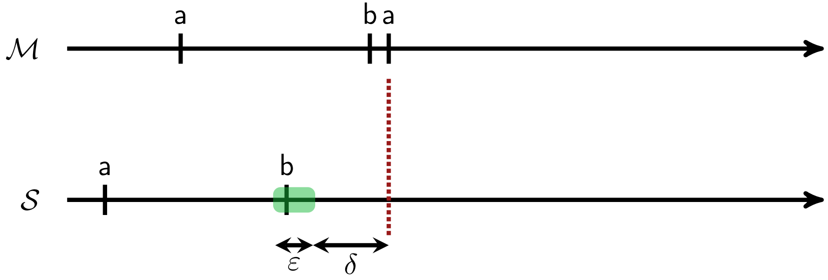
Ground Truths



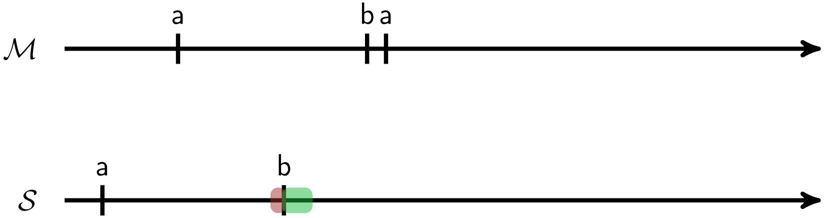
Ground Truths



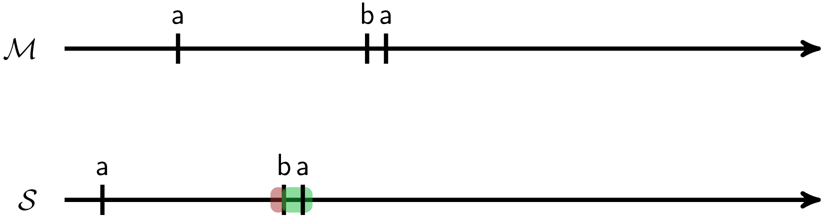
Ground Truths



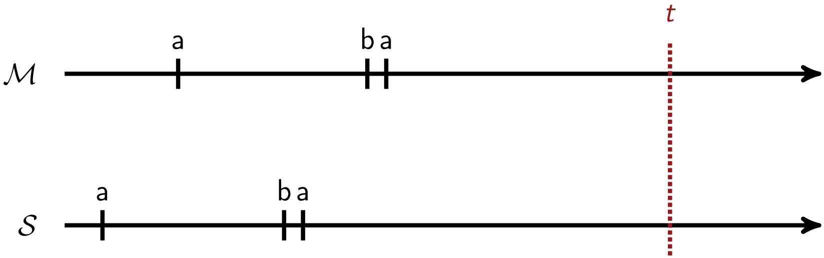
Ground Truths



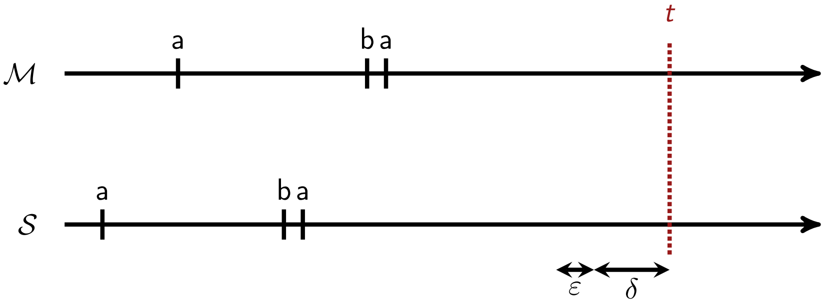
Ground Truths



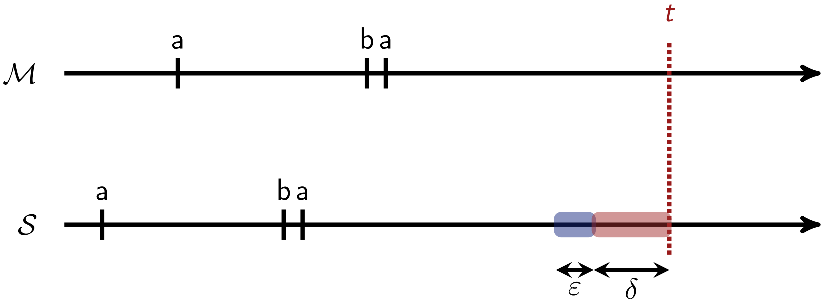
Ground Truths



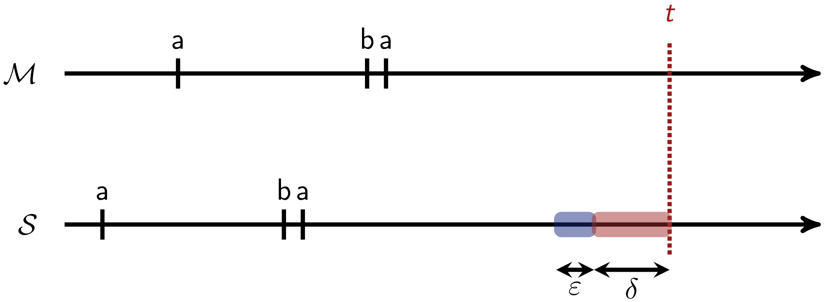
Ground Truths



Ground Truths

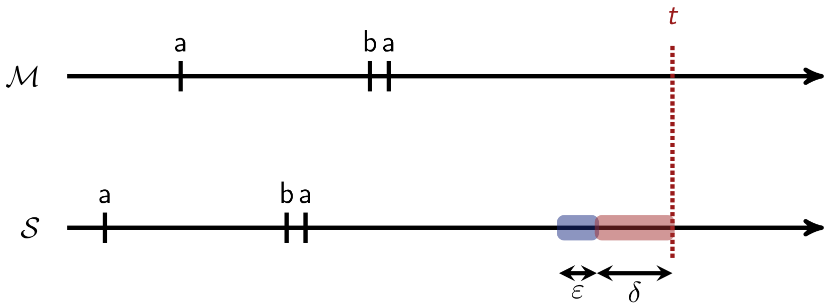


Ground Truths



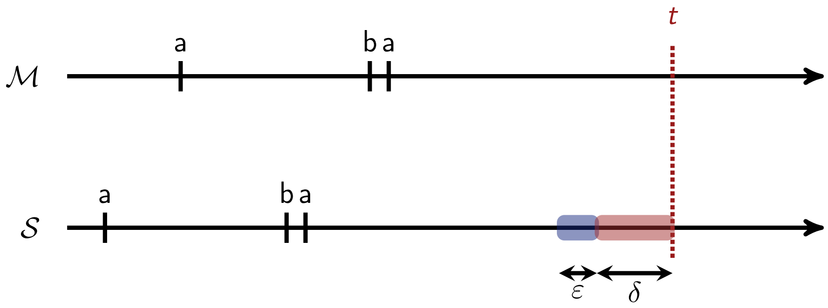
- In the red and blue areas, anything could have happened, as it cannot (may not) have been observed at the monitor at time t.

Ground Truths



- In the red and blue areas, anything could have happened, as it cannot (may not) have been observed at the monitor at time t.
- So, a ground truth for an observation at time t has duration $t - (\delta + \varepsilon)$.

Ground Truths



- In the red and blue areas, anything could have happened, as it cannot (may not) have been observed at the monitor at time t.
- So, a ground truth for an observation at time t has duration $t - (\delta + \varepsilon)$.
- Disclaimer: the full definition is more complicated, as we disallow overtaking!

Monitoring under Delay

Let $\varphi \subseteq T\Sigma^\omega$ be a property and $\mathcal{D}$ a delay set. The monitoring function

$$\mathcal{M}_\varphi^\mathcal{D} \colon T\Sigma^* \times \mathbb{R}_{\geq 0} \to \{✔, ✘, ?\}$$

is defined as

$$\mathcal{M}_\varphi^\mathcal{D}(w, t) = \begin{cases} ✔ & \text{if } w^* \cdot_{t-(\delta+\varepsilon)} \mu \in \varphi \text{ for all } (\delta, \varepsilon) \in \mathcal{D}, \\ & \text{all } w^* \in \mathrm{GT}_\mathcal{D}(w, t), \text{ and all } \mu \in T\Sigma^\omega, \\ ✘ & \text{if } w^* \cdot_{t-(\delta+\varepsilon)} \mu \notin \varphi \text{ for all } (\delta, \varepsilon) \in \mathcal{D}, \\ & \text{all } w^* \in \mathrm{GT}_\mathcal{D}(w, t), \text{ and all } \mu \in T\Sigma^\omega, \\ ? & \text{otherwise,} \end{cases}$$

if w is a timed word compatible with some $(\delta, \varepsilon) \in \mathcal{D}$ and if $t \geq \mathrm{dur}(w)$, otherwise it is undefined.

Monotonicity

Lemma

Let $\varphi \subseteq T\Sigma^\omega$, let $\mathcal{D} \subseteq \mathcal{D}'$ be delay sets, let w be compatible with some $(\delta, \varepsilon) \in \mathcal{D}$, and let $t \geq \mathrm{dur}(w)$. Then,

- $\mathcal{M}_\varphi^{\mathcal{D}'}(w, t) = ✔$ *implies* $\mathcal{M}_\varphi^{\mathcal{D}}(w, t) = ✔$*, and*
- $\mathcal{M}_\varphi^{\mathcal{D}'}(w, t) = ✘$ *implies* $\mathcal{M}_\varphi^{\mathcal{D}}(w, t) = ✘$*.*

Consistent Delays

$$\Delta_{\mathcal{D}}(\varphi, w, t) = \{(\delta, \varepsilon) \in \mathcal{D} \mid \exists w^* \in \mathrm{GT}_{(\delta,\varepsilon)}(w, t) \\ \exists \mu \in T\Sigma^\omega \text{ s.t. } w^* \cdot_{t-(\delta+\varepsilon)} \mu \in \varphi\}.$$

Consistent Delays

$$\Delta_{\mathcal{D}}(\varphi, w, t) = \{(\delta, \varepsilon) \in \mathcal{D} \mid \exists w^* \in \mathrm{GT}_{(\delta,\varepsilon)}(w, t) \\ \exists \mu \in T\Sigma^\omega \text{ s.t. } w^* \cdot_{t-(\delta+\varepsilon)} \mu \in \varphi\}.$$

Lemma

Given $\varphi \subseteq T\Sigma^\omega$, a set $\mathcal{D}$ of delays, w compatible with some $(\delta, \varepsilon) \in \mathcal{D}$, and $t \geq \mathrm{dur}(w)$, we have

1. $\Delta_{\mathcal{D}}(\varphi, w, t) = \emptyset$ *if and only if* $\mathcal{M}_\varphi^{\mathcal{D}}(w, t) =$ ✘, *and*
2. $\Delta_{\mathcal{D}}(\overline{\varphi}, w, t) = \emptyset$ *if and only if* $\mathcal{M}_\varphi^{\mathcal{D}}(w, t) =$ ✔.

Consistent Delays

$$\Delta_{\mathcal{D}}(\varphi, w, t) = \{(\delta, \varepsilon) \in \mathcal{D} \mid \exists w^* \in \mathrm{GT}_{(\delta,\varepsilon)}(w, t)$$
$$\exists \mu \in T\Sigma^\omega \text{ s.t. } w^* \cdot_{t-(\delta+\varepsilon)} \mu \in \varphi\}.$$

Lemma

Let $(w_1, t_1) \sqsubseteq (w_2, t_2)$, $t_1 \geq \mathrm{dur}(w_1)$, $t_2 \geq \mathrm{dur}(w_2)$, and $t_2 \geq t_1$. Then, $\Delta_{\mathcal{D}}(\varphi, w_1, t_1) \supseteq \Delta_{\mathcal{D}}(\varphi, w_2, t_2)$.

Consistent Delays

$$\Delta_{\mathcal{D}}(\varphi, w, t) = \{(\delta, \varepsilon) \in \mathcal{D} \mid \exists w^* \in \mathrm{GT}_{(\delta,\varepsilon)}(w, t)$$
$$\exists \mu \in T\Sigma^\omega \text{ s.t. } w^* \cdot_{t-(\delta+\varepsilon)} \mu \in \varphi\}.$$

Lemma

Let $\varphi \subseteq T\Sigma^\omega$, $\mathcal{D}$ be a set of delays, and $w = (\sigma_0, t_0), \ldots, (\sigma_m, t_m)$ nonempty and compatible with some $(\delta, \varepsilon) \in \mathcal{D}$. Then, for all $t \geq \mathrm{dur}(w)$,

1. *$\Delta_{\mathcal{D}}(\varphi, w, t) \subsetneq \{(\delta, \varepsilon) \in \mathcal{D} \mid \delta \leq t_0\}$ implies there is no $w' \in T\Sigma^*$ such that $\mathcal{M}_\varphi^{\mathcal{D}}(w \cdot_t w', t') = \checkmark$ for any $t' \geq t + \mathrm{dur}(w')$, and*
2. *$\Delta_{\mathcal{D}}(\overline{\varphi}, w, t) \subsetneq \{(\delta, \varepsilon) \in \mathcal{D} \mid \delta \leq t_0\}$ implies there is no $w' \in T\Sigma^*$ such that $\mathcal{M}_\varphi^{\mathcal{D}}(w \cdot_t w', t') = \times$ for any $t' \geq t + \mathrm{dur}(w')$.*

Delayed Monitoring via Automata

Let $\mathcal{R}_{\mathcal{B},\mathcal{D}}(w, t)$ be the set of states of $\mathcal{A}$ reachable by processing a $w^* \in \mathrm{GT}_{\mathcal{D}}(w, t)$ and then delaying $t - (\delta + \varepsilon) - \mathrm{dur}(w^*)$.

Delayed Monitoring via Automata

Let $\mathcal{R}_{\mathcal{B},\mathcal{D}}(w,t)$ be the set of states of $\mathcal{A}$ reachable by processing a $w^* \in \mathrm{GT}_{\mathcal{D}}(w,t)$ and then delaying $t-(\delta+\varepsilon)-\mathrm{dur}(w^*)$.

Lemma (Grosen, Fränzle, Larsen, Z. '24)

- $\mathcal{R}_{\mathcal{B}_{\neg\varphi},\mathcal{D}}(w,t)$ *does not contain a state with nonempty language iff* $\mathcal{M}_{\varphi}(w,t)^{\mathcal{D}} =$ ✔.
- $\mathcal{R}_{\mathcal{B}_{\varphi},\mathcal{D}}(w,t)$ *does not contain a state with nonempty language iff* $\mathcal{M}_{\varphi}(w,t)^{\mathcal{D}} =$ ✘.
- *Otherwise,* $\mathcal{M}_{\varphi}^{\mathcal{D}}(w,t) =$ **?**.

Delayed Monitoring via Automata

Let $\mathcal{R}_{\mathcal{B},\mathcal{D}}(w, t)$ be the set of states of $\mathcal{A}$ reachable by processing a $w^* \in \mathrm{GT}_{\mathcal{D}}(w, t)$ and then delaying $t - (\delta + \varepsilon) - \mathrm{dur}(w^*)$.

Lemma (Grosen, Fränzle, Larsen, Z. '24)

- *$\mathcal{R}_{\mathcal{B}_{\neg\varphi},\mathcal{D}}(w, t)$ does not contain a state with nonempty language iff $\mathcal{M}_{\varphi}(w, t)^{\mathcal{D}} =$ ✔.*
- *$\mathcal{R}_{\mathcal{B}_{\varphi},\mathcal{D}}(w, t)$ does not contain a state with nonempty language iff $\mathcal{M}_{\varphi}(w, t)^{\mathcal{D}} =$ ✘.*
- *Otherwise, $\mathcal{M}_{\varphi}^{\mathcal{D}}(w, t) =$ **?**.*

Lemma (Grosen, Fränzle, Larsen, Z. '24)

Fix $\mathcal{D} = \{(\delta, \varepsilon) \mid \delta \in [\ell, u]\}$ for some $l, u, \varepsilon \in \mathbb{Q}_{\geq 0}$. There is a zone-based algorithm computing $\mathcal{R}_{\mathcal{B},\mathcal{D}}(w, t)$ for a given timed Büchi automaton $\mathcal{B}$, given finite timed word w, and $t \geq \mathrm{dur}(w)$.

Computing Consistent Delays

Recall

$$\Delta_{\mathcal{D}}(\varphi, w, t) = \{(\delta, \varepsilon) \in \mathcal{D} \mid \exists w^* \in \mathrm{GT}_{(\delta,\varepsilon)}(w, t) \\ \exists \mu \in T\Sigma^\omega \text{ s.t. } w^* \cdot_{t-(\delta+\varepsilon)} \mu \in \varphi\}.$$

- Our algorithm extends zones by clocks encoding the latency values that are consistent with the observation processed so far.

Computing Consistent Delays

Recall

$$\Delta_{\mathcal{D}}(\varphi, w, t) = \{(\delta, \varepsilon) \in \mathcal{D} \mid \exists w^* \in \mathrm{GT}_{(\delta,\varepsilon)}(w, t) \\ \exists \mu \in T\Sigma^\omega \text{ s.t. } w^* \cdot_{t-(\delta+\varepsilon)} \mu \in \varphi\}.$$

- Our algorithm extends zones by clocks encoding the latency values that are consistent with the observation processed so far.
- Hence, we are computing $\Delta_{\mathcal{D}}(\varphi, w, t)$ on-the-fly (recall that ε is fixed).

Outline

1. Monitoring in Discrete Time
2. Monitoring in Real-Time
3. Monitoring under Delay in Real-Time
4. **Conclusion**

Conclusion

We have presented a comprehensive framework for monitoring real-time systems using zone-based algorithms for timed automata.

Conclusion

We have presented a comprehensive framework for monitoring real-time systems using zone-based algorithms for timed automata.

Not covered today:

- Monitoring under timing uncertainties.

Conclusion

We have presented a comprehensive framework for monitoring real-time systems using zone-based algorithms for timed automata.

Not covered today:

- Monitoring under timing uncertainties.
- Testing under delay: monitor can send inputs to system via a delayed channel, receives outputs from system.

Conclusion

We have presented a comprehensive framework for monitoring real-time systems using zone-based algorithms for timed automata.

Not covered today:

- Monitoring under timing uncertainties.
- Testing under delay: monitor can send inputs to system via a delayed channel, receives outputs from system.
- Monitoring in real-time under assumptions and partial observations.

Conclusion

We have presented a comprehensive framework for monitoring real-time systems using zone-based algorithms for timed automata.

Not covered today:

- Monitoring under timing uncertainties.
- Testing under delay: monitor can send inputs to system via a delayed channel, receives outputs from system.
- Monitoring in real-time under assumptions and partial observations.
- All these have been implemented in the tool MoniTAal.

Conclusion

We have presented a comprehensive framework for monitoring real-time systems using zone-based algorithms for timed automata.

Not covered today:

- Monitoring under timing uncertainties.
- Testing under delay: monitor can send inputs to system via a delayed channel, receives outputs from system.
- Monitoring in real-time under assumptions and partial observations.
- All these have been implemented in the tool MoniTAal.
- **Friday @CONCUR**: Monitorability of real-time properties.