Runtime Enforcement of CPS against Signal Temporal Logic

Han Su, Saumya Shankar, Srinivas Pinisetty, Partha S. Roop, Naijun Zhan

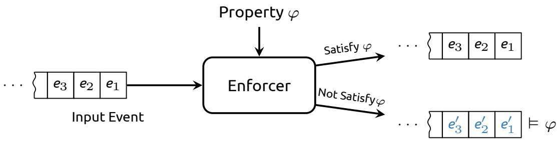
HSCC · Irvine · May 2025

Motivation

Runtime Enforcement is a **lightweight formal method** to monitor the execution of a system at runtime and ensure its complaince against a set of formal requirements.

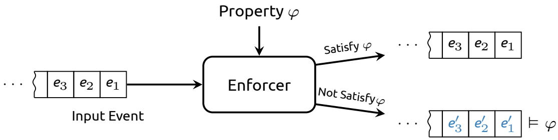
Motivation

Runtime Enforcement is a **lightweight formal method** to monitor the execution of a system at runtime and ensure its complaince against a set of formal requirements.



Motivation

Runtime Enforcement is a **lightweight formal method** to monitor the execution of a system at runtime and ensure its complaince against a set of formal requirements.

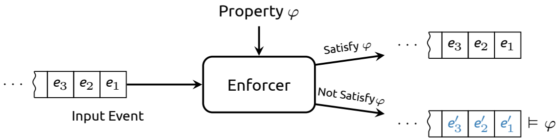


Enforcement Strategies :

- *Fred B. Schneider. "Enforceable security policies"* —— Block Execution
- *Jay Ligatti et al. "Edit automata : enforcement mechanisms for run-time security policies"* —— Suppressing and/or Inserting Actions
- *Srinivas Pinisetty et al. "On the runtime enforcement of timed properties"* —— Delay Actions

Motivation

Runtime Enforcement is a **lightweight formal method** to monitor the execution of a system at runtime and ensure its complaince against a set of formal requirements.

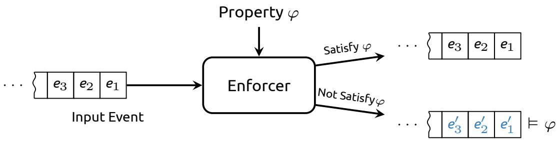


Properties under Consideration :

- *Roderick Bloem et al. "Shield synthesis : Runtime enforcement for reactive systems"* —— Safe DFA
-
- *Partha Roop et al. "Runtime Enforcement of Cyber-Physical Systems"* —— Discrete Timed Automata
- *François Hublet et al. "Proactive Real-Time First-Order Enforcement"* —— Metric First-Order Temporal Logic

Motivation

Runtime Enforcement is a **lightweight formal method** to monitor the execution of a system at runtime and ensure its complaince against a set of formal requirements.

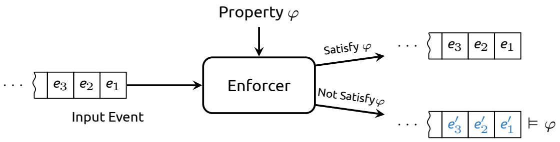


CPS Enforcement Challenges :

- **Real-Time Adaptation** —— delaying reactions or terminating the system is not allowed.
- **Continuous Time Enforcement** —— Discrete time properties are not suitable.

Motivation

Runtime Enforcement is a **lightweight formal method** to monitor the execution of a system at runtime and ensure its complaince against a set of formal requirements.

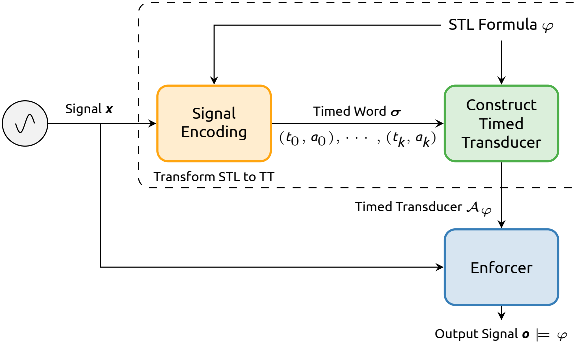


CPS Enforcement Challenges :

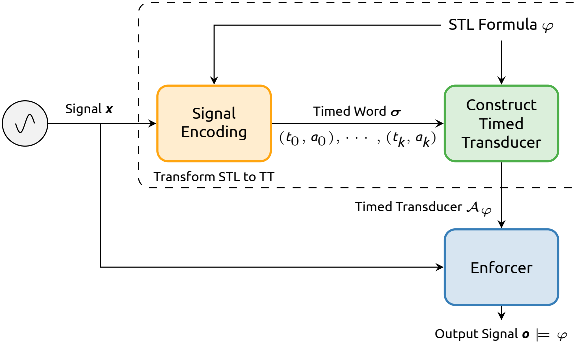
- **Real-Time Adaptation** —— delaying reactions or terminating the system is not allowed.
- **Continuous Time Enforcement** —— Discrete time properties are not suitable.

We proposed an automata-based runtime enforcement framework that **modifies the system's behavior** to satisfy the **STL properties**.

Overall Idea

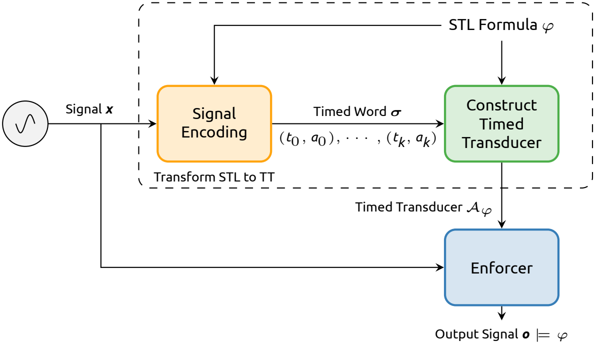


Overall Idea



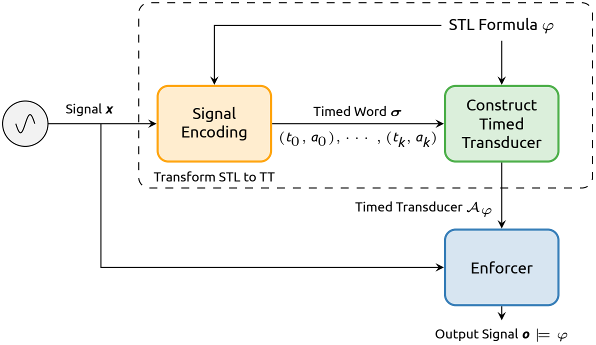
1. Encoding a continuous time signal $\boldsymbol{x}$ as a discrete timed word in accordance with the **STL formula**.

Overall Idea



1. Encoding a continuous time signal $\boldsymbol{x}$ as a discrete timed word in accordance with the **STL formula**.
2. Constructing a **Timed Transducer** $\mathcal{A}_\varphi$ from the given STL formula φ.

Overall Idea



1. Encoding a continuous time signal $\boldsymbol{x}$ as a discrete timed word in accordance with the **STL formula**.
2. Constructing a **Timed Transducer** $\mathcal{A}_\varphi$ from the given STL formula φ.
3. Enforcing a signal $\boldsymbol{x}$ using the TT $\mathcal{A}_\varphi$.

Syntax and Semantics of STL

Signal Temporal Logic (STL)

$$\varphi ::= \top \mid p(\boldsymbol{x}) \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathcal{U}_I \varphi_2 \mid \neg\varphi$$

Syntax and Semantics of STL

Signal Temporal Logic (STL)

$$\varphi ::= \top \mid p(\boldsymbol{x}) \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathcal{U}_I \varphi_2 \mid \neg\varphi$$

$$
\begin{aligned}
&(\boldsymbol{x}, t) \models \top \\
&(\boldsymbol{x}, t) \models p(x) && \text{iff} && p(\boldsymbol{x}(t)) \geq 0 \\
&(\boldsymbol{x}, t) \models \varphi_1 \wedge \varphi_2 && \text{iff} && (\boldsymbol{x}, t) \models \varphi_1 \text{ and } (\boldsymbol{x}, t) \models \varphi_2 \\
&(\boldsymbol{x}, t) \models \varphi_1 \mathcal{U}_{[a,b]} \varphi_2 && \text{iff} && \exists t' \in [t+a, t+b],\ ((\boldsymbol{x}, t') \models \varphi_2, \\
& && && \text{and } \forall t'' \in [t, t'], (\boldsymbol{x}, t'') \models \varphi_1)
\end{aligned}
$$

Syntax and Semantics of STL

Signal Temporal Logic (STL)

$$\varphi ::= \top \mid p(\boldsymbol{x}) \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathcal{U}_I \varphi_2 \mid \neg \varphi$$

Negations appear only adjacent to predicates

STL in Negation Normal Form (NNF)

$$\varphi ::= \top \mid \bot \mid p(\boldsymbol{x}) \mid \neg p(\boldsymbol{x}) \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathcal{U}_I \varphi_2 \mid \varphi_1 \mathcal{R}_I \varphi_2$$

Syntax and Semantics of STL

Signal Temporal Logic (STL)

$$\varphi ::= \top \mid p(\boldsymbol{x}) \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathcal{U}_I \varphi_2 \mid \neg\varphi$$

Negations appear only adjacent to predicates

STL in Negation Normal Form (NNF)

$$\varphi ::= \top \mid \bot \mid p(\boldsymbol{x}) \mid \neg p(\boldsymbol{x}) \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathcal{U}_I \varphi_2 \mid \varphi_1 \mathcal{R}_I \varphi_2$$

$$(\boldsymbol{x}, t) \models \varphi_1 \mathcal{U}_{[a,b]} \varphi_2 \quad \text{iff} \quad \exists t' \in [t+a, t+b],\ ((\boldsymbol{x}, t') \models \varphi_2, \text{ and } \forall t'' \in [t, t'], (\boldsymbol{x}, t'') \models \varphi_1)$$

$$(\boldsymbol{x}, t) \models \varphi_1 \mathcal{R}_{[a,b]} \varphi_2 \quad \text{iff} \quad \forall t' \in [t+a, t+b],\ ((\boldsymbol{x}, t') \models \varphi_2, \text{ or } \exists t'' \in [t, t'], (\boldsymbol{x}, t'') \models \varphi_1)$$

Syntax and Semantics of STL

Signal Temporal Logic (STL)

$$\varphi ::= \top \mid p(\boldsymbol{x}) \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathcal{U}_I \varphi_2 \mid \neg\varphi$$

Negations appear only adjacent to predicates

⇕ **equivalent** [George J. Pappas 2009]

STL in Negation Normal Form (NNF)

$$\varphi ::= \top \mid \bot \mid p(\boldsymbol{x}) \mid \neg p(\boldsymbol{x}) \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathcal{U}_I \varphi_2 \mid \varphi_1 \mathcal{R}_I \varphi_2$$

$$(\boldsymbol{x}, t) \models \varphi_1 \mathcal{U}_{[a,b]} \varphi_2 \quad \text{iff} \quad \exists t' \in [t+a, t+b],\ \big((\boldsymbol{x}, t') \models \varphi_2, \text{ and } \forall t'' \in [t, t'], (\boldsymbol{x}, t'') \models \varphi_1\big)$$

$$(\boldsymbol{x}, t) \models \varphi_1 \mathcal{R}_{[a,b]} \varphi_2 \quad \text{iff} \quad \forall t' \in [t+a, t+b],\ \big((\boldsymbol{x}, t') \models \varphi_2, \text{ or } \exists t'' \in [t, t'], (\boldsymbol{x}, t'') \models \varphi_1\big)$$

Syntax and Semantics of STL

Signal Temporal Logic (STL)

$$\varphi ::= \top \mid p(\boldsymbol{x}) \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathcal{U}_I \varphi_2 \mid \neg\varphi$$

Negations appear only adjacent to predicates

equivalent [George J. Pappas 2009]

STL in Negation Normal Form (NNF)

$$\varphi ::= \top \mid \bot \mid p(\boldsymbol{x}) \mid \neg p(\boldsymbol{x}) \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathcal{U}_I \varphi_2 \mid \varphi_1 \mathcal{R}_I \varphi_2$$

Non-nested STL in NNF

$$\phi ::= \top \mid p(\boldsymbol{x}) \mid \neg p(\boldsymbol{x}) \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2,$$
$$\varphi ::= \phi_1 \mathcal{U}_I \phi_2 \mid \phi_1 \mathcal{R}_I \phi_2 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \,,$$

$$(\boldsymbol{x}, t) \models \varphi_1 \mathcal{U}_{[a,b]} \varphi_2 \quad \text{iff} \quad \exists t' \in [t+a, t+b],\ ((\boldsymbol{x}, t') \models \varphi_2, \text{ and } \forall t'' \in [t, t'], (\boldsymbol{x}, t'') \models \varphi_1)$$

$$(\boldsymbol{x}, t) \models \varphi_1 \mathcal{R}_{[a,b]} \varphi_2 \quad \text{iff} \quad \forall t' \in [t+a, t+b],\ ((\boldsymbol{x}, t') \models \varphi_2, \text{ or } \exists t'' \in [t, t'], (\boldsymbol{x}, t'') \models \varphi_1)$$

Timed Transducer : Timed Automata with Output

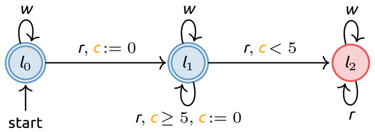
A minimum delay of 5 time units between any two read file requests.

Timed Transducer : Timed Automata with Output

A minimum delay of 5 time units between any two read file requests.

⇓

Timed Automata (TA)

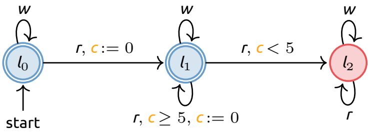


Timed Transducer : Timed Automata with Output

A minimum delay of 5 time units between any two read file requests.

⇓

Timed Automata (TA)



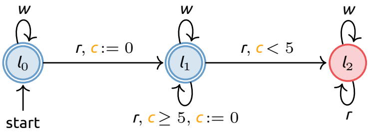
Locations : l_0, l_1, l_2
Initial Location : l_0
Clock : c
Input Alphabet : read file - r, write file - w

Accept Locations : l_0, l_1
Transitions : shown in the figure

Timed Transducer : Timed Automata with Output

A minimum delay of 5 time units between any two read file requests.

Timed Automata (TA) —Add Output $\{\top, \bot\}$→ **Timed Transducer (TT)**

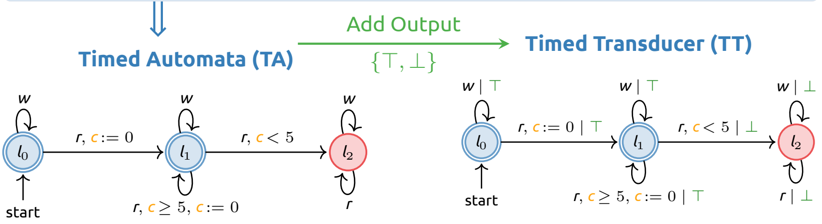


Locations : l_0, l_1, l_2
Initial Location : l_0
Clock : c
Input Alphabet : read file - r, write file - w

Accept Locations : l_0, l_1
Transitions : shown in the figure

Timed Transducer : Timed Automata with Output

A minimum delay of 5 time units between any two read file requests.



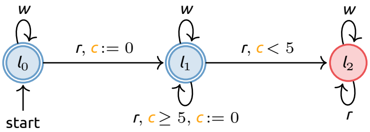
Locations : l_0, l_1, l_2
Initial Location : l_0
Clock : c
Input Alphabet : read file - r, write file - w

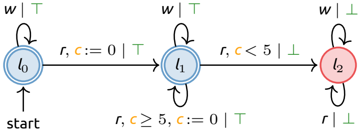
Accept Locations : l_0, l_1
Transitions : shown in the figure

Timed Transducer : Timed Automata with Output

A minimum delay of 5 time units between any two read file requests.

⇓ **Timed Automata (TA)** — Add Output $\{\top, \bot\}$ → **Timed Transducer (TT)**



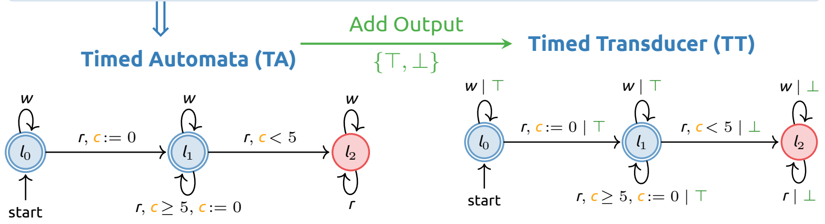


Locations :	l_0, l_1, l_2
Initial Location :	l_0
Clock :	c
Input Alphabet :	read file - r, write file - w
Accept Locations :	l_0, l_1
Transitions :	shown in the figure

Locations :	l_0, l_1, l_2
Initial Location :	l_0
Clock :	c
Input Alphabet :	read file - r, write file - w
Output Alphabet :	$\top, \bot$
Accept Locations :	l_0, l_1
Transitions δ :	shown in the figure
Output Functions :	shown in the figure

Timed Transducer : Timed Automata with Output

A minimum delay of 5 time units between any two read file requests.



Timed Automata (TA)

Locations :	l_0, l_1, l_2
Initial Location :	l_0
Clock :	c
Input Alphabet :	read file - r, write file - w
Accept Locations :	l_0, l_1
Transitions :	shown in the figure

Timed Transducer (TT)

Locations :	l_0, l_1, l_2
Initial Location :	l_0
Clock :	c
Input Alphabet :	read file - r, write file - w
Output Alphabet :	$\top, \bot$
Accept Locations :	l_0, l_1
Transitions δ :	shown in the figure
Output Functions :	shown in the figure

$\top$ **:** current input **may** lead to an accepted run;
$\bot$ **:** current input **definitely** leads to an unacceptable run.

Problem Formulation - Constraints on Enforcer

Given : X - set of signals; φ - STL formula.

Aim : Synthesize $E_\varphi : X \mapsto X$ to satisfy the following constraints :

- **Soundness** :

$$\forall \boldsymbol{x} \in X,\ E_\varphi(\boldsymbol{x}) \models \varphi,$$

- **Transparency** :

$$\forall \boldsymbol{x} \in X,\ \boldsymbol{x} \models \varphi \implies E_\varphi(\boldsymbol{x}) = \boldsymbol{x},$$

- **Minimal Modification** :

$$\forall \boldsymbol{x} \in X,\ \boldsymbol{x} \not\models \varphi \implies E_\varphi(\boldsymbol{x}) = \arg\min_{\boldsymbol{o} \in O} ||\boldsymbol{x} - \boldsymbol{o}||_s,$$

where $O = \{\boldsymbol{o} \mid \boldsymbol{o} \models \varphi \wedge |\boldsymbol{x}| = |\boldsymbol{o}|\}$, $|\boldsymbol{x}|$ is the length of a signal, and $||\boldsymbol{x} - \boldsymbol{o}||_s ::= \max_t ||\boldsymbol{x}(t) - \boldsymbol{o}(t)||$ is the distance between signals, with $|| \cdot ||$ being the Euclidean norm in $\mathbb{R}^n$.

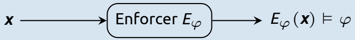
Problem Formulation - Constraints on Enforcer

Given : X - set of signals; φ - STL formula.

Aim : Synthesize $E_\varphi : X \mapsto X$ to satisfy the following constraints :

- **Soundness** :

$$\forall \boldsymbol{x} \in X,\ E_\varphi(\boldsymbol{x}) \models \varphi,$$



- **Transparency** :

$$\forall \boldsymbol{x} \in X,\ \boldsymbol{x} \models \varphi \implies E_\varphi(\boldsymbol{x}) = \boldsymbol{x},$$

- **Minimal Modification** :

$$\forall \boldsymbol{x} \in X,\ \boldsymbol{x} \not\models \varphi \implies E_\varphi(\boldsymbol{x}) = \arg\min_{\boldsymbol{o} \in O} ||\boldsymbol{x} - \boldsymbol{o}||_s,$$

where $O = \{\boldsymbol{o} \mid \boldsymbol{o} \models \varphi \wedge |\boldsymbol{x}| = |\boldsymbol{o}|\}$, $|\boldsymbol{x}|$ is the length of a signal, and $||\boldsymbol{x} - \boldsymbol{o}||_s ::= \max_t ||\boldsymbol{x}(t) - \boldsymbol{o}(t)||$ is the distance between signals, with $||\cdot||$ being the Euclidean norm in $\mathbb{R}^n$.

Problem Formulation - Constraints on Enforcer

Given : X - set of signals; φ - STL formula.

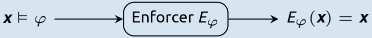
Aim : Synthesize $E_\varphi : X \mapsto X$ to satisfy the following constraints :

- **Soundness** :

$$\forall \boldsymbol{x} \in X,\ E_\varphi(\boldsymbol{x}) \models \varphi,$$

- **Transparency** :

$$\forall \boldsymbol{x} \in X,\ \boldsymbol{x} \models \varphi \implies E_\varphi(\boldsymbol{x}) = \boldsymbol{x},$$



- **Minimal Modification** :

$$\forall \boldsymbol{x} \in X,\ \boldsymbol{x} \not\models \varphi \implies E_\varphi(\boldsymbol{x}) = \arg\min_{\boldsymbol{o} \in O} ||\boldsymbol{x} - \boldsymbol{o}||_s,$$

where $O = \{\boldsymbol{o} \mid \boldsymbol{o} \models \varphi \wedge |\boldsymbol{x}| = |\boldsymbol{o}|\}$, $|\boldsymbol{x}|$ is the length of a signal, and $||\boldsymbol{x} - \boldsymbol{o}||_s ::= \max_t ||\boldsymbol{x}(t) - \boldsymbol{o}(t)||$ is the distance between signals, with $||\cdot||$ being the Euclidean norm in $\mathbb{R}^n$.

Problem Formulation - Constraints on Enforcer

Given : X - set of signals; φ - STL formula.

Aim : Synthesize $E_\varphi : X \mapsto X$ to satisfy the following constraints :

- **Soundness** :

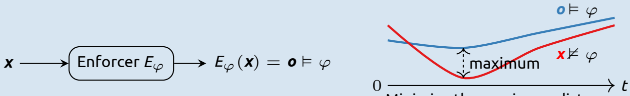
$$\forall \boldsymbol{x} \in X,\ E_\varphi(\boldsymbol{x}) \models \varphi,$$

- **Transparency** :

$$\forall \boldsymbol{x} \in X,\ \boldsymbol{x} \models \varphi \implies E_\varphi(\boldsymbol{x}) = \boldsymbol{x},$$

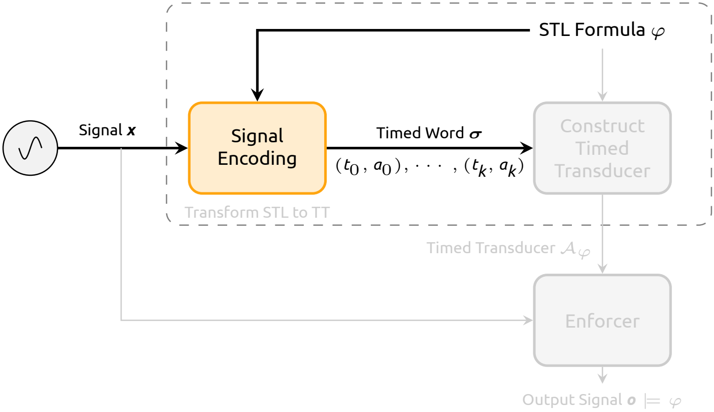
- **Minimal Modification** :

$$\forall \boldsymbol{x} \in X,\ \boldsymbol{x} \not\models \varphi \implies E_\varphi(\boldsymbol{x}) = \arg\min_{\boldsymbol{o} \in O} ||\boldsymbol{x} - \boldsymbol{o}||_s,$$



Minimize the maximum distance between $\boldsymbol{x}$ and $\boldsymbol{o}$

Signal Encoding : Variable Points + Relevant Points

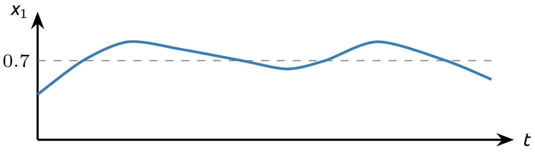


Variable Points *[Jia Lee et al. 2019]*

A variable point is where the **truth value** of a predicate regarding the signal **changes**.

Variable Points *[Jia Lee et al. 2019]*

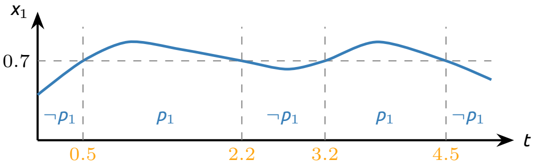
A variable point is where the **truth value** of a predicate regarding the signal **changes**.



Predicate p_1 :
$x_1 - 0.7 \geq 0$

Variable Points *[Jia Lee et al. 2019]*

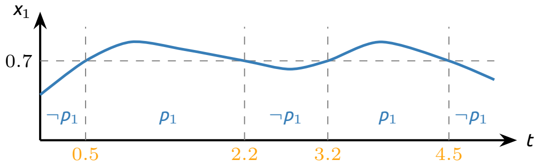
A variable point is where the **truth value** of a predicate regarding the signal **changes**.



Predicate p_1 :
$x_1 - 0.7 \geq 0$

Variable Points *[Jia Lee et al. 2019]*

A variable point is where the **truth value** of a predicate regarding the signal **changes**.

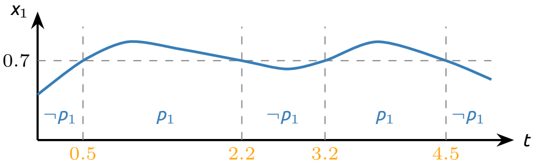


Predicate p_1 :
$x_1 - 0.7 \geq 0$

Variable Points :
$t = 0.5, 2.2, 3.2, 4.5$

Variable Points *[Jia Lee et al. 2019]*

A variable point is where the **truth value** of a predicate regarding the signal **changes**.



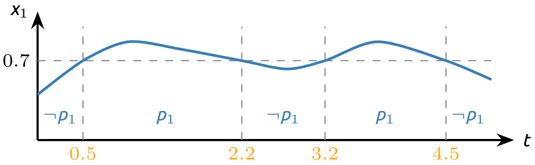
Predicate p_1 :
$x_1 - 0.7 \geq 0$

Variable Points :
$t = 0.5, 2.2, 3.2, 4.5$

The truth value of $p_1(x_1(t))$ remains constant within each open interval $(0.5, 2.2), (2.2, 3.2), (3.2, 4.5)$!

Variable Points *[Jia Lee et al. 2019]*

A variable point is where the **truth value** of a predicate regarding the signal **changes**.



Predicate p_1 :
$x_1 - 0.7 \geq 0$

Variable Points :
$t = 0.5, 2.2, 3.2, 4.5$

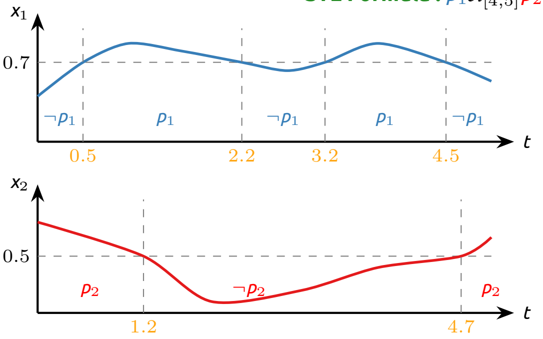
The truth value of $p_1(x_1(t))$ remains constant within each open interval $(0.5, 2.2), (2.2, 3.2), (3.2, 4.5)$!

Signal x_1, Predicate p_1 $\xrightarrow{\text{Encoding}}$ Timed Word : $(0.5, p_1), (2.2, \neg p_1), (3.2, p_1), (4.5, \neg p_1)$

Variable Points *[Jia Lee et al. 2019]*

A variable point is where the **truth value** of a predicate regarding the signal **changes**.

STL Formula : $p_1 \mathcal{U}_{[4,5]} p_2$



Predicate p_1 :
$x_1 - 0.7 \geq 0$

Variable Points :
$t = 0.5, 2.2, 3.2, 4.5$

Predicate p_2 :
$x_2 - 0.5 \geq 0$

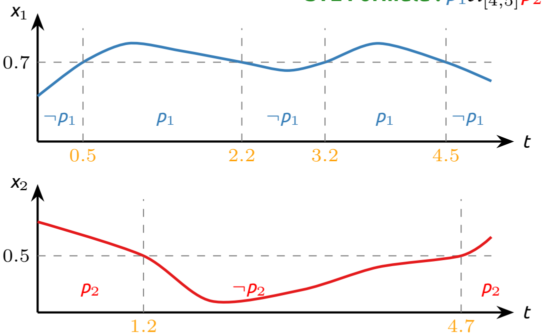
Variable Points :
$t = 1.2, 4.7$

Signal (x_1, x_2)
STL $p_1 \mathcal{U}_{[4,5]} p_2$ $\xrightarrow{\text{Encoding}}$ $(0.5, p_1 \wedge p_2), (1.2, p_1 \wedge \neg p_2), (2.2, \neg p_1 \wedge \neg p_2), (3.2, p_1 \wedge \neg p_2), (4.5, \neg p_1 \wedge \neg p_2), (4.7, \neg p_1 \wedge p_2)$

Variable Points *[Jia Lee et al. 2019]*

A variable point is where the **truth value** of a predicate regarding the signal **changes**.

STL Formula : $p_1 \mathcal{U}_{[4,5]} p_2$



Predicate p_1 :
$x_1 - 0.7 \geq 0$

Variable Points :
$t = 0.5, 2.2, 3.2, 4.5$

Predicate p_2 :
$x_2 - 0.5 \geq 0$

Variable Points :
$t = 1.2, 4.7$

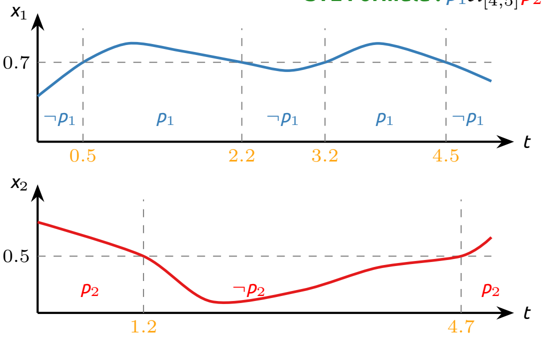
Signal (x_1, x_2)
STL $p_1 \mathcal{U}_{[4,5]} p_2$
$\xrightarrow{\text{Encoding}}$
$(0.5, p_1 \wedge p_2), (1.2, p_1 \wedge \neg p_2), (2.2, \neg p_1 \wedge \neg p_2),$
$(3.2, p_1 \wedge \neg p_2), (4.5, \neg p_1 \wedge \neg p_2), (4.7, \neg p_1 \wedge p_2)$

Does that provide **sufficient** information for enforcing compliance of an STL property?

Variable Points [Jia Lee et al. 2019]

A variable point is where the **truth value** of a predicate regarding the signal **changes**.

STL Formula : $p_1 \mathcal{U}_{[4,5]} p_2$



Predicate p_1 :
$x_1 - 0.7 \geq 0$

Variable Points :
$t = 0.5, 2.2, 3.2, 4.5$

Predicate p_2 :
$x_2 - 0.5 \geq 0$

Variable Points :
$t = 1.2, 4.7$

Signal (x_1, x_2)
STL $p_1 \mathcal{U}_{[4,5]} p_2$ $\xrightarrow{\text{Encoding}}$ $(0.5, p_1 \wedge p_2), (1.2, p_1 \wedge \neg p_2), (2.2, \neg p_1 \wedge \neg p_2),$
$(3.2, p_1 \wedge \neg p_2), (4.5, \neg p_1 \wedge \neg p_2), (4.7, \neg p_1 \wedge p_2)$

Does that provide **sufficient** information for enforcing compliance of an STL property? NO! Information for $t \in (0, 0.5)$ — $(\neg p_1 \wedge p_2)$ — is missing!

Relevant Points

Relevant points:
- **Interval boundaries of the given formula**
- **Initial instant of the given signal**

Definition

Given an STL formula φ, the set of relevant points $rp(\varphi)$ is inductively defined by :

$$
\begin{aligned}
rp(\top) = \emptyset, \quad rp(\varphi_1 \wedge \varphi_2) = rp(\varphi_1) \cup rp(\varphi_2), \\
rp(p(x)) = \{0\}, \quad rp(\varphi_1 \vee \varphi_2) = rp(\varphi_1) \cup rp(\varphi_2), \\
rp(\varphi_1 \mathcal{U}_{[t_1, t_2]} \varphi_2) = \{t_1, t_2\} \cup rp(\varphi_1) \cup rp(\varphi_2), \\
rp(\varphi_1 \mathcal{R}_{[t_1, t_2]} \varphi_2) = \{t_1, t_2\} \cup rp(\varphi_1) \cup rp(\varphi_2).
\end{aligned}
$$

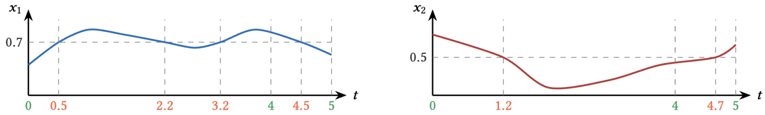
Relevant Points

Relevant points { **Interval boundaries of the given formula** / **Initial instant of the given signal** }

Definition

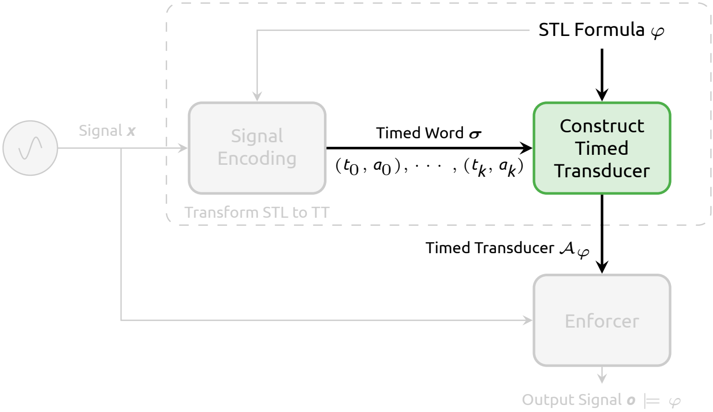
Given an STL formula φ, the set of relevant points $rp(\varphi)$ is inductively defined by :

$$rp(\top) = \emptyset, \quad rp(\varphi_1 \wedge \varphi_2) = rp(\varphi_1) \cup rp(\varphi_2),$$
$$rp(p(x)) = \{0\}, \quad rp(\varphi_1 \vee \varphi_2) = rp(\varphi_1) \cup rp(\varphi_2),$$
$$rp(\varphi_1 \mathcal{U}_{[t_1,t_2]} \varphi_2) = \{t_1, t_2\} \cup rp(\varphi_1) \cup rp(\varphi_2),$$
$$rp(\varphi_1 \mathcal{R}_{[t_1,t_2]} \varphi_2) = \{t_1, t_2\} \cup rp(\varphi_1) \cup rp(\varphi_2).$$

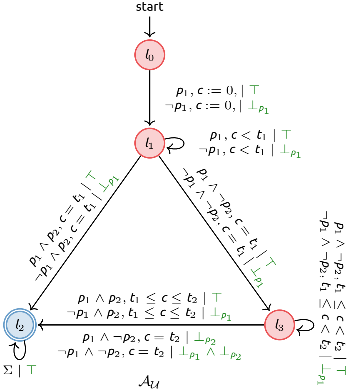


timestamps / actions: $\binom{0}{\neg p_1 \wedge p_2}\binom{0.5}{p_1 \wedge p_2}\binom{1.2}{p_1 \wedge \neg p_2}\binom{2.2}{\neg p_1 \wedge \neg p_2}\binom{3.2}{p_1 \wedge \neg p_2}\binom{4}{p_1 \wedge \neg p_2}\binom{4.5}{\neg p_1 \wedge \neg p_2}\binom{4.7}{\neg p_1 \wedge p_2}\binom{5}{\neg p_1 \wedge p_2}$

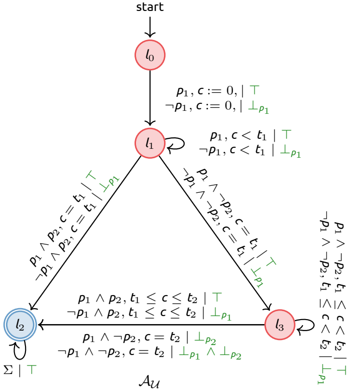
Construction of TT : $\mathcal{A}_{\mathcal{U}} + \mathcal{A}_{\mathcal{R}}$ + Composition



Timed Transducer for $p_1\mathcal{U}_{[t_1,t_2]}p_2$



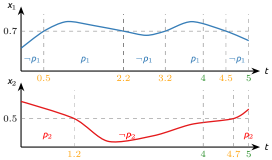
Timed Transducer for $p_1\mathcal{U}_{[t_1,t_2]}p_2$



STL : $p_1\mathcal{U}_{[4,5]}p_2$, with

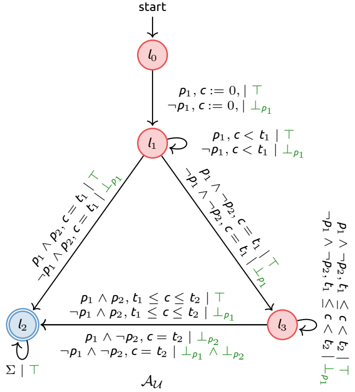
- $p_1 \equiv x_1 - 0.7 \geq 0$,
- $p_2 \equiv x_2 - 0.5 \geq 0$.

Signal :



Timed Word : $(0, \neg p_1 \wedge p_2), (0.5, p_1 \wedge p_2), (1.2, p_1 \wedge \neg p_2), (2.2, \neg p_1 \wedge \neg p_2), (3.2, p_1 \wedge \neg p_2), (4, p_1 \wedge \neg p_2), (4.5, \neg p_1 \wedge \neg p_2), (4.7, \neg p_1 \wedge p_2), (5, \neg p_1 \wedge p_2)$

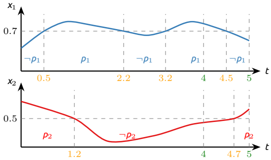
Timed Transducer for $p_1\mathcal{U}_{[t_1,t_2]}p_2$



STL : $p_1\mathcal{U}_{[4,5]}p_2$, with

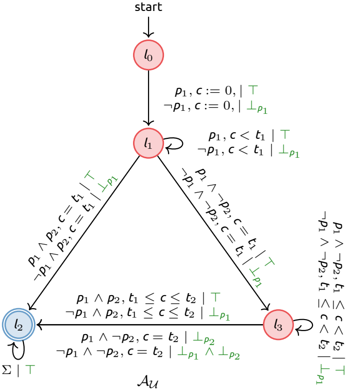
- $p_1 \equiv x_1 - 0.7 \geq 0$,
- $p_2 \equiv x_2 - 0.5 \geq 0$.

Signal :



Timed Word : $(0, \neg p_1 \wedge p_2), (0.5, p_1 \wedge p_2), (1.2, p_1 \wedge \neg p_2), (2.2, \neg p_1 \wedge \neg p_2), (3.2, p_1 \wedge \neg p_2), (4, p_1 \wedge \neg p_2), (4.5, \neg p_1 \wedge \neg p_2), (4.7, \neg p_1 \wedge p_2), (5, \neg p_1 \wedge p_2)$

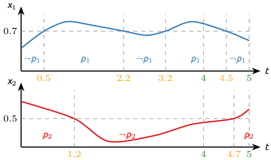
Timed Transducer for $p_1\mathcal{U}_{[t_1,t_2]}p_2$



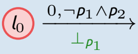
STL : $p_1\mathcal{U}_{[4,5]}p_2$, with

- $p_1 \equiv x_1 - 0.7 \geq 0$,
- $p_2 \equiv x_2 - 0.5 \geq 0$.

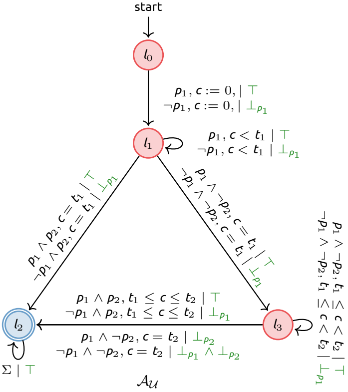
Signal :



Timed Word : $(0, \neg p_1 \wedge p_2), (0.5, p_1 \wedge p_2), (1.2, p_1 \wedge \neg p_2), (2.2, \neg p_1 \wedge \neg p_2), (3.2, p_1 \wedge \neg p_2), (4, p_1 \wedge \neg p_2), (4.5, \neg p_1 \wedge \neg p_2), (4.7, \neg p_1 \wedge p_2), (5, \neg p_1 \wedge p_2)$



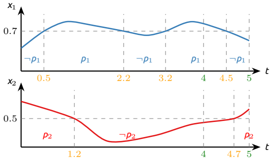
Timed Transducer for $p_1\mathcal{U}_{[t_1,t_2]}p_2$



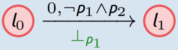
STL : $p_1\mathcal{U}_{[4,5]}p_2$, with

- $p_1 \equiv x_1 - 0.7 \geq 0$,
- $p_2 \equiv x_2 - 0.5 \geq 0$.

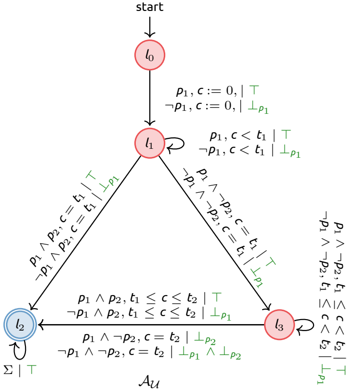
Signal :



Timed Word : $(0, \neg p_1 \wedge p_2), (0.5, p_1 \wedge p_2), (1.2, p_1 \wedge \neg p_2), (2.2, \neg p_1 \wedge \neg p_2), (3.2, p_1 \wedge \neg p_2), (4, p_1 \wedge \neg p_2), (4.5, \neg p_1 \wedge \neg p_2), (4.7, \neg p_1 \wedge p_2), (5, \neg p_1 \wedge p_2)$



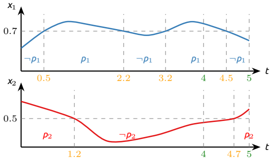
Timed Transducer for $p_1\mathcal{U}_{[t_1,t_2]}p_2$



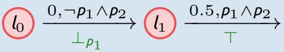
STL : $p_1\mathcal{U}_{[4,5]}p_2$, with

- $p_1 \equiv x_1 - 0.7 \geq 0,$
- $p_2 \equiv x_2 - 0.5 \geq 0.$

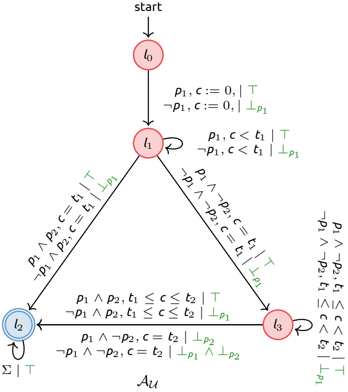
Signal :



Timed Word : $(0, \neg p_1 \wedge p_2), (0.5, p_1 \wedge p_2), (1.2, p_1 \wedge \neg p_2), (2.2, \neg p_1 \wedge \neg p_2), (3.2, p_1 \wedge \neg p_2), (4, p_1 \wedge \neg p_2), (4.5, \neg p_1 \wedge \neg p_2), (4.7, \neg p_1 \wedge p_2), (5, \neg p_1 \wedge p_2)$



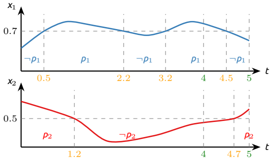
Timed Transducer for $p_1\mathcal{U}_{[t_1,t_2]}p_2$



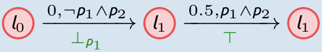
STL : $p_1\mathcal{U}_{[4,5]}p_2$, with

- $p_1 \equiv x_1 - 0.7 \geq 0$,
- $p_2 \equiv x_2 - 0.5 \geq 0$.

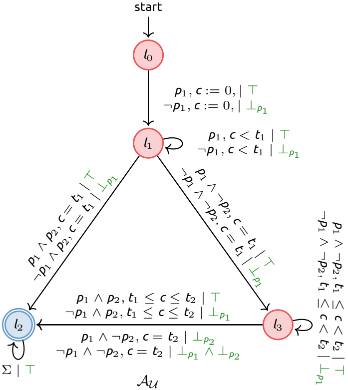
Signal :



Timed Word : $(0, \neg p_1 \wedge p_2), (0.5, p_1 \wedge p_2), (1.2, p_1 \wedge \neg p_2), (2.2, \neg p_1 \wedge \neg p_2), (3.2, p_1 \wedge \neg p_2), (4, p_1 \wedge \neg p_2), (4.5, \neg p_1 \wedge \neg p_2), (4.7, \neg p_1 \wedge p_2), (5, \neg p_1 \wedge p_2)$



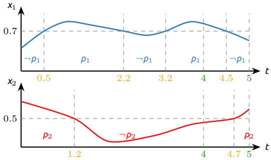
Timed Transducer for $p_1\mathcal{U}_{[t_1,t_2]}p_2$



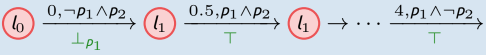
STL : $p_1\mathcal{U}_{[4,5]}p_2$, with

- $p_1 \equiv x_1 - 0.7 \geq 0,$
- $p_2 \equiv x_2 - 0.5 \geq 0.$

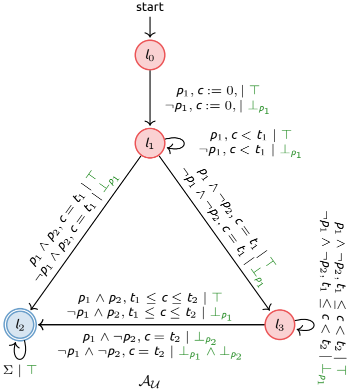
Signal :



Timed Word : $(0, \neg p_1 \wedge p_2), (0.5, p_1 \wedge p_2), (1.2, p_1 \wedge \neg p_2), (2.2, \neg p_1 \wedge \neg p_2), (3.2, p_1 \wedge \neg p_2), (4, p_1 \wedge \neg p_2), (4.5, \neg p_1 \wedge \neg p_2), (4.7, \neg p_1 \wedge p_2), (5, \neg p_1 \wedge p_2)$



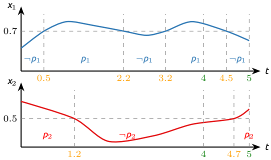
Timed Transducer for $p_1\mathcal{U}_{[t_1,t_2]}p_2$



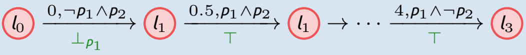
STL : $p_1\mathcal{U}_{[4,5]}p_2$, with

- $p_1 \equiv x_1 - 0.7 \ge 0,$
- $p_2 \equiv x_2 - 0.5 \ge 0.$

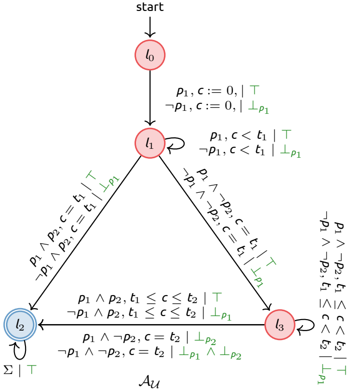
Signal :



Timed Word : $(0, \neg p_1 \wedge p_2), (0.5, p_1 \wedge p_2), (1.2, p_1 \wedge \neg p_2), (2.2, \neg p_1 \wedge \neg p_2), (3.2, p_1 \wedge \neg p_2), (4, p_1 \wedge \neg p_2), (4.5, \neg p_1 \wedge \neg p_2), (4.7, \neg p_1 \wedge p_2), (5, \neg p_1 \wedge p_2)$



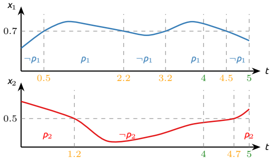
Timed Transducer for $p_1\mathcal{U}_{[t_1,t_2]}p_2$



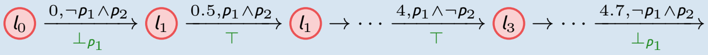
STL : $p_1\mathcal{U}_{[4,5]}p_2$, with

- $p_1 \equiv x_1 - 0.7 \geq 0,$
- $p_2 \equiv x_2 - 0.5 \geq 0.$

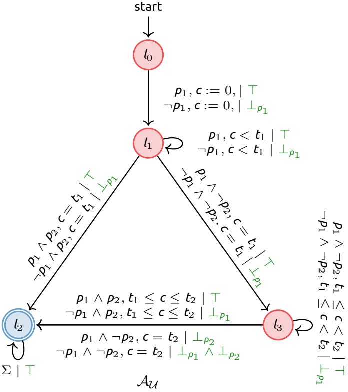
Signal :



Timed Word : $(0, \neg p_1 \wedge p_2), (0.5, p_1 \wedge p_2), (1.2, p_1 \wedge \neg p_2), (2.2, \neg p_1 \wedge \neg p_2), (3.2, p_1 \wedge \neg p_2), (4, p_1 \wedge \neg p_2), (4.5, \neg p_1 \wedge \neg p_2), (4.7, \neg p_1 \wedge p_2), (5, \neg p_1 \wedge p_2)$



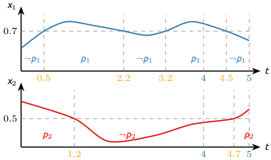
Timed Transducer for $p_1\mathcal{U}_{[t_1,t_2]}p_2$



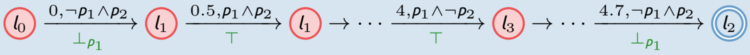
STL : $p_1\mathcal{U}_{[4,5]}p_2$, with

- $p_1 \equiv x_1 - 0.7 \geq 0$,
- $p_2 \equiv x_2 - 0.5 \geq 0$.

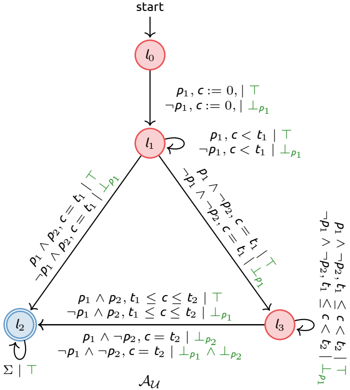
Signal :



Timed Word : $(0, \neg p_1 \wedge p_2), (0.5, p_1 \wedge p_2), (1.2, p_1 \wedge \neg p_2), (2.2, \neg p_1 \wedge \neg p_2), (3.2, p_1 \wedge \neg p_2), (4, p_1 \wedge \neg p_2), (4.5, \neg p_1 \wedge \neg p_2), (4.7, \neg p_1 \wedge p_2), (5, \neg p_1 \wedge p_2)$



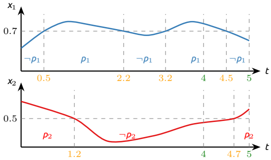
Timed Transducer for $p_1\mathcal{U}_{[t_1,t_2]}p_2$



STL : $p_1\mathcal{U}_{[4,5]}p_2$, with

- $p_1 \equiv x_1 - 0.7 \geq 0$,
- $p_2 \equiv x_2 - 0.5 \geq 0$.

Signal :



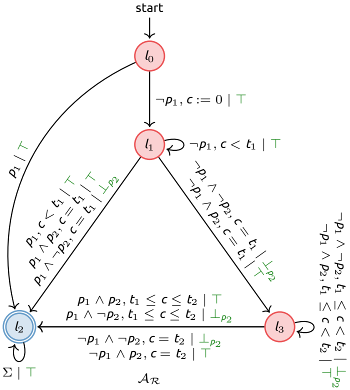
Timed Word : $(0, \neg p_1 \wedge p_2), (0.5, p_1 \wedge p_2), (1.2, p_1 \wedge \neg p_2), (2.2, \neg p_1 \wedge \neg p_2), (3.2, p_1 \wedge \neg p_2), (4, p_1 \wedge \neg p_2), (4.5, \neg p_1 \wedge \neg p_2), (4.7, \neg p_1 \wedge p_2), (5, \neg p_1 \wedge p_2)$

Proposition

Let $\mathbf{x}$ be a signal and $\boldsymbol{\sigma}$ denote its encoded timed word against the STL formula $p_1\mathcal{U}_{[t_1,t_2]}p_2$. Define $\boldsymbol{\omega}_\top$ as the timed word where all event actions are $\top$. The following equivalence is then established :

$$[\![\mathcal{A}_\mathcal{U}]\!](\boldsymbol{\sigma}) = \boldsymbol{\omega}_\top \iff \mathbf{x} \models p_1\mathcal{U}_{[t_1,t_2]}p_2$$

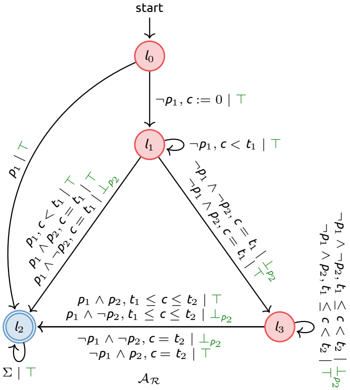
Timed Transducer for $\mathcal{R}_I$



TT for $p_1 \mathcal{R}_{[t_1, t_2]} p_2$:

- $L = \{l_0, l_1, l_2, l_3\}$;
- $l_0 = l_0$;
- $\mathcal{C} = \{c\}$;
- $\Sigma = \{p_1, p_2\}$;
- $\Lambda = \{\top, \bot_{p_1}, \bot_{p_2}\}$;
- $F = \{l_2\}$.

Timed Transducer for $\mathcal{R}_I$



TT for $p_1 \mathcal{R}_{[t_1, t_2]} p_2$:

- $L = \{l_0, l_1, l_2, l_3\}$;
- $l_0 = l_0$;
- $\mathcal{C} = \{c\}$;
- $\Sigma = \{p_1, p_2\}$;
- $\Lambda = \{\top, \bot_{p_1}, \bot_{p_2}\}$;
- $F = \{l_2\}$.

Proposition

Let $\boldsymbol{x}$ be a signal and $\boldsymbol{\sigma}$ denote its encoded timed word against the given STL formula $p_1 \mathcal{R}_{[t_1, t_2]} p_2$. Let $\boldsymbol{\omega}_\top$ be defined as before. The following equivalence is then established :

$$[\![\mathcal{A}_\mathcal{R}]\!](\boldsymbol{\sigma}) = \boldsymbol{\omega}_\top \iff \boldsymbol{x} \models p_1 \mathcal{R}_{[t_1, t_2]} p_2$$

Composition between TT

Given two TTs $\mathcal{A}_1 = (L_1, l_0^1, \mathcal{C}_1, \Sigma_1, \Lambda_1, \Delta_1, \lambda_1, F_1)$ and $\mathcal{A}_2 = (L_2, l_0^2, \mathcal{C}_2, \Sigma_2, \Lambda_2, \Delta_2, \lambda_2, F_2)$, the **$\wedge$-product** automaton $\mathcal{A}_1 \times_\wedge \mathcal{A}_2 ::= (L, l_0, \mathcal{C}, \Sigma, \Lambda, \Delta, \lambda, F)$, where

- $L = L_1 \times L_2$,
- $l_0 = (l_0^1, l_0^2)$,
- $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$,
- $\Sigma = \Sigma_1 \cup \Sigma_2$,
- $\Lambda = \Lambda_1 \cup \Lambda_2$,
- $\delta = \big((l_1, l_2), (a_1, a_2), g_1 \wedge g_2, \mathcal{C}_1' \cup \mathcal{C}_2', (l_1', l_2')\big) \in \Delta$ iff $\delta_1 = (l_1, a_1, g_1, \mathcal{C}_1', l_1') \in \Delta_1$ and $\delta_2 = (l_2, a_2, g_2, \mathcal{C}_2', l_2') \in \Delta_2$,
- $\lambda(\delta) = \lambda_1(\delta_1) \wedge \lambda_2(\delta_2)$,
- $F = F_1 \times F_2$.

Given two TTs $\mathcal{A}_1 = (L_1, l_0^1, \mathcal{C}_1, \Sigma_1, \Lambda_1, \Delta_1, \lambda_1, F_1)$ and $\mathcal{A}_2 = (L_2, l_0^2, \mathcal{C}_2, \Sigma_2, \Lambda_2, \Delta_2, \lambda_2, F_2)$, the **$\vee$-product** automaton $\mathcal{A}_1 \times_\vee \mathcal{A}_2 ::= (L, l_0, \mathcal{C}, \Sigma, \Lambda, \Delta, \lambda, F)$, where

- $L = L_1 \times L_2$,
- $l_0 = (l_0^1, l_0^2)$,
- $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$,
- $\Sigma = \Sigma_1 \cup \Sigma_2$,
- $\Lambda = \Lambda_1 \cup \Lambda_2$,
- $\delta = \big((l_1, l_2), (a_1, a_2), g_1 \wedge g_2, \mathcal{C}_1' \cup \mathcal{C}_2', (l_1', l_2')\big) \in \Delta$ iff $\delta_1 = (l_1, a_1, g_1, \mathcal{C}_1', l_1') \in \Delta_1$ and $\delta_2 = (l_2, a_2, g_2, \mathcal{C}_2', l_2') \in \Delta_2$,
- $\lambda(\delta) = \lambda_1(\delta_1) \vee \lambda_2(\delta_2)$,
- $F = (F_1 \times L_2) \cup (L_1 \times F_2)$.

Composition between TT

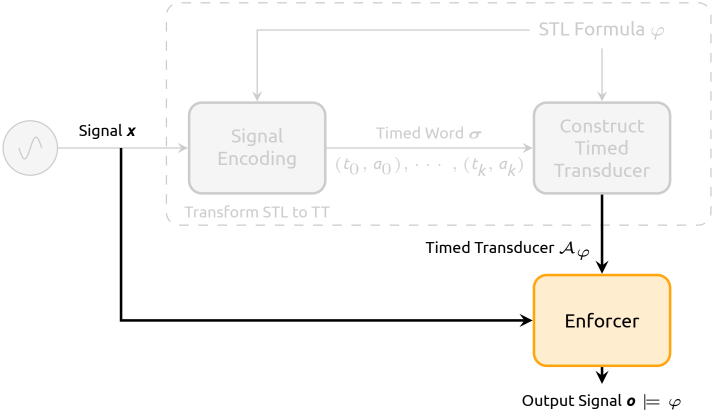
Given two TTs
$\mathcal{A}_1 = (L_1, l_0^1, \mathcal{C}_1, \Sigma_1, \Lambda_1, \Delta_1, \lambda_1, F_1)$ and
$\mathcal{A}_2 = (L_2, l_0^2, \mathcal{C}_2, \Sigma_2, \Lambda_2, \Delta_2, \lambda_2, F_2)$, the
$\wedge$-product automaton
$\mathcal{A}_1 \times_\wedge \mathcal{A}_2 ::= (L, l_0, \mathcal{C}, \Sigma, \Lambda, \Delta, \lambda, F)$,
where

- $L = L_1 \times L_2$,
- $l_0 = (l_0^1, l_0^2)$,
- $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$,
- $\Sigma = \Sigma_1 \cup \Sigma_2$,
- $\Lambda = \Lambda_1 \cup \Lambda_2$,
- $\delta = ((l_1, l_2), (a_1, a_2), g_1 \wedge g_2, \mathcal{C}_1' \cup \mathcal{C}_2', (l_1', l_2')) \in \Delta$ iff
  $\delta_1 = (l_1, a_1, g_1, \mathcal{C}_1', l_1') \in \Delta_1$ and
  $\delta_2 = (l_2, a_2, g_2, \mathcal{C}_2', l_2') \in \Delta_2$,
- $\lambda(\delta) = \lambda_1(\delta_1) \wedge \lambda_2(\delta_2)$,
- $F = F_1 \times F_2$.

Given two TTs
$\mathcal{A}_1 = (L_1, l_0^1, \mathcal{C}_1, \Sigma_1, \Lambda_1, \Delta_1, \lambda_1, F_1)$ and
$\mathcal{A}_2 = (L_2, l_0^2, \mathcal{C}_2, \Sigma_2, \Lambda_2, \Delta_2, \lambda_2, F_2)$, the
$\vee$-product automaton
$\mathcal{A}_1 \times_\vee \mathcal{A}_2 ::= (L, l_0, \mathcal{C}, \Sigma, \Lambda, \Delta, \lambda, F)$,
where

- $L = L_1 \times L_2$,
- $l_0 = (l_0^1, l_0^2)$,
- $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$,
- $\Sigma = \Sigma_1 \cup \Sigma_2$,
- $\Lambda = \Lambda_1 \cup \Lambda_2$,
- $\delta = ((l_1, l_2), (a_1, a_2), g_1 \wedge g_2, \mathcal{C}_1' \cup \mathcal{C}_2', (l_1', l_2')) \in \Delta$ iff
  $\delta_1 = (l_1, a_1, g_1, \mathcal{C}_1', l_1') \in \Delta_1$ and
  $\delta_2 = (l_2, a_2, g_2, \mathcal{C}_2', l_2') \in \Delta_2$,
- $\lambda(\delta) = \lambda_1(\delta_1) \vee \lambda_2(\delta_2)$,
- $F = (F_1 \times L_2) \cup (L_1 \times F_2)$.

Automata-Based Runtime Enforcement



Minimal Modification of Signal : Optimization-Based

A $\bot_{p_k}$ output $\Rightarrow$ the truth value of p_k should be flipped.

Minimal Modification of Signal : Optimization-Based

A $\perp_{p_k}$ output $\Rightarrow$ the truth value of p_k should be flipped.

- Constructing replaced variable $x[x^{p_k}/y]$:

$$x^{p_1} ::= \{x_1, x_2, x_m\}, \quad \cdots, \quad x^{p_k} ::= \{x_m, x_{m+1}, x_n\}$$

$$p_1 \equiv \mu_1(x_1, x_2, x_m) \geq 0, \cdots, p_k \equiv \mu_k(x_m, x_{m+1}, x_n) \geq 0$$

$$x = (x_1, x_2, x_3, \cdots, x_m, x_{m+1}, \cdots, x_{n-1}, x_n)$$

$$x[x^{p_k}/y] = (x_1, x_2, x_3, \cdots, y_1, y_2, \cdots, x_{n-1}, y_3)$$

Minimal Modification of Signal : Optimization-Based

A $\perp_{p_k}$ output $\Rightarrow$ the truth value of p_k should be flipped.

- Constructing replaced variable $x[x^{p_k}/y]$:

$$x^{p_1} ::= \{x_1, x_2, x_m\}, \quad \cdots, \quad x^{p_k} ::= \{x_m, x_{m+1}, x_n\}$$

$$p_1 \equiv \mu_1(x_1, x_2, x_m) \geq 0, \; \cdots, \; p_k \equiv \mu_k(x_m, x_{m+1}, x_n) \geq 0$$

$$x \quad = \left(x_1, \; x_2, \; x_3, \; \cdots, \; x_m, \; x_{m+1}, \; \cdots, \; x_{n-1}, \; x_n \right)$$

$$x[x^{p_k}/y] = \left(x_1, \; x_2, \; x_3, \; \cdots, \; y_1, \; y_2, \; \cdots, \; x_{n-1}, \; y_3 \right)$$

- Solve the optimization problem Modify $(x$, input action a, output action $b, \varphi)$:

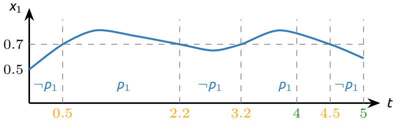
$$\begin{aligned} \text{Minimize :} \quad & \|y - x^{p_k}\| \\ \text{Subject to :} \quad & \mu_i(x[x^{p_k}/y]) \geq 0, \quad \forall p_i \in a, \\ & \mu_j(x[x^{p_k}/y]) < 0, \quad \forall \neg p_j \in a, \\ & \mu_k(y) \bowtie 0, \end{aligned} \tag{1}$$

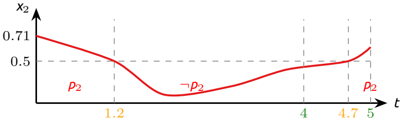
where $\bowtie$ is $\geq$ if $\neg p_k \in a$ and $\bowtie$ is $<$ otherwise.

Minimal Modification : Example

STL Formula : $p_1 \mathcal{U}_{[4,5]} p_2$, with $p_1 \equiv x_1 - 0.7 \geq 0$, $p_2 \equiv x_2 - 0.5 \geq 0$.

Signal $x = (x_1, x_2)$:



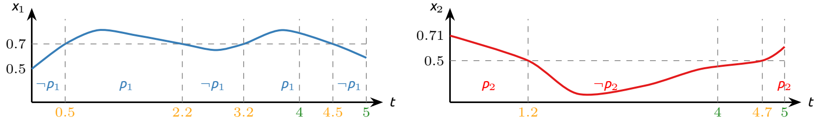


Input at $t = 0$: $(0, \neg p_1 \wedge p_2)$;
Output at $t = 0$: $\perp_{p_1}$.

Minimal Modification : Example

STL Formula : $p_1 \mathcal{U}_{[4,5]} p_2$, with $p_1 \equiv x_1 - 0.7 \geq 0$, $p_2 \equiv x_2 - 0.5 \geq 0$.

Signal $x = (x_1, x_2)$:



Input at $t = 0$: $(0, \neg p_1 \wedge p_2)$;
Output at $t = 0$: $\perp_{p_1}$.

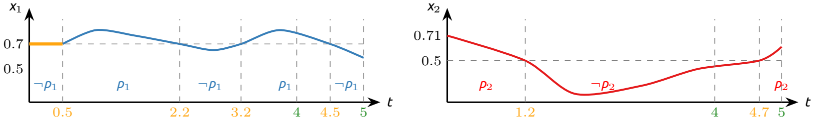
Replaced Variable : $x[x^{p_1}/y] = (y, 0.71)$
Optimization Problem :

$$\begin{aligned} \text{Minimize :} \quad & ||y - 0.5|| \\ \text{Subject to :} \quad & 0.71 - 0.5 \geq 0, \\ & y - 0.7 \geq 0, \end{aligned}$$

Minimal Modification : Example

STL Formula : $p_1 \mathcal{U}_{[4,5]} p_2$, with $p_1 \equiv x_1 - 0.7 \geq 0$, $p_2 \equiv x_2 - 0.5 \geq 0$.

Signal $x = (x_1, x_2)$ **:**



Input at $t = 0$ **:** $(0, \neg p_1 \wedge p_2)$;
Output at $t = 0$ **:** $\perp_{p_1}$.

Replaced Variable : $x[x^{p_1}/y] = (y, 0.71)$
Optimization Problem :

$$\begin{aligned} \text{Minimize}: \quad & ||y - 0.5|| \\ \text{Subject to}: \quad & 0.71 - 0.5 \geq 0, \quad \Longrightarrow \quad y = 0.7 \\ & y - 0.7 \geq 0, \end{aligned}$$

Algorithm and Theoretical Results

Signal Encoding

Algorithm 1: SignEncode(φ)

Require : φ: STL formula
Ensure : σ: time word encoded from $\boldsymbol{x}$

Rele ← $rp(\varphi)$, Vari ← $vv(\varphi)$, Pred ← $pd(\varphi)$;
while *true* **do**
  $\boldsymbol{x}$ ← await_signal();
  t ← current_time() ; // Get the current time t
  CurrPred ← Truth values of predicates $p \in$ Pred with respect to $\boldsymbol{x}$ at t;
  if $\boldsymbol{x}(t) \in$ Vari *or* $t \in$ Rele **then**
    Emit (t, CurrPred);

Runtime Enforcement

Algorithm 2: Algorithm Enforcer $E_\varphi(\boldsymbol{x})$

$\mathcal{A}_\varphi$ ← TT constructed from φ ;
currState ← $[l_0, c := 0]$;
while *true* **do**
  (t, a) ← event emitted by Alg. 1;
  currState, b = make_transition$_{\mathcal{A}_\varphi}$(currState, t, a);
  if $b \neq \top$ **then**
    $\boldsymbol{x}(t)$ = Modify($\boldsymbol{x}(t)$, a, b, φ);
  release $\boldsymbol{x}$;

Algorithm and Theoretical Results

Signal Encoding

Algorithm 1: SignEncode(φ)

Require : φ: STL formula
Ensure : σ: time word encoded from $\boldsymbol{x}$

```
Rele ← rp(φ), Vari ← vv(φ), Pred ← pd(φ);
while true do
    x ← await_signal();
    t ← current_time() ;     // Get the current time t
    CurrPred ← Truth values of predicates p ∈ Pred with
      respect to x at t;
    if x(t) ∈ Vari or t ∈ Rele then
        Emit (t, CurrPred);
```

Runtime Enforcement

Algorithm 2: Algorithm Enforcer $E_\varphi(\boldsymbol{x})$

```
A_φ ← TT constructed from φ ;
currState ← [l_0, c := 0];
while true do
    (t, a) ← event emitted by Alg. 1;
    currState, b = make_transition_{A_φ}(currState, t, a);
    if b ≠ ⊤ then
        x(t) = Modify(x(t), a, b, φ);
    release x;
```

Theorem

*Given an STL formula φ and a signal $\mathbf{x}$, the enforcer E_φ in Alg. 2 can enforce $\mathbf{x}$ to satisfy φ, while ensuring that the **soundness**, **transparency**, and **minimal modification** conditions are met.*

Algorithm and Theoretical Results

Signal Encoding

Algorithm 1: SignEncode(φ)

```
Require: φ: STL formula
Ensure : σ: time word encoded from x
Rele ← rp(φ), Vari ← vv(φ), Pred ← pd(φ);
while true do
    x ← await_signal();
    t ← current_time() ;     // Get the current time t
    CurrPred ← Truth values of predicates p ∈ Pred with
      respect to x at t;
    if x(t) ∈ Vari or t ∈ Rele then
        Emit (t, CurrPred);
```

Runtime Enforcement

Algorithm 2: Algorithm Enforcer $E_\varphi(\boldsymbol{x})$

```
A_φ ← TT constructed from φ ;
currState ← [l_0, c := 0];
while true do
    (t, a) ← event emitted by Alg. 1;
    currState, b = make_transition_{A_φ}(currState, t, a);
    if b ≠ ⊤ then
        x(t) = Modify(x(t), a, b, φ);
    release x;
```

Theorem

*Given an STL formula φ and a signal $\mathbf{x}$, the enforcer E_φ in Alg. 2 can enforce $\mathbf{x}$ to satisfy φ, while ensuring that the **soundness**, **transparency**, and **minimal modification** conditions are met.*

- **Soundness :** The output event $\perp_p$ - How to modify the input action to get the output action $\top$.
- **Transparency :** The output event $\top$ - Leave the input action unchanged.
- **Minimal Modification :** The construction of the optimization problem.

Experimantal Evaluation

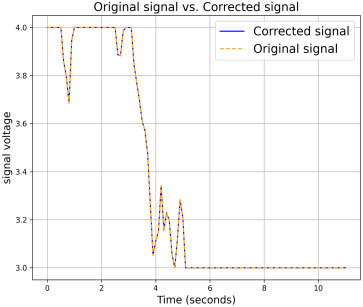
Table 2: Experimental results with varying violation points in the signal

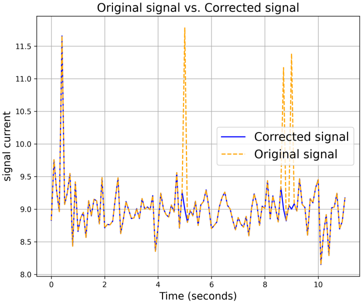
$\#v$	*Safe stopping of AVs*		*Safe charging of AVs*		*Safe deceleration of AVs*	
	len(σ)	time(ms)	len(σ)	time(ms)	len(σ)	time(ms)
2	8	0.077	7	0.079	9	0.111
4	12	0.078	11	0.091	13	0.131
6	16	0.138	15	0.113	17	0.169
8	16	0.099	19	0.175	21	0.209
10	22	0.132	21	0.182	23	0.218
12	22	0.135	23	0.170	22	0.238
14	28	0.196	24	0.181	27	0.284
16	32	0.175	33	0.247	25	0.244
18	26	0.148	31	0.236	35	0.339
20	24	0.142	29	0.216	27	0.268

len(σ): the length of time word encoded from the signal; $\#v$: the number of violation points in signal

Wxperimental Evaluation

- Safe Charging of Autonomous Vehicles (3 violation points)





Summary

- **Contribution :**
  - This work proposed a method to encode **dense time signals** into **timed words** while preserving the information required to adjust the compliance of the signals with STL formulas.
  - This work introduced a uniform approach to **construct timed transducers against STL formulae**, enabling these transducers to enforce the compliance of the STL formula on the input timed word.
  - This work developed a method to **minimally modify the signal to ensure its satisfaction against the given STL formula**

Han Su, Saumya Shankar, Srinivas Pinisetty, Partha S. Roop, and Naijun Zhan : Runtime Enforcement of CPS against Signal Temporal Logic. HSCC '25.

Summary

- **Contribution :**
  - This work proposed a method to encode **dense time signals** into **timed words** while preserving the information required to adjust the compliance of the signals with STL formulas.
  - This work introduced a uniform approach to **construct timed transducers against STL formulae**, enabling these transducers to enforce the compliance of the STL formula on the input timed word.
  - This work developed a method to **minimally modify the signal to ensure its satisfaction against the given STL formula**
- **Future Work :**
  - Extend the range of **STL formulae** to include **nested STL formulae**.
  - Explore the possibility of **bi-direction enforcement process** to improve efficiency.

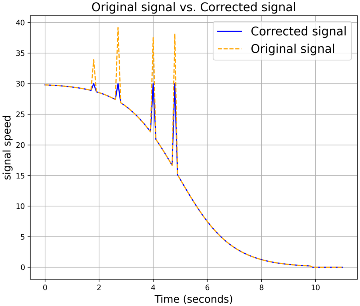
Han Su, Saumya Shankar, Srinivas Pinisetty, Partha S. Roop, and Naijun Zhan : Runtime Enforcement of CPS against Signal Temporal Logic. HSCC '25.

Efficiency Evaluation

- Safe Stopping of Autonomous Vehicles : $(v \leq 30)\mathcal{U}_{[5,10]}(v = 0)$
- Safe Charging of Autonomous Vehicles : $(V = 4.2)\mathcal{R}_{[2,10]}(I < 10)$
- Safe Deceleration of Autonomous Vehicles :$(w \leq 30)\mathcal{U}_{[5,10]}(w = 0) \wedge (m \leq 30)\mathcal{U}_{[5,10]}(m = 0)$

Efficiency Evaluation

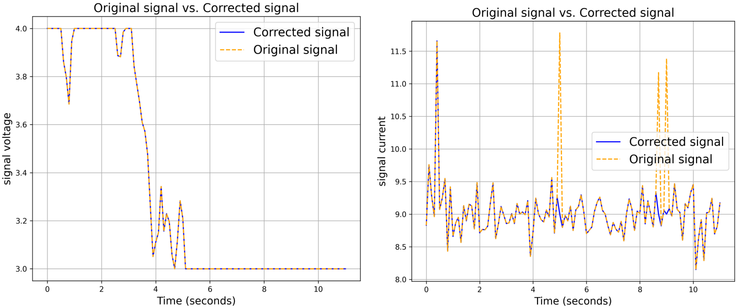
- Safe Stopping of Autonomous Vehicles : $(v \leq 30)\mathcal{U}_{[5,10]}(v = 0)$



- Safe Charging of Autonomous Vehicles : $(V = 4.2)\mathcal{R}_{[2,10]}(I < 10)$
- Safe Deceleration of Autonomous Vehicles :$(w \leq 30)\mathcal{U}_{[5,10]}(w = 0) \wedge (m \leq 30)\mathcal{U}_{[5,10]}(m = 0)$

Efficiency Evaluation

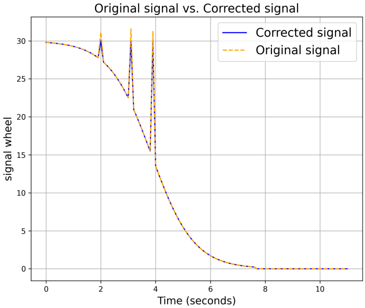
- **Safe Stopping of Autonomous Vehicles :** $(v \leq 30)\mathcal{U}_{[5,10]}(v = 0)$
- **Safe Charging of Autonomous Vehicles :** $(V = 4.2)\mathcal{R}_{[2,10]}(I < 10)$

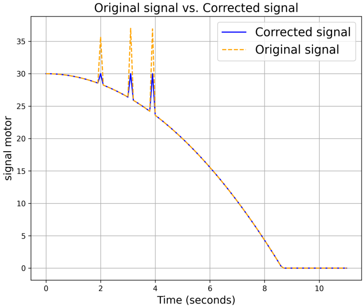


- **Safe Deceleration of Autonomous Vehicles :** $(w \leq 30)\mathcal{U}_{[5,10]}(w = 0) \wedge (m \leq 30)\mathcal{U}_{[5,10]}(m = 0)$

Efficiency Evaluation

- Safe Stopping of Autonomous Vehicles : $(v \leq 30)\mathcal{U}_{[5,10]}(v = 0)$
- Safe Charging of Autonomous Vehicles : $(V = 4.2)\mathcal{R}_{[2,10]}(I < 10)$
- Safe Deceleration of Autonomous Vehicles :$(w \leq 30)\mathcal{U}_{[5,10]}(w = 0) \wedge (m \leq 30)\mathcal{U}_{[5,10]}(m = 0)$





Types of Predicates in STL Supported

- **Linear Predicates**
  - $x(t) > c$ (signal greater than a constant)
    - Example : $x(t) < 30$
  - $a_1 x_1(t) + a_2 x_2(t) + \cdots + a_n x_n(t) \leq c$ (linear combination comparison)
    - Example : $2x(t) + 3y(t) \geq 5$
- **Non-linear Predicates** (but they complicate monitoring and enforcement)
  - $x(t)^2 + y(t)^2 \leq 1$
  - $\sin(x(t)) > 0.5$
- **Boolean Combinations of Predicates** (Predicates can be combined using logical operators)
  - Example : $(x(t) < 30) \wedge (y(t) > 10)$

Nesting of STL Formulas Not Supported

Nesting of STL formulas is not supported. For example, a nested formula like :

$$x < 5 \, \mathbf{U}_{[0,5]} \, \left(y > 1 \, \mathbf{R}_{[2,4]} \, (z = 0)\right)$$

is disallowed.
We do support connections between two sub-formulas that each contain temporal operators and are connected using either conjunction $(\wedge)$ or disjunction $(\vee)$.

Bounded Enforcement

In our approach, enforcement is performed such that the signal satisfies the given STL formula at time $t = 0$. We do not perform continuous enforcement over the entire signal trace.
However, in the context of cyber-physical systems (CPS), this limited enforcement may not be sufficient. In such systems, both monitoring and enforcement are expected to be active at all times, continuously checking and correcting the signal to ensure safety and correctness throughout the system's operation.
What we do is basically **bounded enforcement** (enforcing over a finite interval like $[0, T]$) or even **pointwise enforcement** (enforcing at $t = 0$).

Continuous Enforcement (Future Work)

Continuous enforcement is our future work. This can be enabled by including support for nested STL formulas. For example :

- "Always, until some condition, another property must eventually hold."
- Example formula :

$$\mathbf{G}_{[0,10]}\left(x < 5\ \mathbf{U}_{[2,4]}\ (y > 1)\right)$$

Unidirectional Enforcement

Currently, we perform enforcement in a single direction. For **bidirectional enforcement** (i.e., enforcement from plant to controller and controller to plant), we would need to employ two of our enforcers operating in each direction.