Bounded-Memory Runtime Enforcement of Timed Properties

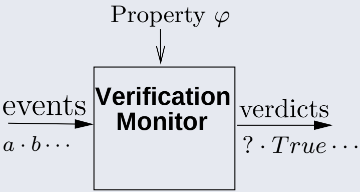
Saumya Shankar[1], Srinivas Pinisetty[1], Thierry Jéron[2]

[1] Indian Institute of Technology Bhubaneswar, India
[2] Univ Rennes, Inria, IRISA, Rennes, France

TIME 2023, Athens, Greece

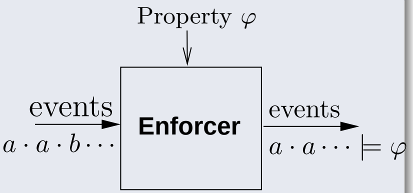
Runtime verification and enforcement

Runtime verification



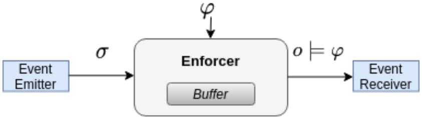
- Does σ satisfy φ ?
- Output: stream of **verdicts**.

Runtime enforcement



- Input: stream of events.
- Modified to satisfy the property.
- Output: stream of **events**.

Runtime enforcement (previous work)

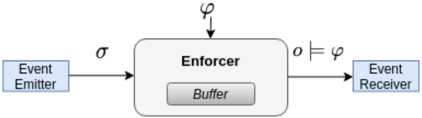


Enforcer for φ operating at runtime

An EM modifies the current execution sequence (sometimes like a "filter").

- φ: any regular property (defined as automaton).

Runtime enforcement (previous work)

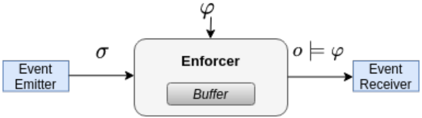


Enforcer for φ operating at runtime

An EM modifies the current execution sequence (sometimes like a "filter").

- φ: any regular property (defined as automaton).
- An enforcer behaves as a function $E : \Sigma^* \rightarrow \Sigma^*$.
  - Input ($\sigma \in \Sigma^*$): any sequence of events over Σ (Event emitter is a black-box).
  - Output ($o \in \Sigma^*$): a sequence of events such that $o \models \varphi$.

Runtime enforcement (previous work)

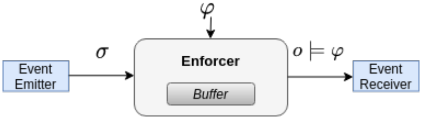


Enforcer for φ operating at runtime

An EM modifies the current execution sequence (sometimes like a "filter").

- φ: any regular property (defined as automaton).
- An enforcer behaves as a function $E : \Sigma^* \rightarrow \Sigma^*$.
  - Input ($\sigma \in \Sigma^*$): any sequence of events over Σ (Event emitter is a black-box).
  - Output ($o \in \Sigma^*$): a sequence of events such that $o \models \varphi$.
- Enforcer should satisfy soundness, transparency and monotonicity constraints.

Runtime enforcement (previous work)

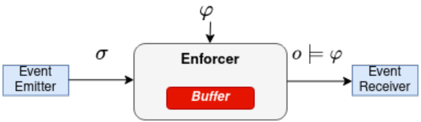


Enforcer for φ operating at runtime

An EM modifies the current execution sequence (sometimes like a "filter").

- φ: any regular property (defined as automaton).
- An enforcer behaves as a function $E : \Sigma^* \rightarrow \Sigma^*$.
  - Input ($\sigma \in \Sigma^*$): any sequence of events over Σ (Event emitter is a black-box).
  - Output ($o \in \Sigma^*$): a sequence of events such that $o \models \varphi$.
- Enforcer should satisfy soundness, transparency and monotonicity constraints.
- Enforcer augmented with a **memorization mechanism**- **Unbounded buffer**.
- What can an Enforcer do?
  - CANNOT insert events,
  - CAN block and delay events, suppress when necessary.

Motivation I : Bounded buffer



- usually [1, 2], buffer $\rightarrow$ unbounded/infinite $\rightarrow$ assumption not realistic- *Ideal* Enforcer;
- real implementation $\rightarrow$ buffer bounded
- *buffer is full?* $\rightarrow$ Lets get rid of events
  - discard the received event
  - remove some from buffer (arbitrarily)
- Problem: $\not\models$ specified property; minimal deviation from an "ideal" enforcer

Motivation II : Real-time properties

- Safety-critical systems → real-time systems
- time between events matters
- constraints on time that should elapse between (sequences of) events
- Timed properties → Timed Automata (TA)

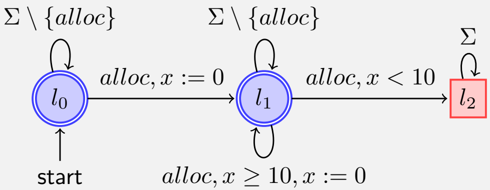


Figure: In every 10 time units (tu), there cannot be more than 1 alloc action.

Related Work

Authors	Framework	Ref.
Schneider	RE, security automata	[4]
Ligatti et al.	RE, edit automata	[5, 2]
Falcone et al.	Generic RE for untimed regular properties	[1]
Pinisetty et al.	RE for timed properties	[6, 7, 8]
Shankar et al.	RE with memory limitations (untimed)	[9]

- RE is introduced in [4] by Schneider, where security policies are specified by security automata, a variant of the Büchi automaton.
- Ligatti et al. introduced edit automata [5, 2] which not only recognise (truncate) the incorrect sequence of events but also correct those using edit functions.
- Falcone et. al [1] proposed generic enforcement monitors which are able to enforce the set of (untimed) response regular properties in the safetyprogress classification.
- Different formal RE monitor synthesis approaches have been proposed for timed properties modelled by timed automata, e.g., [6, 8, 7, 10].
- [9] introduces RE framework with bounded memory, for untimed properties

Objective (This Work)

- Given a timed property (TA) and memory constraints on the buffer
- Obtain an enforcer
  - Consider any Regular Property specified as TA
  - Removing/cleaning in an optimal way (minimal deviation from ideal enforcer, minimal dropping of events)
  - Enforcer should be: Sound, Transparent, Optimal,...

Bounded-Memory Runtime Enforcement of Timed Properties

Outline

1. Runtime Enforcement for Timed Properties with Unbounded Buffer
   Constraints · Enforcement Function · Example
2. Bounded-Memory Runtime Enforcement for Timed Properties
   Constraints · Enforcement Function · Example · Algorithm
3. Performance Analysis
4. Conclusion and Future Work

Runtime Enforcement for Timed Properties with Unbounded Buffer

Constraints · Enforcement Function · Example

Constraints:

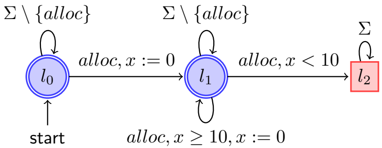
Given property $\varphi \subseteq tw(\Sigma)$, a runtime enforcer for φ is a function, $E^{\varphi} : tw(\Sigma) \rightarrow tw(\Sigma)$, satisfying the constraints:

Soundness	**(Snd)**	$\forall \sigma \in tw(\Sigma) : E^{\varphi}(\sigma) \models \varphi \vee E^{\varphi}(\sigma) = \epsilon$
Monotonicity	**(Mo)**	$\forall \sigma, \sigma' \in tw(\Sigma) : \sigma \preccurlyeq \sigma' \implies E^{\varphi}(\sigma) \preccurlyeq E^{\varphi}(\sigma')$
Transparency	**(Tr1)**	$\forall \sigma \in tw(\Sigma), \mathrm{delayable1}_{\varphi}(\sigma) = \emptyset \implies E^{\varphi}(\sigma) \triangleleft_d \sigma$
	(Tr2)	$\forall \sigma \in tw(\Sigma), \mathrm{delayable1}_{\varphi}(\sigma) \neq \emptyset \implies E^{\varphi}(\sigma) \preccurlyeq_d \sigma$

Given property $\varphi \subseteq tw(\Sigma)$, and a timed word $\sigma \in tw(\Sigma)$, function $\mathrm{delayable1}_{\varphi}(\sigma)$ returns the set of delayed words of σ, s.t. the delayed word can be extended to satisfy the property φ in the future.

Optimal Suppression	**(Opts)**	$\forall \sigma \in tw(\Sigma), \exists \sigma_s, \sigma_c \in tw(\Sigma) : \text{store}^{\varphi}(\sigma) = (\sigma_s, \sigma_c) \wedge$ $\forall (t,a) \in (\mathbb{R}_{\geq 0} \times \Sigma),\ t \geq end(\sigma_c) :$ $\text{delayable2}_{\varphi}(\sigma_s, \sigma_c \cdot (t,a)) = \emptyset$ $\implies \forall \sigma_{\text{con}} \in tw(\Sigma) : start(\sigma_{\text{con}}) \geq t,$ $E^{\varphi}(\sigma \cdot (t,a) \cdot \sigma_{\text{con}}) = E^{\varphi}(\sigma \cdot \sigma_{\text{con}})$
Optimality (min delay)	**(Opt)**	$\forall \sigma \in tw(\Sigma) : E^{\varphi}(\sigma) = \epsilon \vee \exists m, w \in tw(\Sigma) :$ $E^{\varphi}(\sigma) = m \cdot w(\models \varphi), m = max^{\varphi}_{\prec,\epsilon}(E^{\varphi}(\sigma))$ and $w = min_{\preceq,lex,end}\{w' \in m^{-1} \cdot \varphi \mid$ $\Pi_{\Sigma}(w') = \Pi_{\Sigma}(m^{-1} \cdot E^{\varphi}(\sigma)) \wedge$ $m \cdot w' \triangleleft_d \sigma \ \wedge \ start(w') \geq end(\sigma)$

Soundness: For any word $\sigma \in tw(\Sigma)$, the output produced by the enforcer (i.e., $E^{\varphi}(\sigma)$) should satisfy the property φ, as soon as it is non-empty.

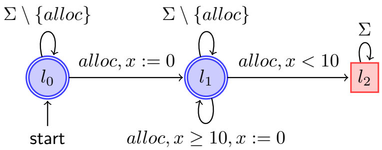


$$\sigma = (1, alloc) \cdot (2, alloc)$$

- $E^{\varphi}((1, alloc) \cdot (2, alloc)) = (1, alloc) \cdot (2, alloc)$ ×
- $E^{\varphi}((1, alloc) \cdot (2, alloc)) = (1, alloc) \cdot (12, alloc)$ ✓

Monotonicity: The output produced for the extension σ' of an input word σ (i.e., $E^{\varphi}(\sigma')$) extends the output produced for σ (i.e., $E^{\varphi}(\sigma)$).

- output is a continuously growing timed word
- what is output can only be changed by appending new events with greater dates

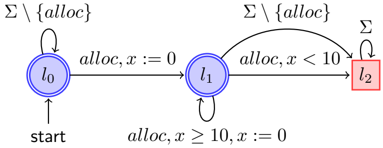


- $E^{\varphi}((1, alloc)) = (1, alloc)$
- $E^{\varphi}((1, alloc) \cdot (2, alloc)) = (1, alloc) \cdot (12, alloc)$

Transparency

- Tr_1 : no delayed word of $\sigma \in tw(\Sigma)$ exists that can satisfy the property in the future, (discard/suppress)
- Tr_2 : delayed word of $\sigma \in tw(\Sigma)$ exists that can satisfy the property in the future, then output is prefix of σ (do not discard/suppress)

Output should be a delayed prefix or a delayed subword (no new events can be inserted !!)

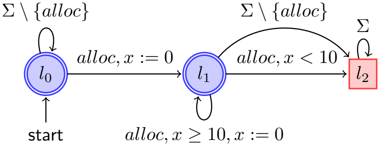


- $E^{\varphi}((1, alloc) \cdot (2, alloc) \cdot (3, req)) = (1, alloc) \cdot (12, alloc)$ (Tr)
- $E^{\varphi}((1, alloc) \cdot (2, req)) = (1, alloc) \cdot (12, alloc)$ (!Tr)

Optimal Suppression

When buffer content (σ_c) extended with a new timed event (t, a),

- no delayed word of $\sigma_c \cdot (t, a)$ exists s.t. previous output (σ_s) followed by delayed version of $\sigma_c \cdot (t, a)$ can be extended to satisfy the property φ in future then,
- SUPPRESS event a.

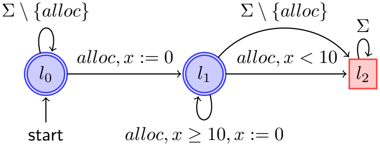


- $E^{\varphi}((1, alloc)) = (1, alloc)$

Optimal Suppression

When buffer content (σ_c) extended with a new timed event (t, a),

- no delayed word of $\sigma_c \cdot (t, a)$ exists s.t. previous output (σ_s) followed by delayed version of $\sigma_c \cdot (t, a)$ can be extended to satisfy the property φ in future then,
- SUPPRESS event a.

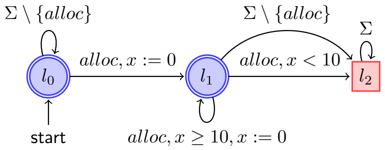


- $E^{\varphi}((1, alloc)) = (1, alloc)$
- $E^{\varphi}((1, alloc) \cdot (2, req)) = (1, alloc)$ Suppress event !!

Optimal Suppression

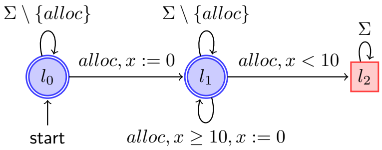
When buffer content (σ_c) extended with a new timed event (t, a),

- no delayed word of $\sigma_c \cdot (t, a)$ exists s.t. previous output (σ_s) followed by delayed version of $\sigma_c \cdot (t, a)$ can be extended to satisfy the property φ in future then,
- SUPPRESS event a.



- $E^{\varphi}((1, alloc)) = (1, alloc)$
- $E^{\varphi}((1, alloc) \cdot (2, req)) = (1, alloc)$ Suppress event !!
- $E^{\varphi}((1, alloc) \cdot (2, req) \cdot (3, alloc)) = (1, alloc) \cdot (11, alloc)$

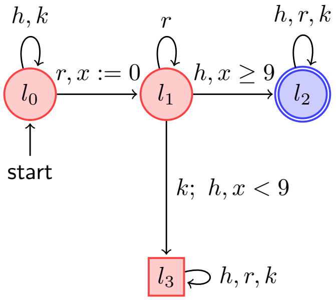
Optimality (Minimum delay):



$$\sigma = (1, alloc), (2, alloc)$$

- $E^{\varphi}((1, alloc) \cdot (2, alloc)) = (1, alloc) \cdot (11, alloc)$ ✓
- $E^{\varphi}((1, alloc) \cdot (2, alloc)) = (1, alloc) \cdot (12, alloc)$ ×
- $E^{\varphi}((1, alloc) \cdot (2, alloc)) = (1, alloc) \cdot (15, alloc)$ ×

Enforcement Function- Concept

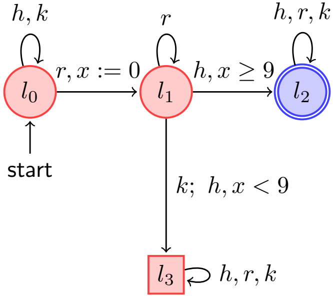


$T = 1, \sigma = (1, r) \quad \Rightarrow \quad$ *store in buffer*

Consider a new event at time $T = 2$

- $\sigma = (1, r) \cdot$ **(2,h)** $\Rightarrow$ *compute minimal delays, emit as output*

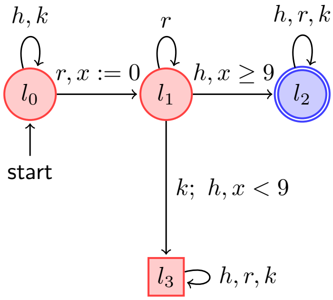
Enforcement Function- Concept



$T = 1, \sigma = (1, r) \Rightarrow$ *store in buffer*

Consider a new event at time $T = 2$

- $\sigma = (1, r) \cdot$ **(2,h)** $\Rightarrow$ *compute minimal delays, emit as output*
- $\sigma = (1, r) \cdot$ **(2,k)** $\Rightarrow$ *suppress* "k"



$T = 1, \sigma = (1, r) \Rightarrow$ *store in buffer*

Consider a new event at time $T = 2$

- $\sigma = (1, r) \cdot$ **(2,h)** $\Rightarrow$ *compute minimal delays, emit as output*
- $\sigma = (1, r) \cdot$ **(2,k)** $\Rightarrow$ *suppress* "k"
- $\sigma = (1, r) \cdot$ **(2,r)** $\Rightarrow$ *Add* "r" *to the buffer*

Enforcement Function[1]:

The enforcer for a property $\varphi \subseteq tw(\Sigma)$ is the function $E^{\varphi} : tw(\Sigma) \rightarrow tw(\Sigma)$ defined as:
$\forall \sigma \in tw(\Sigma), \forall t \in \mathbb{R}_{\geq 0}, \forall a \in \Sigma,$

$$E^{\varphi}(\sigma) = \Pi_1(\mathrm{store}^{\varphi}(\sigma)), \text{ where}$$

$\mathrm{store}^{\varphi} : tw(\Sigma) \rightarrow tw(\Sigma) \times tw(\Sigma)$ is defined as:

- $\mathrm{store}^{\varphi}(\epsilon) = (\epsilon, \epsilon)$

$$\mathrm{store}^{\varphi}(\sigma \cdot (t,a)) = \begin{cases} (\sigma_s \cdot min_{\preceq_{lex},end}(k^{\varphi}(\sigma_s, \sigma_{ca})), \epsilon) & \text{if } k^{\varphi}(\sigma_s, \sigma_{ca}) \neq \emptyset, \\ (\sigma_s, \sigma_c) & \text{if } k^{pref(\varphi)}(\sigma_s, \sigma_{ca}) = \emptyset, \\ (\sigma_s, \sigma_{ca}) & \text{otherwise.} \end{cases}$$

with:

- $(\sigma_s, \sigma_c) = \mathrm{store}^{\varphi}(\sigma)$
- $\sigma_{ca} = \sigma_c \cdot (t,a)$

- $k^{\mathcal{L}}(\sigma_1, \sigma_2)$: Given $\mathcal{L} \subseteq tw(\Sigma)$ and $\sigma_1, \sigma_2 \in tw(\Sigma)$, it computes the set of timed words w that delay σ_2, start at or after the ending date of σ_2, s.t. when σ_1 is extended with w, the word should belong to $\mathcal{L}$.

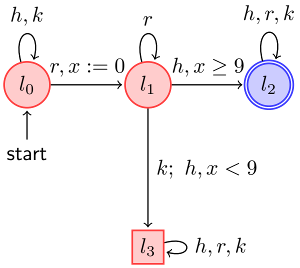
[1] It has been proved that this definition of enforcement function satisfies the constraints

Example of incremental computation by the enforcement function

Let input word $\sigma = (1,h) \cdot (2,h) \cdot (3,h) \cdot (4,h) \cdot (5,r) \cdot (6,k) \cdot (7,h)$

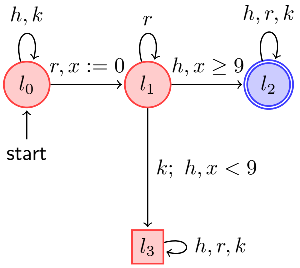
$t \in [0,1)$

$\text{obs}(\sigma, t) = \epsilon$
$\text{store}^P(\text{obs}(\sigma, t)) = (\epsilon, \epsilon)$



Example of incremental computation by the enforcement function

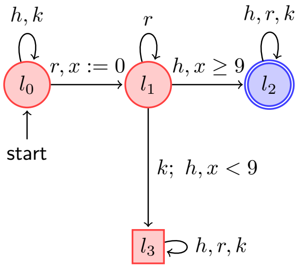
Let input word $\sigma = (1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h) \cdot (5, r) \cdot (6, k) \cdot (7, h)$



$t \in [0, 1)$	$\text{obs}(\sigma, t) = \epsilon$ $\text{store}^P(\text{obs}(\sigma, t)) = (\epsilon, \epsilon)$
$t \in [1, 2)$	$\text{obs}(\sigma, t) = (1, h)$ $\text{store}^P(\text{obs}(\sigma, t)) = (\epsilon, (1, h))$

Example of incremental computation by the enforcement function

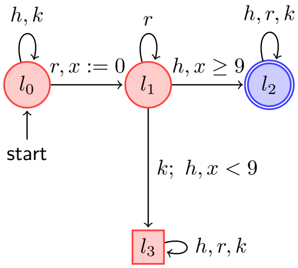
Let input word $\sigma = (1,h)\cdot(2,h)\cdot(3,h)\cdot(4,h)\cdot(5,r)\cdot(6,k)\cdot(7,h)$



$t \in [0,1)$	$\text{obs}(\sigma,t) = \epsilon$ $\text{store}^P(\text{obs}(\sigma,t)) = (\epsilon,\epsilon)$
$t \in [1,2)$	$\text{obs}(\sigma,t) = (1,h)$ $\text{store}^P(\text{obs}(\sigma,t)) = (\epsilon,(1,h))$
$t \in [2,3)$	$\text{obs}(\sigma,t) = (1,h)\cdot(2,h)$ $\text{store}^P(\text{obs}(\sigma,t)) = (\epsilon,(1,h)\cdot(2,h))$
$t \in [3,4)$	$\text{obs}(\sigma,t) = (1,h)\cdot(2,h)\cdot(3,h)$ $\text{store}^P(\text{obs}(\sigma,t)) = (\epsilon,(1,h)\cdot(2,h)\cdot(3,h))$
$t \in [4,5)$	$\text{obs}(\sigma,t) = (1,h)\cdot(2,h)\cdot(3,h)\cdot(4,h)$ $\text{store}^P(\text{obs}(\sigma,t)) = (\epsilon,(1,h)\cdot(2,h)\cdot(3,h)\cdot(4,h))$
$t \in [5,6)$	$\text{obs}(\sigma,t) = (1,h)\cdot(2,h)\cdot(3,h)\cdot(4,h)\cdot(5,r)$ $\text{store}^P(\text{obs}(\sigma,t)) = (\epsilon,(1,h)\cdot(2,h)\cdot(3,h)\cdot(4,h)\cdot(5,r))$

Example of incremental computation by the enforcement function

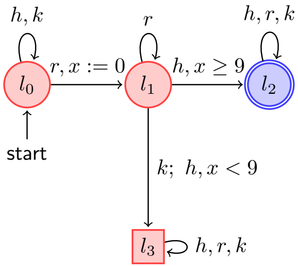
Let input word $\sigma = (1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h) \cdot (5, r) \cdot (6, k) \cdot (7, h)$



$t \in [0, 1)$	$\text{obs}(\sigma, t) = \epsilon$ $\text{store}^P(\text{obs}(\sigma, t)) = (\epsilon, \epsilon)$
$t \in [1, 2)$	$\text{obs}(\sigma, t) = (1, h)$ $\text{store}^P(\text{obs}(\sigma, t)) = (\epsilon, (1, h))$
$t \in [2, 3)$	$\text{obs}(\sigma, t) = (1, h) \cdot (2, h)$ $\text{store}^P(\text{obs}(\sigma, t)) = (\epsilon, (1, h) \cdot (2, h))$
$t \in [3, 4)$	$\text{obs}(\sigma, t) = (1, h) \cdot (2, h) \cdot (3, h)$ $\text{store}^P(\text{obs}(\sigma, t)) = (\epsilon, (1, h) \cdot (2, h) \cdot (3, h))$
$t \in [4, 5)$	$\text{obs}(\sigma, t) = (1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h)$ $\text{store}^P(\text{obs}(\sigma, t)) = (\epsilon, (1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h))$
$t \in [5, 6)$	$\text{obs}(\sigma, t) = (1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h) \cdot (5, r)$ $\text{store}^P(\text{obs}(\sigma, t)) = (\epsilon, (1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h) \cdot (5, r))$
$t \in [6, 7)$	$\text{obs}(\sigma, t) = (1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h) \cdot (5, r) \cdot (6, k)$ $\text{store}^P(\text{obs}(\sigma, t)) = (\epsilon, (1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h) \cdot (5, r))$

Example of incremental computation by the enforcement function

Let input word $\sigma = (1,h) \cdot (2,h) \cdot (3,h) \cdot (4,h) \cdot (5,r) \cdot (6,k) \cdot (7,h)$



$t \in [0,1)$	$\text{obs}(\sigma,t) = \epsilon$ $\text{store}^P(\text{obs}(\sigma,t)) = (\epsilon, \epsilon)$
$t \in [1,2)$	$\text{obs}(\sigma,t) = (1,h)$ $\text{store}^P(\text{obs}(\sigma,t)) = (\epsilon, (1,h))$
$t \in [2,3)$	$\text{obs}(\sigma,t) = (1,h) \cdot (2,h)$ $\text{store}^P(\text{obs}(\sigma,t)) = (\epsilon, (1,h) \cdot (2,h))$
$t \in [3,4)$	$\text{obs}(\sigma,t) = (1,h) \cdot (2,h) \cdot (3,h)$ $\text{store}^P(\text{obs}(\sigma,t)) = (\epsilon, (1,h) \cdot (2,h) \cdot (3,h))$
$t \in [4,5)$	$\text{obs}(\sigma,t) = (1,h) \cdot (2,h) \cdot (3,h) \cdot (4,h)$ $\text{store}^P(\text{obs}(\sigma,t)) = (\epsilon, (1,h) \cdot (2,h) \cdot (3,h) \cdot (4,h))$
$t \in [5,6)$	$\text{obs}(\sigma,t) = (1,h) \cdot (2,h) \cdot (3,h) \cdot (4,h) \cdot (5,r)$ $\text{store}^P(\text{obs}(\sigma,t)) = (\epsilon, (1,h) \cdot (2,h) \cdot (3,h) \cdot (4,h) \cdot (5,r))$
$t \in [6,7)$	$\text{obs}(\sigma,t) = (1,h) \cdot (2,h) \cdot (3,h) \cdot (4,h) \cdot (5,r) \cdot (6,k)$ $\text{store}^P(\text{obs}(\sigma,t)) = (\epsilon, (1,h) \cdot (2,h) \cdot (3,h) \cdot (4,h) \cdot (5,r))$
$t \in [7,\infty)$	$\text{obs}(\sigma,t) = (1,h) \cdot (2,h) \cdot (3,h) \cdot (4,h) \cdot (5,r) \cdot (6,k) \cdot (7,h)$ $\text{store}^P(\text{obs}(\sigma,t)) = ((7,h) \cdot (7,h) \cdot (7,h) \cdot (7,h) \cdot (7,r) \cdot (16,h), \epsilon)$

Bounded-Memory Runtime Enforcement for Timed Properties

Constraints · Enforcement Function · Example · Algorithm

Constraints:

A bounded enforcer for a timed property $\varphi \subseteq tw(\Sigma)$, equipped with a buffer of size k, is a function $E^{\varphi,k}$

$$E^{\varphi,k} : tw(\Sigma) \to tw(\Sigma) \times \{\bot, \top, stop\}$$

satisfying the following constraints

Soundness	**(SndB)**	$\forall \sigma \in tw(\Sigma) : E^{\varphi,k}_{out}(\sigma) \models \varphi \vee E^{\varphi,k}_{out}(\sigma) = \epsilon$
Monotonicity	**(Mo1B)**	$\forall \sigma, \sigma' \in tw(\Sigma) : \sigma \preccurlyeq \sigma' \implies E^{\varphi,k}_{out}(\sigma) \preccurlyeq E^{\varphi,k}_{out}(\sigma')$
	(Mo2B)	$\forall \sigma, \sigma' \in tw(\Sigma) : \sigma \preccurlyeq \sigma', (E^{\varphi,k}_{mode}(\sigma) = \bot \implies E^{\varphi,k}_{mode}(\sigma') = \bot)$
Transparency	**(Tr1B)**	$\forall \sigma \in tw(\Sigma), \text{delayable1}_{\varphi}(\sigma) = \emptyset \vee E^{\varphi,k}_{\text{mode}}(\sigma) = \bot \implies E^{\varphi,k}_{out}(\sigma) \lhd_d \sigma$
	(Tr2B)	$\forall \sigma \in tw(\Sigma), \text{delayable1}_{\varphi}(\sigma) \neq \emptyset \wedge E^{\varphi,k}_{\text{mode}}(\sigma) = \top \implies E^{\varphi,k}_{out}(\sigma) \preccurlyeq_d \sigma$

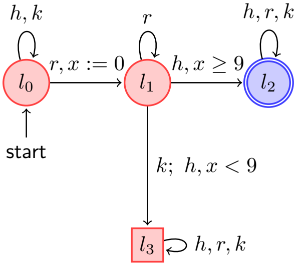
Optimal Suppression	**(Opts)**	$\forall \sigma \in tw(\Sigma), \exists \sigma_s, \sigma_c \in tw(\Sigma) : \mathrm{store}^{\varphi,k}(\sigma) = (\sigma_s, \sigma_c),$ $E^{\varphi,k}_{mode}(\sigma) = \top, \forall (t,a) \in (\mathbb{R}_{\geq 0} \times \Sigma),\ t \geq end(\sigma_c) :$ $\mathrm{delayable2}_{\varphi}(\sigma_s, \sigma_c \cdot (t,a)) = \emptyset \implies$ $\forall \sigma_{\mathrm{con}} \in tw(\Sigma) : start(\sigma_{\mathrm{con}}) \geq t,$ $E^{\varphi,k}(\sigma \cdot (t,a) \cdot \sigma_{\mathrm{con}}) = E^{\varphi,k}(\sigma \cdot \sigma_{\mathrm{con}}) \wedge$ $E^{\varphi,k}_{mode}(\sigma \cdot (t,a) \cdot \sigma_{\mathrm{con}}) = \bot$
Optimality (min delay)	**(Opt)**	$\forall \sigma \in tw(\Sigma) : E^{\varphi,k}_{\mathrm{out}}(\sigma) = \epsilon \vee \exists m, w \in tw(\Sigma) :$ $E^{\varphi,k}_{\mathrm{out}}(\sigma) = m \cdot w(\models \varphi), m = max^{\varphi}_{\prec,\epsilon}(E^{\varphi,k}(\sigma))$ and $w = min_{\preceq,lex,end}\{w' \in m^{-1} \cdot \varphi \mid$ $\Pi_{\Sigma}(w') = \Pi_{\Sigma}(m^{-1} \cdot E^{\varphi,k}(\sigma)) \wedge$ $m \cdot w' \triangleleft_d \sigma \quad \wedge \quad start(w') \geq end(\sigma)$
Optimality (max length)	**(Opt)**	$(E^{\varphi,k}_{\mathrm{out}}(\sigma) \cdot \mathrm{buff}(E^{\varphi,k}(\sigma) = F^{\varphi,k}_{\mathrm{out}}(\sigma) \cdot \mathrm{buff}(F^{\varphi,k}(\sigma))$ $\wedge \lvert E^{\varphi,k}_{\mathrm{out}}(\sigma \cdot (t,a)) \cdot \mathrm{buff}(E^{\varphi,k}(\sigma \cdot (t,a))) \rvert <$ $\lvert F^{\varphi,k}_{\mathrm{out}}(\sigma \cdot (t,a)) \cdot \mathrm{buff}(F^{\varphi,k}(\sigma \cdot (t,a)) \rvert)$ $\implies \neg(\infty\text{-compatible}(F^{\varphi,k}))$

Equivalence of two timed words

- Given:
  - $\sigma = (t_1, a_1) \cdot (t_2, a_2) \cdots (t_n, a_n)$
  - $\sigma' = (t'_1, a'_1) \cdot (t'_2, a'_2) \cdots (t'_m, a'_m)$
  - $\mathcal{A}_\varphi$
- $\sigma \sim_\varphi \sigma'$, if runs ρ and ρ' from q_0 of $\mathcal{A}_\varphi$
  - $\rho = q_0 \xrightarrow{(\delta_1, a_1)} q_1 \cdots q_{n-1} \xrightarrow{(\delta_n, a_n)} q_n$
  - $\rho' = q_0 \xrightarrow{(\delta'_1, a'_1)} q'_1 \cdots q'_{m-1} \xrightarrow{(\delta'_m, a'_m)} q_m$

end on q_n and q_m respectively s.t. $\mathcal{L}(\mathcal{A}_\varphi, q_n) = \mathcal{L}(\mathcal{A}_\varphi, q_m)$.

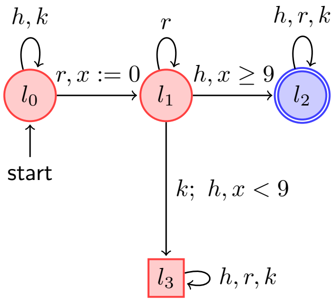
Optimality (max length): For any input σ, the output is of maximum length and is φ-equivalent to one produced by an ideal enforcer.



k=4, $\sigma = (1,h) \cdot (2,h) \cdot (3,h) \cdot (4,h) \cdot (5,r) \cdot (6,h)$

- $E^{\varphi}(\underline{(1,h)} \cdot (2,h) \cdot (3,h) \cdot (4,h) \cdot (5,r) \cdot (6,h))$= (6, h)·(6,h)·(6, h)·(6, r)·(15,h) ✓
- $E^{\varphi}(\underline{(1,h) \cdot (2,h)} \cdot (3,h) \cdot (4,h) \cdot (5,r) \cdot (6,h))$ =(6,h)·(6, h)· (6, r)·(15,h) ×

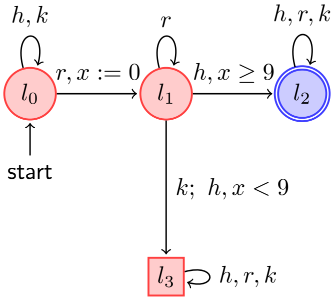
Fundamental Concept- Bounded-Memory RE



$T = 1, \sigma = (1, r) \Rightarrow$ *store in buffer*

Consider a new event at time $T = 2$

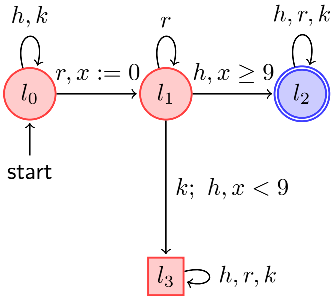
- $\sigma = (1, r) \cdot (2, h) \Rightarrow$ *compute minimal delays, emit as output*
- $\sigma = (1, r) \cdot (2, k) \Rightarrow$ *suppress* "k"



$T = 4$, $\sigma = (1, h) \cdot (2, h) \cdot (3, r) \cdot (4, r) \Rightarrow$ *store in buffer*

Consider a new event at time $T = 5$

- $\sigma = (1, h) \cdot (2, h) \cdot (3, r) \cdot (4, r) \cdot$ **(5,r)** & bufferFull=F $\Rightarrow$ *Add "r" to the buffer*



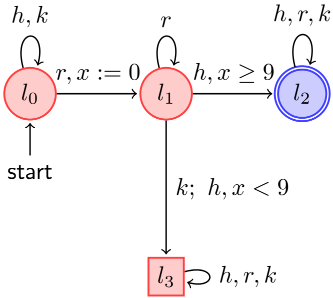
$T = 4$, $\sigma = (1, h) \cdot (2, h) \cdot (3, r) \cdot (4, r) \Rightarrow$ *store in buffer*

Consider a new event at time $T = 5$

- $\sigma = (1, h) \cdot (2, h) \cdot (3, r) \cdot (4, r) \cdot$ **(5,r)** & bufferFull=F $\Rightarrow$ *Add "r" to the buffer*
- $\sigma = (1, h) \cdot (2, h) \cdot (3, r) \cdot (4, r) \cdot$ **(5,r)** & bufferFull=T $\Rightarrow$
  - $delayed \sim_\varphi subword = (5, h) \cdot (5, r) \cdot (5, r) \cdot (5, r),$
    $(5, h) \cdot (5, h) \cdot (5, r) \cdot (5, r)$
  - clean the buffer
  - $\sigma_c = (5, h) \cdot (5, r) \cdot (5, r) \cdot (5, r)$

$T = 4$, $\sigma = (1, h) \cdot (2, h) \cdot (3, r) \cdot (4, r)$ $\Rightarrow$ *store in buffer*

Consider a new event at time $T = 5$



- $\sigma = (1, h) \cdot (2, h) \cdot (3, r) \cdot (4, r) \cdot$ **(5,r)** & bufferFull=F $\Rightarrow$ *Add "r" to the buffer*
- $\sigma = (1, h) \cdot (2, h) \cdot (3, r) \cdot (4, r) \cdot$ **(5,r)** & bufferFull=T $\Rightarrow$
  - $delayed \sim_\varphi subword = (5, h) \cdot (5, r) \cdot (5, r) \cdot (5, r)$, $(5, h) \cdot (5, h) \cdot (5, r) \cdot (5, r)$
  - clean the buffer
  - $\sigma_c = (5, h) \cdot (5, r) \cdot (5, r) \cdot (5, r)$
  - $delayed \sim_\varphi subword = \emptyset$ (stop the enforcer)

Minimal sequence of events in the buffer engaged in a loop that is most obsolute..

Bounded Enforcement Function $E^{\varphi,k}$:

A bounded enforcer for a property $\varphi \subseteq tw(\Sigma)$ is the function $E^{\varphi,k} : tw(\Sigma) \rightarrow tw(\Sigma) \times \{\top, \bot, stop\}$, and is defined as:
$\forall \sigma \in tw(\Sigma), \forall t \in \mathbb{R}_{\geq 0}, \forall a \in \Sigma,$

$$E^{\varphi,k}(\sigma) = (\Pi_1(\mathrm{store}^{\varphi,k}(\sigma)), \Pi_3(\mathrm{store}^{\varphi,k}(\sigma))),$$ where:

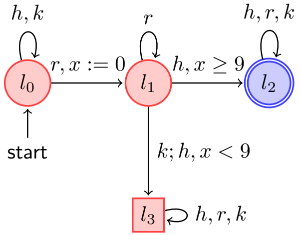
$\mathrm{store}^{\varphi,k} : tw(\Sigma) \rightarrow tw(\Sigma) \times tw(\Sigma) \times \{\top, \bot, stop\}$ is defined as:

- $\mathrm{store}^{\varphi,k}(\epsilon) = (\epsilon, \epsilon, \top)$
- $\mathrm{store}^{\varphi,k}(\sigma \cdot (t, a)) =$

$$\begin{cases} (\sigma_s \cdot min_{\preceq_{lex},end}(k^{\varphi}(\sigma_s, \sigma_{ca})), \epsilon, \{\top, \bot\}) & \text{if } k^{\varphi}(\sigma_s, \sigma_{ca}) \neq \emptyset, \\ (\sigma_s, \sigma_c, \bot) & \text{if } k^{pref(\varphi)}(\sigma_s, \sigma_{ca}) = \emptyset, \\ (\sigma_s, \sigma_{ca}, \{\top, \bot\}) & \text{if } k^{pref(\varphi)}(\sigma_s, \sigma_{ca}) \neq \emptyset \ \wedge |\sigma_{ca}| \leq k \\ (\sigma_s, \sigma_c, stop) & \text{if } k^{pref(\varphi)}(\sigma_s, \sigma_{ca}) \neq \emptyset \ \wedge \ |\sigma_{ca}| > k \\ & \quad \wedge \ \mathrm{Get\_SW}^{\varphi,k}(\sigma_s, \sigma_{ca}) = \emptyset \\ (\sigma_s, \mathrm{Clean}^{\varphi,k}(\sigma_s, \sigma_{ca}), \bot) & \text{if } k^{pref(\varphi)}(\sigma_s, \sigma_{ca}) \neq \emptyset \ \wedge |\sigma_{ca}| > k \\ & \quad \wedge \ \mathrm{Get\_SW}^{\varphi,k}(\sigma_s, \sigma_{ca}) \neq \emptyset \end{cases}$$

Example of incremental computation by enforcement function with $k = 4$

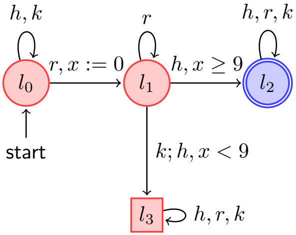
Let input word $\sigma = (1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h) \cdot (5, r) \cdot (6, k) \cdot (7, h)$



$t \in [0, 1)$	$\text{obs}(\sigma, t) = \epsilon$ $\text{store}^{P,4}(\text{obs}(\sigma, t)) = (\epsilon, \epsilon)$
$t \in [1, 2)$	$\text{obs}(\sigma, t) = (1, h)$ $\text{store}^{P,4}(\text{obs}(\sigma, t)) = (\epsilon, (1, h))$
$t \in [2, 3)$	$\text{obs}(\sigma, t) = (1, h) \cdot (2, h)$ $\text{store}^{P,4}(\text{obs}(\sigma, t)) = (\epsilon, (1, h) \cdot (2, h))$
$t \in [3, 4)$	$\text{obs}(\sigma, t) = (1, h) \cdot (2, h) \cdot (3, h)$ $\text{store}^{P,4}(\text{obs}(\sigma, t)) = (\epsilon, (1, h) \cdot (2, h) \cdot (3, h))$
$t \in [4, 5)$	$\text{obs}(\sigma, t) = (1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h)$ $\text{store}^{P,4}(\text{obs}(\sigma, t)) = (\epsilon, (1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h))$

Example of incremental computation by enforcement function with $k = 4$

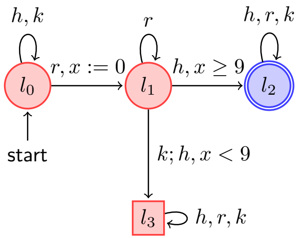
Let input word $\sigma = (1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h) \cdot (5, r) \cdot (6, k) \cdot (7, h)$



$t \in [0, 1)$	$\text{obs}(\sigma, t) = \epsilon$ $\text{store}^{P,4}(\text{obs}(\sigma, t)) = (\epsilon, \epsilon)$
$t \in [1, 2)$	$\text{obs}(\sigma, t) = (1, h)$ $\text{store}^{P,4}(\text{obs}(\sigma, t)) = (\epsilon, (1, h))$
$t \in [2, 3)$	$\text{obs}(\sigma, t) = (1, h) \cdot (2, h)$ $\text{store}^{P,4}(\text{obs}(\sigma, t)) = (\epsilon, (1, h) \cdot (2, h))$
$t \in [3, 4)$	$\text{obs}(\sigma, t) = (1, h) \cdot (2, h) \cdot (3, h)$ $\text{store}^{P,4}(\text{obs}(\sigma, t)) = (\epsilon, (1, h) \cdot (2, h) \cdot (3, h))$
$t \in [4, 5)$	$\text{obs}(\sigma, t) = (1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h)$ $\text{store}^{P,4}(\text{obs}(\sigma, t)) = (\epsilon, (1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h))$
$t \in [5, 6)$	$\text{obs}(\sigma, t) = (1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h) \cdot (5, r)$ $\text{store}^{P,4}(\text{obs}(\sigma, t)) = (\epsilon, (5, h) \cdot (5, h) \cdot (5, h) \cdot (5, r))$

Example of incremental computation by enforcement function with $k = 4$

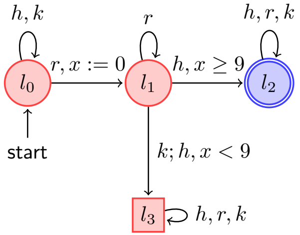
Let input word $\sigma = (1,h)\cdot(2,h)\cdot(3,h)\cdot(4,h)\cdot(5,r)\cdot(6,k)\cdot(7,h)$



$t \in [0,1)$	$\mathrm{obs}(\sigma,t) = \epsilon$ $\mathrm{store}^{P,4}(\mathrm{obs}(\sigma,t)) = (\epsilon,\epsilon)$
$t \in [1,2)$	$\mathrm{obs}(\sigma,t) = (1,h)$ $\mathrm{store}^{P,4}(\mathrm{obs}(\sigma,t)) = (\epsilon,(1,h))$
$t \in [2,3)$	$\mathrm{obs}(\sigma,t) = (1,h)\cdot(2,h)$ $\mathrm{store}^{P,4}(\mathrm{obs}(\sigma,t)) = (\epsilon,(1,h)\cdot(2,h))$
$t \in [3,4)$	$\mathrm{obs}(\sigma,t) = (1,h)\cdot(2,h)\cdot(3,h)$ $\mathrm{store}^{P,4}(\mathrm{obs}(\sigma,t)) = (\epsilon,(1,h)\cdot(2,h)\cdot(3,h))$
$t \in [4,5)$	$\mathrm{obs}(\sigma,t) = (1,h)\cdot(2,h)\cdot(3,h)\cdot(4,h)$ $\mathrm{store}^{P,4}(\mathrm{obs}(\sigma,t)) = (\epsilon,(1,h)\cdot(2,h)\cdot(3,h)\cdot(4,h))$
$t \in [5,6)$	$\mathrm{obs}(\sigma,t) = (1,h)\cdot(2,h)\cdot(3,h)\cdot(4,h)\cdot(5,r)$ $\mathrm{store}^{P,4}(\mathrm{obs}(\sigma,t)) = (\epsilon,(5,h)\cdot(5,h)\cdot(5,h)\cdot(5,r))$
$t \in [6,7)$	$\mathrm{obs}(\sigma,t) = (1,h)\cdot(2,h)\cdot(3,h)\cdot(4,h)\cdot(5,r)\cdot(6,k)$ $\mathrm{store}^{P,4}(\mathrm{obs}(\sigma,t)) = (\epsilon,(5,h)\cdot(5,h)\cdot(5,h)\cdot(5,r))$

Example of incremental computation by enforcement function with $k = 4$

Let input word $\sigma = (1,h)\cdot(2,h)\cdot(3,h)\cdot(4,h)\cdot(5,r)\cdot(6,k)\cdot(7,h)$



$t \in [0,1)$	$\mathrm{obs}(\sigma,t) = \epsilon$ $\mathrm{store}^{P,4}(\mathrm{obs}(\sigma,t)) = (\epsilon,\epsilon)$
$t \in [1,2)$	$\mathrm{obs}(\sigma,t) = (1,h)$ $\mathrm{store}^{P,4}(\mathrm{obs}(\sigma,t)) = (\epsilon,(1,h))$
$t \in [2,3)$	$\mathrm{obs}(\sigma,t) = (1,h)\cdot(2,h)$ $\mathrm{store}^{P,4}(\mathrm{obs}(\sigma,t)) = (\epsilon,(1,h)\cdot(2,h))$
$t \in [3,4)$	$\mathrm{obs}(\sigma,t) = (1,h)\cdot(2,h)\cdot(3,h)$ $\mathrm{store}^{P,4}(\mathrm{obs}(\sigma,t)) = (\epsilon,(1,h)\cdot(2,h)\cdot(3,h))$
$t \in [4,5)$	$\mathrm{obs}(\sigma,t) = (1,h)\cdot(2,h)\cdot(3,h)\cdot(4,h)$ $\mathrm{store}^{P,4}(\mathrm{obs}(\sigma,t)) = (\epsilon,(1,h)\cdot(2,h)\cdot(3,h)\cdot(4,h))$
$t \in [5,6)$	$\mathrm{obs}(\sigma,t) = (1,h)\cdot(2,h)\cdot(3,h)\cdot(4,h)\cdot(5,r)$ $\mathrm{store}^{P,4}(\mathrm{obs}(\sigma,t)) = (\epsilon,(5,h)\cdot(5,h)\cdot(5,h)\cdot(5,r))$
$t \in [6,7)$	$\mathrm{obs}(\sigma,t) = (1,h)\cdot(2,h)\cdot(3,h)\cdot(4,h)\cdot(5,r)\cdot(6,k)$ $\mathrm{store}^{P,4}(\mathrm{obs}(\sigma,t)) = (\epsilon,(5,h)\cdot(5,h)\cdot(5,h)\cdot(5,r))$
$t \in [7,\infty)$	$\mathrm{obs}(\sigma,t) = (1,h)\cdot(2,h)\cdot(3,h)\cdot(4,h)\cdot(5,r)\cdot(6,k)\cdot(7,h)$ $\mathrm{store}^{P,4}(\mathrm{obs}(\sigma,t)) = ((7,h)\cdot(7,h)\cdot(7,h)\cdot(7,r)\cdot(16,h),\epsilon)$

Algorithm Enforcer $(\mathcal{A}_\varphi, k)$

$\sigma_c, \sigma_s = [\,]$
$currState \leftarrow [l_0, 0]$
$c = 0$
while true **do**
  $(\delta, a) \leftarrow$ `await_event()`
  if $c \neq 0$ **then**
    $\delta = \delta + c$
  $currState[1] = currState[1] + \delta$
  $allPaths =$ `check_reachability`$(currState, \sigma_c \cdot (\delta, a))$
  $accPaths =$ `get_acc_paths`$(allPaths)$
  if $accPaths \neq \emptyset$ **then**
    $(\sigma_{ca}, state) =$ `get_od`$(currState, \sigma_c \cdot (\delta, a))$
    for $event \in \sigma_{ca}$ **do**
      append$(\sigma_s, \sigma_{ca}[event])$
    $\sigma_c = [\,]$
    $currState = state$
    `release`(σ_s)

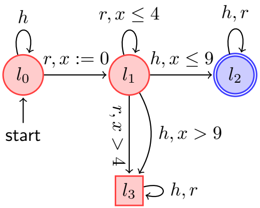
if $accPaths = \emptyset$ **then**
  $isReachable =$ `check_reach_acc`$(allPaths)$
  if $isReachable == False$ **then**
    continue
  else
    if `len`$(\sigma_c) < k$ **then**
      `append`$(\sigma_c, (\delta, a))$
    else
      if `Get_SW`$(\sigma_s, \sigma_{ca}) \neq \emptyset$ **then**
        $(\sigma_c, c) =$ `clean`$(currState, \sigma_c \cdot (\delta, a))$
      else
        exit

Performance Analysis

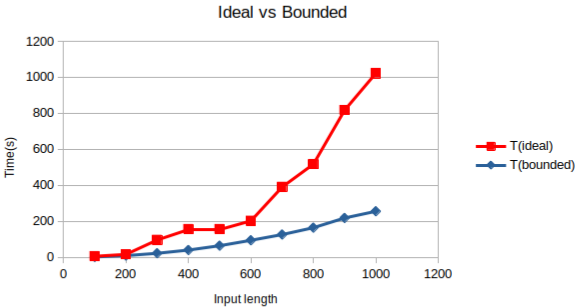
Performance analysis

- Implemented algorithm and developed an experimentation framework in Python
  - To validate the feasibility of proposed enforcement monitoring
  - To analyse the performance of the enforcer through experiments
- TiPEX [11]: RE monitor generation tool

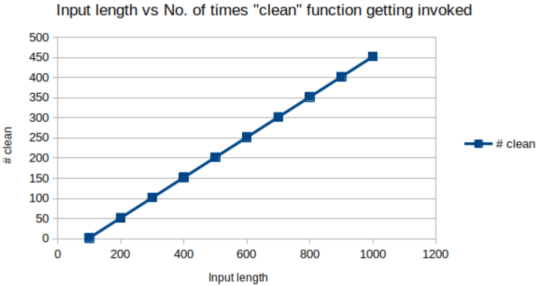
With k=50



h
r, x ≤ 4
h, r
r, x := 0
h, x ≤ 9
l_0
l_1
l_2
start
r, x > 4
h, x > 9
l_3
h, r

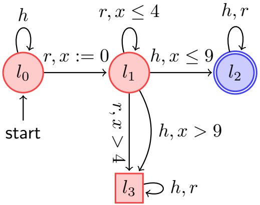


Ideal vs Bounded
Time(s)
Input length
T(ideal)
T(bounded)



Input length vs No. of times "clean" function getting invoked
clean
Input length

With k=50



- Time taken by the bounded-memory enforcer increases non-linearly with the linear increase in trace length.
- Average time taken by the function clean (per call) is 0.019 s and is low/reasonable.

The behaviour is non-linear because of reasons such as, overhead in maintaining the list of uncorrected events, the more numbers of times function clean is called, etc.

Conclusion and Future Work

Conclusion and Future Work

- Presents a novel framework for enforcing timed properties with memory constraints on the enforcer
- Enforcer (delay or suppress events) $\rightarrow$ property violations or buffer overflows
- Constraints $\rightarrow$ how a bounded enforcer transforms words
  - Additional constraints to provide some guarantees on the output sequence produced by the enforcer in terms of delay, length
- Functional definition & algorithmic version
- Proofs $\rightarrow$ functional definition $\models$ constraints
- Implementation, performance evaluation
- Additionally, syntactic conditions $\models$ TAs $\rightarrow$ enforcer never halts

Future works include developing other alternative implementations of the proposed enforcement framework using other TA frameworks such as TChecker[2] (to obtain the zone graphs and for reachability analysis).

[2] TChecker https://www.labri.fr/perso/herbrete/tchecker/

References I

- Y. Falcone, L. Mounier, J. Fernandez, and J. Richier, "Runtime enforcement monitors: composition, synthesis, and enforcement abilities," Formal Methods Syst. Des., vol. 38, no. 3, pp. 223–262, 2011.
- J. Ligatti, L. Bauer, and D. Walker, "Run-time enforcement of nonsafety policies," ACM Trans. Inf. Syst. Secur., vol. 12, Jan. 2009.
- R. Alur and D. L. Dill, "A theory of timed automata," Theoretical Computer Science, vol. 126, no. 2, pp. 183–235, 1994.
- F. B. Schneider, "Enforceable security policies," ACM Trans. Inf. Syst. Secur., vol. 3, pp. 30–50, Feb. 2000.
- J. Ligatti, L. Bauer, and D. Walker, "Edit automata: enforcement mechanisms for run-time security policies," Int. J. Inf. Sec., vol. 4, no. 1-2, pp. 2–16, 2005.

References II

- S. Pinisetty, Y. Falcone, T. Jéron, H. Marchand, A. Rollet, and O. L. Nguena-Timo, "Runtime enforcement of timed properties," in Runtime Verification, Third International Conference, RV 2012, Istanbul, Turkey, September 25-28, 2012, Revised Selected Papers (S. Qadeer and S. Tasiran, eds.), vol. 7687 of Lecture Notes in Computer Science, pp. 229–244, Springer, 2012.
- S. Pinisetty, Y. Falcone, T. Jéron, H. Marchand, A. Rollet, and O. Nguena-Timo, "Runtime enforcement of timed properties revisited," Formal Methods in System Design, vol. 45, no. 3, pp. 381–422, 2014.
- Y. Falcone, T. Jéron, H. Marchand, and S. Pinisetty, "Runtime enforcement of regular timed properties by suppressing and delaying events," Systems & Control Letters, vol. 123, pp. 2–41, 2016.

- S. Shankar, A. Rollet, S. Pinisetty, and Y. Falcone, "Bounded-memory runtime enforcement," in Model Checking Software (O. Legunsen and G. Rosu, eds.), (Cham), pp. 114–133, Springer International Publishing, 2022.
- M. Renard, Y. Falcone, A. Rollet, S. Pinisetty, T. Jéron, and H. Marchand, "Enforcement of (timed) properties with uncontrollable events," in Theoretical Aspects of Computing - ICTAC 2015 - 12th International Colloquium Cali, Colombia, October 29-31, 2015, Proceedings, pp. 542–560, 2015.
- S. Pinisetty, Y. Falcone, T. Jéron, and H. Marchand, "Tipex: A tool chain for timed property enforcement during execution," in Runtime Verification (E. Bartocci and R. Majumdar, eds.), (Cham), pp. 306–320, Springer International Publishing, 2015.

Acknowledgment

This work has been partially supported by The Ministry of Human Resource Development, Government of India (SPARC P#701), IIT Bhubaneswar Seed Grant (SP093)

Thank you.