Scaling Up Proactive Enforcement

François Hublet
francois.hublet@inf.ethz.ch

Leonardo Lima
leonardo@di.ku.dk

David Basin
basin@inf.ethz.ch

Srđan Krstić
srdan.krstic@inf.ethz.ch

Dmitriy Traytel
traytel@di.ku.dk

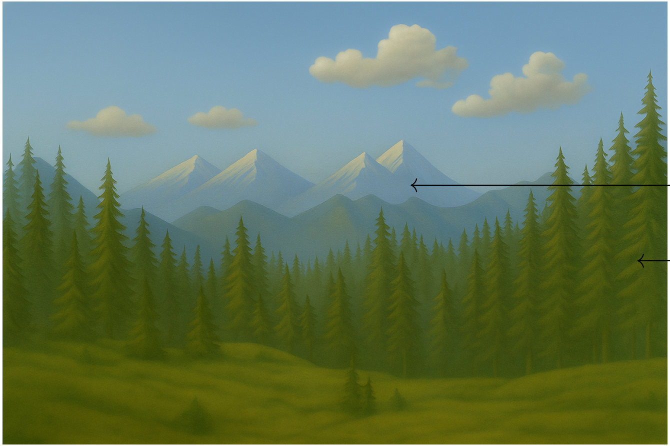
International Conference on Computer-Aided Verification — Zagreb, July 23, 2025

ETH zürich

The Fairytale of Complex Systems

Forest of software

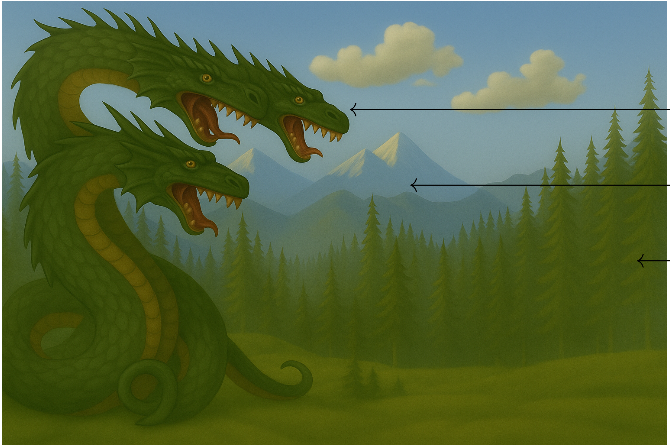
The Fairytale of Complex Systems



Mountains of hardware

Forest of software

The Fairytale of Complex Systems

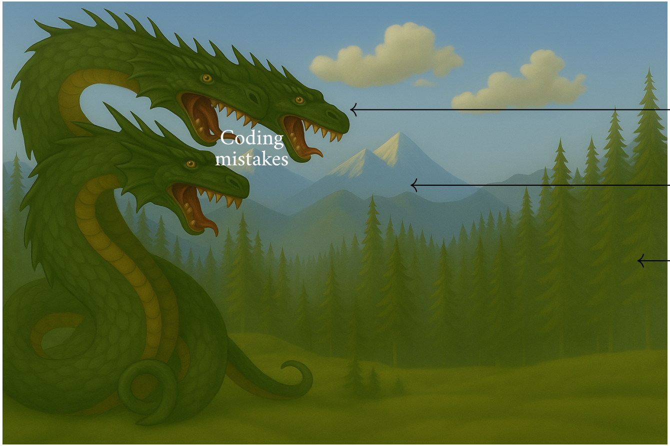


Hydra of non-compliance

Mountains of hardware

Forest of software

The Fairytale of Complex Systems

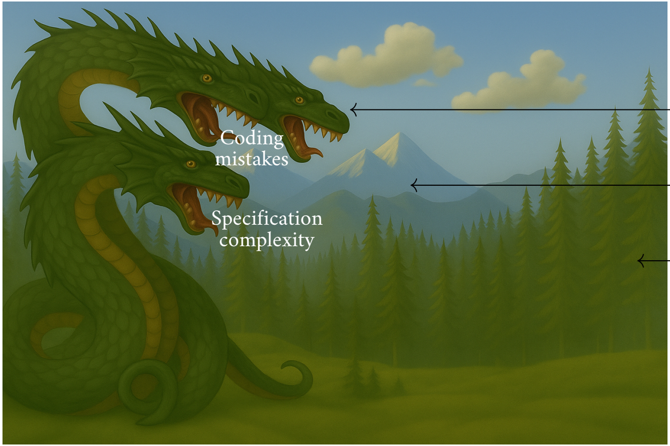


Hydra of non-compliance

Mountains of hardware

Forest of software

The Fairytale of Complex Systems

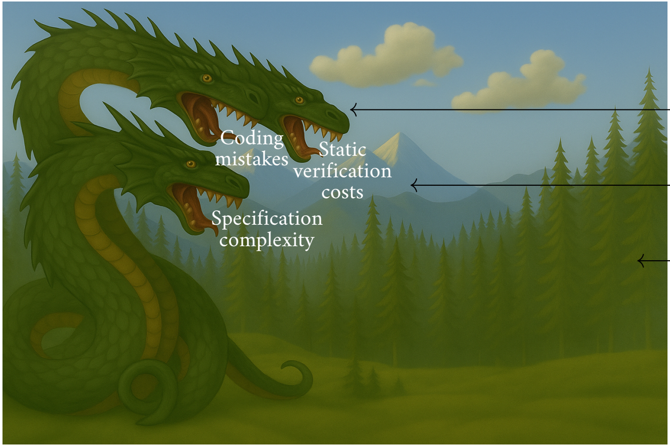


Hydra of non-compliance

Mountains of hardware

Forest of software

The Fairytale of Complex Systems



Hydra of non-compliance

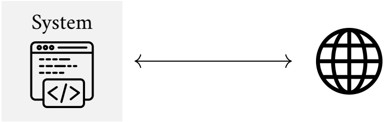
Mountains of hardware

Forest of software

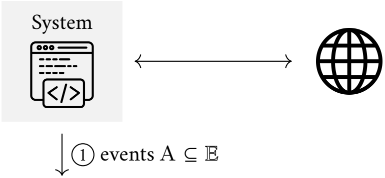
The Fairytale of Complex Systems

Our new
runtime enforcement tool

Runtime Enforcement

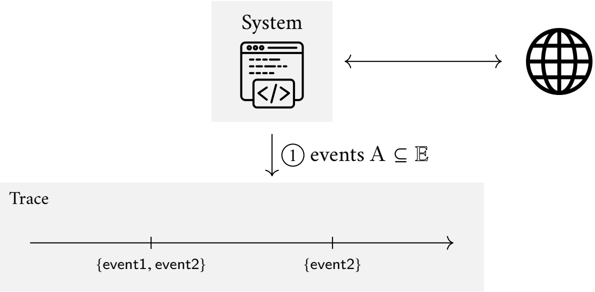


Runtime Enforcement

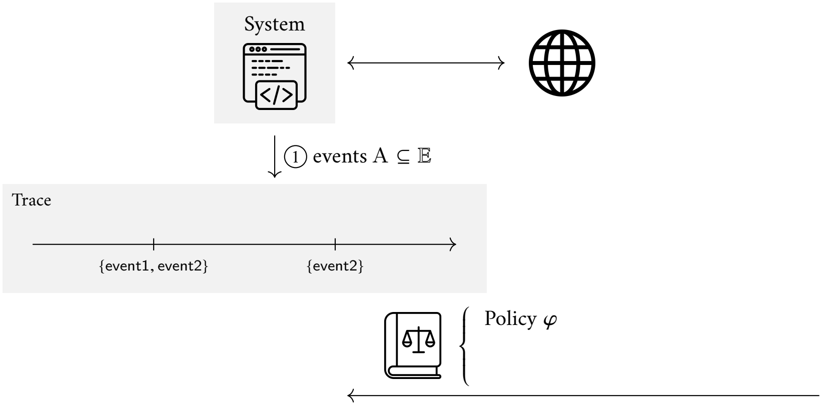


$\textcircled{1}$ events $A \subseteq \mathbb{E}$

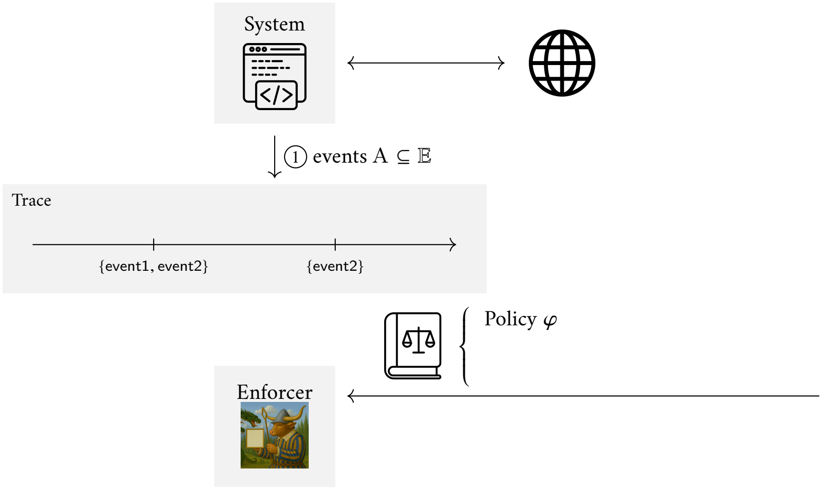
Runtime Enforcement



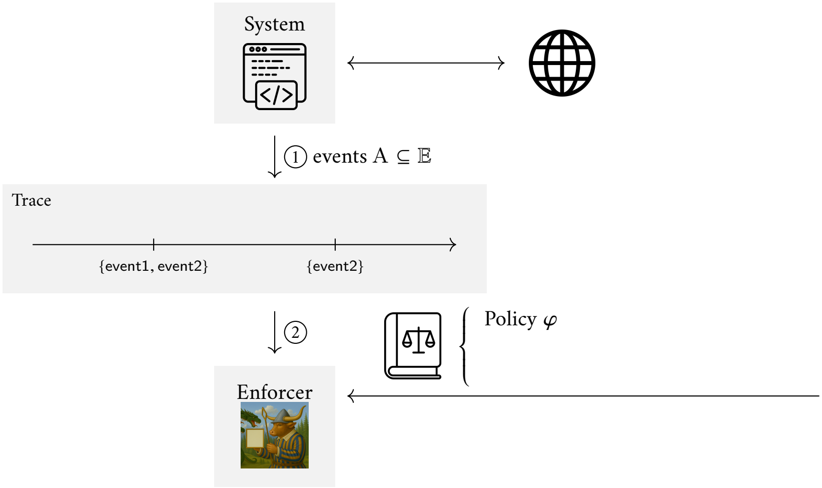
Runtime Enforcement



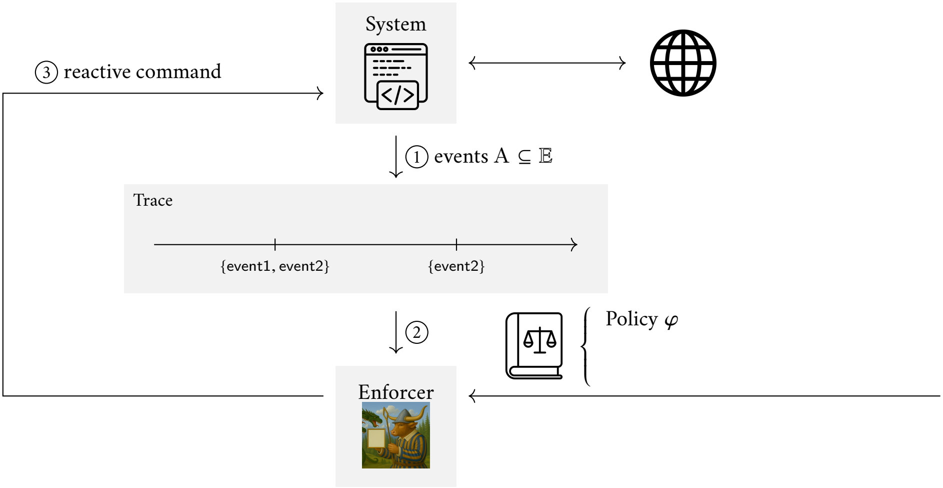
Runtime Enforcement



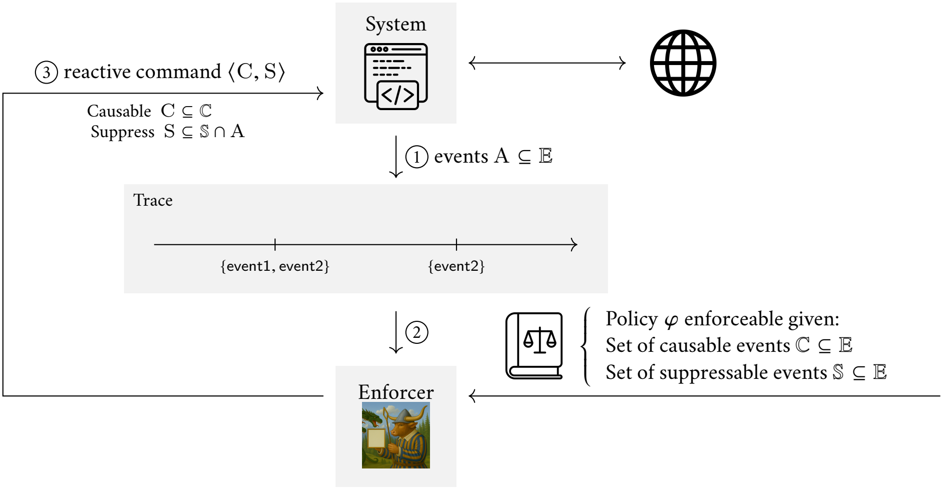
Runtime Enforcement



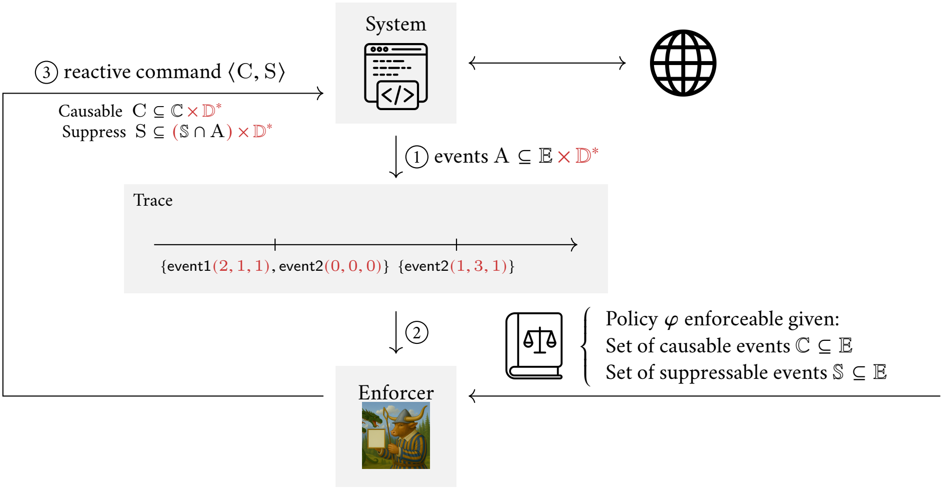
Runtime Enforcement



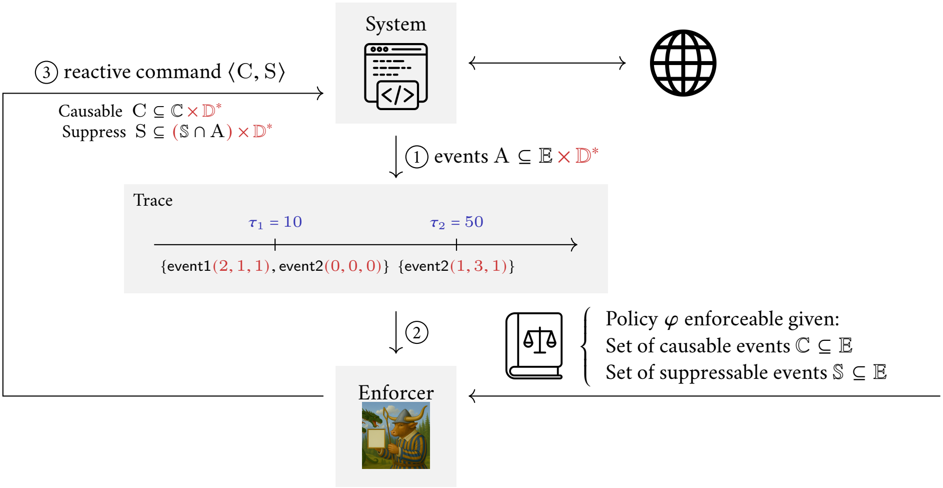
Runtime Enforcement



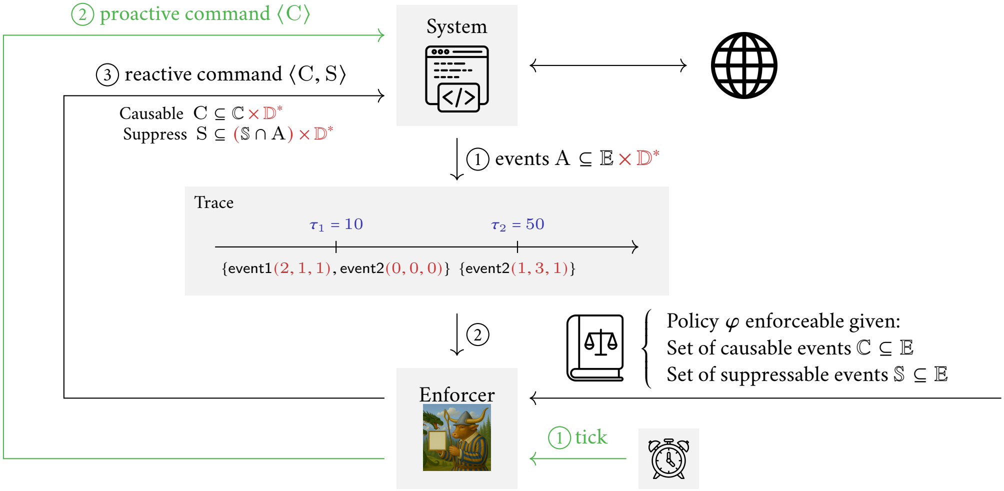
First-Order Runtime Enforcement



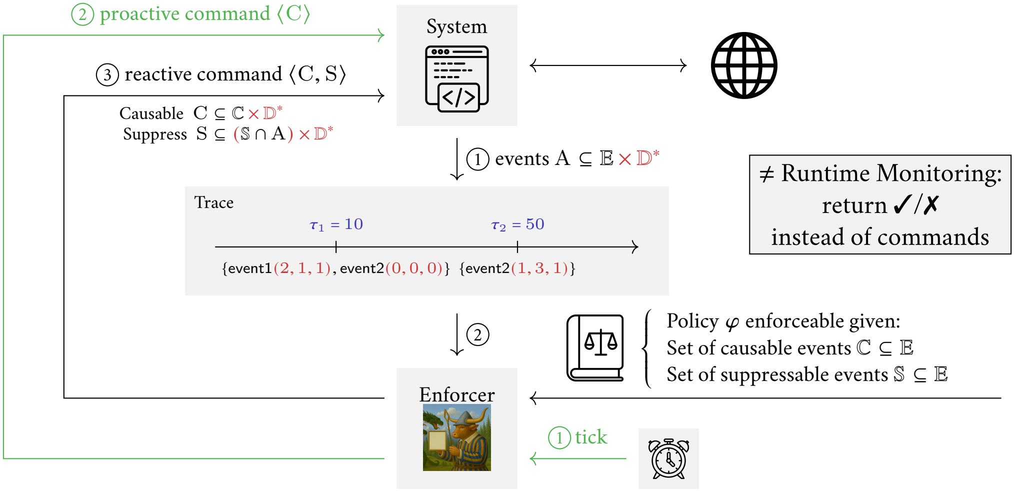
Real-Time First-Order Runtime Enforcement



Proactive Real-Time First-Order Runtime Enforcement



Proactive Real-Time First-Order Runtime Enforcement



Metric First-Order Temporal Logic (MFOTL)

Let $x \in \mathbb{V}$ be a variable, $c \in \mathbb{C}$ be a constant, $e \in \mathbb{E}$ be an event and $I \in \mathbb{N} \times \mathbb{N}$ be an interval,

- Syntax

$$
\begin{aligned}
t &::= x \mid c \\
\varphi &::= \mathsf{e}(t, \ldots, t) \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \rightarrow \varphi \mid \\
&\quad\;\; \exists x.\, \varphi \mid \forall x.\, \varphi \mid \bullet_I \varphi \mid \bigcirc_I \varphi \mid \varphi \mathcal{S}_I \varphi \mid \varphi \mathcal{U}_I \varphi
\end{aligned}
$$

Metric First-Order Temporal Logic (MFOTL)

Let $x \in \mathbb{V}$ be a variable, $c \in \mathbb{C}$ be a constant, $e \in \mathbb{E}$ be an event and $I \in \mathbb{N} \times \mathbb{N}$ be an interval,

- Syntax

$$
\begin{aligned}
t &::= x \mid c \\
\varphi &::= \mathsf{e}(t, \ldots, t) \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \rightarrow \varphi \mid \\
&\quad\; \exists x.\, \varphi \mid \forall x.\, \varphi \mid \bullet_I \varphi \mid \bigcirc_I \varphi \mid \varphi \mathcal{S}_I \varphi \mid \varphi \mathcal{U}_I \varphi
\end{aligned}
$$

Metric First-Order Temporal Logic (MFOTL)

Let $x \in \mathbb{V}$ be a variable, $c \in \mathbb{C}$ be a constant, $e \in \mathbb{E}$ be an event and $I \in \mathbb{N} \times \mathbb{N}$ be an interval,

- Syntax

$$
\begin{aligned}
t &::= x \mid c \\
\varphi &::= \mathsf{e}(t, \ldots, t) \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \rightarrow \varphi \mid \\
&\quad \exists x.\, \varphi \mid \forall x.\, \varphi \mid \bullet_I \varphi \mid \bigcirc_I \varphi \mid \varphi \mathcal{S}_I \varphi \mid \varphi \mathcal{U}_I \varphi
\end{aligned}
$$

Metric First-Order Temporal Logic (MFOTL)

Let $x \in \mathbb{V}$ be a variable, $c \in \mathbb{C}$ be a constant, $e \in \mathbb{E}$ be an event and $I \in \mathbb{N} \times \mathbb{N}$ be an interval,

- Syntax

$$
\begin{array}{lcl}
t & ::= & x \mid c \\
\varphi & ::= & \mathsf{e}(t, \ldots, t) \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \rightarrow \varphi \mid \\
 & & \exists x.\ \varphi \mid \forall x.\ \varphi \mid \bullet_I \varphi \mid \circ_I \varphi \mid \varphi \mathcal{S}_I \varphi \mid \varphi \mathcal{U}_I \varphi
\end{array}
$$

Metric First-Order Temporal Logic (MFOTL)

Let $x \in \mathbb{V}$ be a variable, $c \in \mathbb{C}$ be a constant, $e \in \mathbb{E}$ be an event and $I \in \mathbb{N} \times \mathbb{N}$ be an interval,

- Syntax

$$
\begin{aligned}
t &::= x \mid c \\
\varphi &::= \mathsf{e}(t, \ldots, t) \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \rightarrow \varphi \mid \\
&\quad\; \exists x.\, \varphi \mid \forall x.\, \varphi \mid \bullet_I \varphi \mid \bigcirc_I \varphi \mid \varphi \mathcal{S}_I \varphi \mid \varphi \mathcal{U}_I \varphi
\end{aligned}
$$

Metric First-Order Temporal Logic (MFOTL)

Let $x \in \mathbb{V}$ be a variable, $c \in \mathbb{C}$ be a constant, $e \in \mathbb{E}$ be an event and $I \in \mathbb{N} \times \mathbb{N}$ be an interval,

- Syntax

$$\begin{aligned} t &::= x \mid c \\ \varphi &::= \mathsf{e}(t, \ldots, t) \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \rightarrow \varphi \mid \\ &\quad\; \exists x.\ \varphi \mid \forall x.\ \varphi \mid \bullet_I \varphi \mid \bigcirc_I \varphi \mid \varphi \mathcal{S}_I \varphi \mid \varphi \mathcal{U}_I \varphi \end{aligned}$$

- Semantic judgments $v, i \models_\sigma \phi$ for valuation $v : \mathbb{V} \rightarrow \mathbb{D}$, trace σ, and time-point $i \in \mathbb{N}$: "ϕ holds on σ under v at time-point i"

Metric First-Order Temporal Logic (MFOTL)

Let $x \in \mathbb{V}$ be a variable, $c \in \mathbb{C}$ be a constant, $e \in \mathbb{E}$ be an event and $I \in \mathbb{N} \times \mathbb{N}$ be an interval,

- Syntax

$$
\begin{aligned}
t &::= x \mid c \\
\varphi &::= \mathsf{e}(t, \ldots, t) \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \rightarrow \varphi \mid \\
&\quad\; \exists x.\, \varphi \mid \forall x.\, \varphi \mid \bullet_I \varphi \mid \bigcirc_I \varphi \mid \varphi \mathcal{S}_I \varphi \mid \varphi \mathcal{U}_I \varphi
\end{aligned}
$$

- Semantic judgments $v, i \models_\sigma \phi$ for valuation $v : \mathbb{V} \rightarrow \mathbb{D}$, trace σ, and time-point $i \in \mathbb{N}$: "ϕ holds on σ under v at time-point i"
- Notation "always ϕ": $\Box\phi \equiv \neg(\top \,\mathcal{U}_{[0,\infty)}\, \neg\phi)$

Last year in Montréal (CAV'24)

First algorithm & tool for Proactive Real-Time First-Order Enforcement

1. EMFOTL, an enforceable fragment of MFOTL
2. Enforcement algorithm for EMFOTL
3. WhyEnf enforcement tool

Limitations:

- Expressiveness: "only" basic MFOTL
- Performance: 1–2 OOMs slower than monitors

Proactive Real-Time First-Order Enforcement

François Hublet[1], Leonardo Lima[2], David Basin[1], Srđan Krstić[1], and Dmitriy Traytel[2]

[1] ETH Zürich, Zurich, Switzerland
{francois.hublet, basin, srdan.krstic}@inf.ethz.ch
[2] University of Copenhagen, Denmark
{leonardo, traytel}@di.ku.dk

Abstract. Modern software systems must comply with increasingly complex regulations in domains ranging from industrial automation to data protection. Runtime enforcement addresses this challenge by empowering systems to not only observe, but also actively control, the behavior of target systems by modifying their actions to ensure policy compliance. We propose a novel approach to the proactive real-time enforcement of policies expressed in metric first-order temporal logic (MFOTL). We introduce a new system model, define an expressive MFOTL fragment that is enforceable in that model, and develop a sound enforcement algorithm for this fragment. We implement this algorithm in a tool called WhyEnf and carry out a case study on enforcing GDPR-related policies. Our tool can enforce all policies from the study in real-time with modest overhead. Our work thus provides the first tool-supported approach that can proactively enforce expressive first-order policies in real time.

Keywords: runtime enforcement · temporal logic · obligations

1 Introduction

As modern software systems become increasingly complex, they are required to comply with a myriad of growingly intricate regulations. The ability to monitor and control such systems is an important, technically challenging task.

Runtime *enforcement* [58] tackles this problem by observing and controlling a target system under scrutiny (SuS), so that its actions, possibly modified, comply with a given policy. Runtime enforcement is performed by a component called *enforcer*, which observes the SuS and influences its behavior as permitted by the system model, e.g., by suppressing or causing SuS actions. Enforcement is thus an inherently *online* problem performed during the SuS's execution. When time constraints are involved, enforcement is called *real-time*. This is a more difficult problem than runtime *monitoring* [8], where the SuS is only observed and policy violations are reported, but not prevented. Applications of runtime enforcement are manifold, ranging from safety protocols in industrial automation to regulatory compliance and it is closely related to the problem of controller synthesis [1,56].

Policies can be decomposed into provisions and obligations [37]. Compliance with provisions depends on past and present SuS behavior, and it is sufficient for an enforcer to react to the current SuS action. Compliance with obligations,

This year in Zagreb (CAV'25)

We extend and optimize our Proactive Real-Time First-Order Enforcement algorithm

This year in Zagreb (CAV'25)

We extend and optimize our Proactive Real-Time First-Order Enforcement algorithm

1. We extend our approach to support more expressive specifications

This year in Zagreb (CAV'25)

We extend and optimize our Proactive Real-Time First-Order Enforcement algorithm

1. We extend our approach to support more expressive specifications
   - Function applications in terms (built-in or user-defined functions)

This year in Zagreb (CAV'25)

We extend and optimize our Proactive Real-Time First-Order Enforcement algorithm

1. We extend our approach to support more expressive specifications
   - Function applications in terms (built-in or user-defined functions)
   - Aggregations (SQL-style `SUM`, `MAX`, `CNT`... or user-defined)

This year in Zagreb (CAV'25)

We extend and optimize our Proactive Real-Time First-Order Enforcement algorithm

1. We extend our approach to support more expressive specifications
   - Function applications in terms (built-in or user-defined functions)
   - Aggregations (SQL-style `SUM`, `MAX`, `CNT`... or user-defined)
   - Non-recursive let bindings

This year in Zagreb (CAV'25)

We extend and optimize our Proactive Real-Time First-Order Enforcement algorithm

1. We extend our approach to support more expressive specifications
   - Function applications in terms (built-in or user-defined functions)
   - Aggregations (SQL-style `SUM`, `MAX`, `CNT`... or user-defined)
   - Non-recursive let bindings

   For each extension: monitorability, monitoring, enforceability, enforcement

This year in Zagreb (CAV'25)

We extend and optimize our Proactive Real-Time First-Order Enforcement algorithm

1. We extend our approach to support more expressive specifications
   - Function applications in terms (built-in or user-defined functions)
   - Aggregations (SQL-style `SUM`, `MAX`, `CNT`... or user-defined)
   - Non-recursive let bindings

   For each extension: monitorability, monitoring, enforceability, enforcement

2. We optimize our algorithm and implement it in a new tool, EnfGuard

This year in Zagreb (CAV'25)

We extend and optimize our Proactive Real-Time First-Order Enforcement algorithm

1. We extend our approach to support more expressive specifications
   - Function applications in terms (built-in or user-defined functions)
   - Aggregations (SQL-style `SUM`, `MAX`, `CNT`... or user-defined)
   - Non-recursive let bindings

   For each extension: monitorability, monitoring, enforceability, enforcement
2. We optimize our algorithm and implement it in a new tool, EnfGuard
3. We comprehensively evaluate EnfGuard's performance

This year in Zagreb (CAV'25)

We extend and optimize our Proactive Real-Time First-Order Enforcement algorithm

1. We extend our approach to support more expressive specifications
   - Function applications in terms (built-in or user-defined functions)
   - Aggregations (SQL-style `SUM`, `MAX`, `CNT`... or user-defined)
   - Non-recursive let bindings

   For each extension: monitorability, monitoring, enforceability, enforcement
2. We optimize our algorithm and implement it in a new tool, EnfGuard
3. We comprehensively evaluate EnfGuard's performance
   - Novel benchmark set

This year in Zagreb (CAV'25)

We extend and optimize our Proactive Real-Time First-Order Enforcement algorithm

1. We extend our approach to support more expressive specifications
   - Function applications in terms (built-in or user-defined functions)
   - Aggregations (SQL-style `SUM`, `MAX`, `CNT`... or user-defined)
   - Non-recursive let bindings

   For each extension: monitorability, monitoring, enforceability, enforcement
2. We optimize our algorithm and implement it in a new tool, EnfGuard
3. We comprehensively evaluate EnfGuard's performance
   - Novel benchmark set
   - Improved performance: 1 OOM speed-up over previous work

Functions: Monitorability

Function symbols $f \in \mathbb{F}$ each associated with a computable function $\hat{f} : \mathbb{D}^{a(f)} \longrightarrow \mathbb{D}$.
Standard semantics given a valuation $v : \mathbb{V} \longrightarrow \mathbb{D}$:

$$[\![c]\!]_v = c \qquad [\![x]\!]_v = v(x) \qquad [\![f(t_1, \ldots, t_{a(f)})]\!]_v = \hat{f}([\![t_1]\!]_v, \ldots, [\![t_{a(f)}]\!]_v)$$

Functions: Monitorability

Function symbols $f \in \mathbb{F}$ each associated with a computable function $\hat{f} : \mathbb{D}^{a(f)} \longrightarrow \mathbb{D}$.
Standard semantics given a valuation $v : \mathbb{V} \longrightarrow \mathbb{D}$:

$$[\![c]\!]_v = c \qquad [\![x]\!]_v = v(x) \qquad [\![f(t_1, \ldots, t_{a(f)})]\!]_v = \hat{f}([\![t_1]\!]_v, \ldots, [\![t_{a(f)}]\!]_v)$$

Can we always compute the satisfactions of MFOTL formulae with function applications?

Functions: Monitorability

Function symbols $f \in \mathbb{F}$ each associated with a computable function $\hat{f} : \mathbb{D}^{a(f)} \longrightarrow \mathbb{D}$.
Standard semantics given a valuation $v : \mathbb{V} \longrightarrow \mathbb{D}$:

$$[\![c]\!]_v = c \qquad [\![x]\!]_v = v(x) \qquad [\![f(t_1, \ldots, t_{a(f)})]\!]_v = \hat{f}([\![t_1]\!]_v, \ldots, [\![t_{a(f)}]\!]_v)$$

Can we always compute the satisfactions of MFOTL formulae with function applications?

$$A(x) \longrightarrow B(f(x))$$

Functions: Monitorability

Function symbols $f \in \mathbb{F}$ each associated with a computable function $\hat{f} : \mathbb{D}^{a(f)} \longrightarrow \mathbb{D}$.
Standard semantics given a valuation $v : \mathbb{V} \longrightarrow \mathbb{D}$:

$$[\![c]\!]_v = c \qquad [\![x]\!]_v = v(x) \qquad [\![f(t_1, \ldots, t_{a(f)})]\!]_v = \hat{f}([\![t_1]\!]_v, \ldots, [\![t_{a(f)}]\!]_v)$$

Can we always compute the satisfactions of MFOTL formulae with function applications?

$A(x) \longrightarrow B(f(x))$ ✓

Functions: Monitorability

Function symbols $f \in \mathbb{F}$ each associated with a computable function $\hat{f} : \mathbb{D}^{a(f)} \to \mathbb{D}$.
Standard semantics given a valuation $v : \mathbb{V} \to \mathbb{D}$:

$$[\![c]\!]_v = c \qquad [\![x]\!]_v = v(x) \qquad [\![f(t_1, \ldots, t_{a(f)})]\!]_v = \hat{f}([\![t_1]\!]_v, \ldots, [\![t_{a(f)}]\!]_v)$$

Can we always compute the satisfactions of MFOTL formulae with function applications?

$A(x) \to B(f(x))$ ✓ for all v, if $\neg A(v(x))$, then $\top$, else $B(\hat{f}(v(x)))$

Functions: Monitorability

Function symbols $f \in \mathbb{F}$ each associated with a computable function $\hat{f} : \mathbb{D}^{a(f)} \longrightarrow \mathbb{D}$.
Standard semantics given a valuation $v : \mathbb{V} \longrightarrow \mathbb{D}$:

$$[\![c]\!]_v = c \qquad [\![x]\!]_v = v(x) \qquad [\![f(t_1, \ldots, t_{a(f)})]\!]_v = \hat{f}([\![t_1]\!]_v, \ldots, [\![t_{a(f)}]\!]_v)$$

Can we always compute the satisfactions of MFOTL formulae with function applications?

$A(x) \rightarrow B(f(x))$ ✓ for all v, if $\neg A(v(x))$, then $\top$, else $B(\hat{f}(v(x)))$

$A(f(x)) \rightarrow B(x)$

Functions: Monitorability

Function symbols $f \in \mathbb{F}$ each associated with a computable function $\hat{f} : \mathbb{D}^{a(f)} \longrightarrow \mathbb{D}$.
Standard semantics given a valuation $v : \mathbb{V} \longrightarrow \mathbb{D}$:

$$[\![c]\!]_v = c \qquad [\![x]\!]_v = v(x) \qquad [\![f(t_1, \ldots, t_{a(f)})]\!]_v = \hat{f}([\![t_1]\!]_v, \ldots, [\![t_{a(f)}]\!]_v)$$

Can we always compute the satisfactions of MFOTL formulae with function applications?

$A(x) \longrightarrow B(f(x))$	✓	for all v, if $\neg A(v(x))$, then $\top$, else $B(\hat{f}(v(x)))$
$A(f(x)) \longrightarrow B(x)$	✗	

Functions: Monitorability

Function symbols $f \in \mathbb{F}$ each associated with a computable function $\hat{f} : \mathbb{D}^{a(f)} \longrightarrow \mathbb{D}$.
Standard semantics given a valuation $v : \mathbb{V} \longrightarrow \mathbb{D}$:

$$[\![c]\!]_v = c \qquad [\![x]\!]_v = v(x) \qquad [\![f(t_1, \ldots, t_{a(f)})]\!]_v = \hat{f}([\![t_1]\!]_v, \ldots, [\![t_{a(f)}]\!]_v)$$

Can we always compute the satisfactions of MFOTL formulae with function applications?

$A(x) \rightarrow B(f(x))$	✓	for all v, if $\neg A(v(x))$, then $\top$, else $B(\hat{f}(v(x)))$
$A(f(x)) \rightarrow B(x)$	✗	what if $\hat{f}^{-1}$ is not computable?

Functions: Monitorability (cont'd)

How to ensure that the satisfactions can always be computed with *finitely many function evaluations*?

Functions: Monitorability (cont'd)

How to ensure that the satisfactions can always be computed with *finitely many function evaluations*?

Idea: the formula must evaluate to $\top$ of $\bot$ for values not present in the trace

Functions: Monitorability (cont'd)

How to ensure that the satisfactions can always be computed with *finitely many function evaluations*?

Idea: the formula must evaluate to $\top$ of $\bot$ for values not present in the trace

We introduced 'past-guardedness' typing of formulae such that

$$\vdash \phi : \mathsf{PG}^{+}(x) \implies \text{if } v, i \models_{\sigma} \phi, \text{ then } v(x) \text{ must be in } \sigma \text{ with time-point} \leq i$$
$$\vdash \phi : \mathsf{PG}^{-}(x) \implies \text{if } v, i \not\models_{\sigma} \phi, \text{ then } v(x) \text{ must be in } \sigma \text{ with time-point} \leq i$$

Functions: Monitorability (cont'd)

How to ensure that the satisfactions can always be computed with *finitely many function evaluations*?

Idea: the formula must evaluate to $\top$ of $\bot$ for values not present in the trace

We introduced 'past-guardedness' typing of formulae such that

$$\vdash \phi : \mathsf{PG}^{+}(x) \implies \text{if } v, i \models_{\sigma} \phi\text{, then } v(x) \text{ must be in } \sigma \text{ with time-point} \leq i$$
$$\vdash \phi : \mathsf{PG}^{-}(x) \implies \text{if } v, i \not\models_{\sigma} \phi\text{, then } v(x) \text{ must be in } \sigma \text{ with time-point} \leq i$$

Definition: Monitorability

A closed MFOTL formula ϕ is *monitorable* iff for any of its quantified subformulae $Qx.\ \psi$, where $Q \in \{\forall, \exists\}$, either $\vdash \psi : \mathsf{PG}^{+}(x)$, or $\vdash \psi : \mathsf{PG}^{-}(x)$, or x does not appear in any function argument in ψ.

Functions: Enforceability – EMFOTL (CAV'24)

- Typing judgements $\Gamma \vdash \phi : \mathbb{C}$ ("ϕ can be made true") or $\Gamma \vdash \phi : \mathbb{S}$ ("ϕ can be made false")
  + $\Gamma : \mathbb{E} \longrightarrow \{\mathbb{C}, \mathbb{S}\}$ fixes events that the enforcer will cause or suppress

Functions: Enforceability – EMFOTL (CAV'24)

- Typing judgements $\Gamma \vdash \phi : \mathbb{C}$ ("ϕ can be made true") or $\Gamma \vdash \phi : \mathbb{S}$ ("ϕ can be made false")
  + $\Gamma : \mathbb{E} \longrightarrow \{\mathbb{C}, \mathbb{S}\}$ fixes events that the enforcer will cause or suppress
- $\phi \in$ EMFOTL iff there exists Γ such that $\Gamma \vdash \phi : \mathbb{C}$

Functions: Enforceability – EMFOTL (CAV'24)

- Typing judgements $\Gamma \vdash \phi : \mathbb{C}$ ("ϕ can be made true") or $\Gamma \vdash \phi : \mathbb{S}$ ("ϕ can be made false")
  + $\Gamma : \mathbb{E} \longrightarrow \{\mathbb{C}, \mathbb{S}\}$ fixes events that the enforcer will cause or suppress
- $\phi \in$ EMFOTL iff there exists Γ such that $\Gamma \vdash \phi : \mathbb{C}$
- (Selected) rules

$$\frac{\Gamma(e) = \mathbb{C} \quad e \in \mathbb{C}}{\Gamma \vdash e(t_1, \ldots, t_{a(e)}) : \mathbb{C}}\ \mathbb{E}^{\mathbb{C}} \qquad \frac{\Gamma \vdash \psi : \mathbb{C}}{\Gamma \vdash \phi \longrightarrow \psi : \mathbb{C}} \longrightarrow^{\mathbb{C}R}$$

$$\frac{\Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \Box\phi : \mathbb{C}}\ \Box^{\mathbb{C}} \qquad \frac{\vdash \phi : \mathsf{PG}^{-}(x) \quad \Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \forall x.\, \phi : \mathbb{C}}\ \forall^{\mathbb{C}}$$

Function: Enforceability

$$\frac{\Gamma(e) = \mathbb{C} \quad e \in \mathbb{C}}{\Gamma \vdash e(t_1, \ldots, t_{a(e)}) : \mathbb{C}} \ \mathbb{E}^{\mathbb{C}} \qquad \frac{\Gamma \vdash \psi : \mathbb{C}}{\Gamma \vdash \phi \rightarrow \psi : \mathbb{C}} \rightarrow^{\mathbb{C}R}$$

$$\frac{\Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \Box \phi : \mathbb{C}} \ \Box^{\mathbb{C}} \qquad \frac{\vdash \phi : \mathsf{PG}(x)^{-} \quad \Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \forall x.\, \phi : \mathbb{C}} \ \forall^{\mathbb{C}}$$

Does this work with functions?

Function: Enforceability

$$\frac{\Gamma(e) = \mathbb{C} \quad e \in \mathbb{C}}{\Gamma \vdash e(t_1, \ldots, t_{a(e)}) : \mathbb{C}} \; \mathbb{E}^{\mathbb{C}} \qquad \frac{\Gamma \vdash \psi : \mathbb{C}}{\Gamma \vdash \phi \rightarrow \psi : \mathbb{C}} \; \rightarrow^{\mathbb{C}R}$$

$$\frac{\Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \Box\phi : \mathbb{C}} \; \Box^{\mathbb{C}} \qquad \frac{\vdash \phi : \mathsf{PG}(x)^{-} \quad \Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \forall x.\, \phi : \mathbb{C}} \; \forall^{\mathbb{C}}$$

Does this work with functions? Assume $A, B \in \mathbb{C}$.

$$\Box(\forall x.\, A(x) \rightarrow B(x+1))$$

Function: Enforceability

$$\frac{\Gamma(e) = \mathbb{C} \quad e \in \mathbb{C}}{\Gamma \vdash e(t_1, \ldots, t_{a(e)}) : \mathbb{C}}\ \mathbb{E}^{\mathbb{C}} \qquad \frac{\Gamma \vdash \psi : \mathbb{C}}{\Gamma \vdash \phi \rightarrow \psi : \mathbb{C}} \rightarrow^{\mathbb{C}\mathrm{R}}$$

$$\frac{\Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \Box\phi : \mathbb{C}}\ \Box^{\mathbb{C}} \qquad \frac{\vdash \phi : \mathsf{PG}(x)^{-} \quad \Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \forall x.\, \phi : \mathbb{C}}\ \forall^{\mathbb{C}}$$

Does this work with functions? Assume $A, B \in \mathbb{C}$.

$\Box(\forall x.\, A(x) \rightarrow B(x+1))$ ✓

Function: Enforceability

$$\frac{\Gamma(e) = \mathbb{C} \quad e \in \mathbb{C}}{\Gamma \vdash e(t_1, \ldots, t_{a(e)}) : \mathbb{C}}\ \mathbb{E}^{\mathbb{C}} \qquad \frac{\Gamma \vdash \psi : \mathbb{C}}{\Gamma \vdash \phi \to \psi : \mathbb{C}} \to^{\mathbb{C}R}$$

$$\frac{\Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \Box\phi : \mathbb{C}}\ \Box^{\mathbb{C}} \qquad \frac{\vdash \phi : \mathsf{PG}(x)^{-} \quad \Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \forall x.\ \phi : \mathbb{C}}\ \forall^{\mathbb{C}}$$

Does this work with functions? Assume $A, B \in \mathbb{C}$.

$\Box(\forall x.\ A(x) \to B(x+1))$ ✓ Cause $B(x+1)$ for each $A(x)$

Function: Enforceability

$$\frac{\Gamma(e) = \mathbb{C} \quad e \in \mathbb{C}}{\Gamma \vdash e(t_1, \ldots, t_{a(e)}) : \mathbb{C}}\ \mathbb{E}^{\mathbb{C}} \qquad \frac{\Gamma \vdash \psi : \mathbb{C}}{\Gamma \vdash \phi \to \psi : \mathbb{C}} \to^{\mathbb{C}R}$$

$$\frac{\Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \Box\phi : \mathbb{C}}\ \Box^{\mathbb{C}} \qquad \frac{\vdash \phi : \mathsf{PG}(x)^{-} \quad \Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \forall x.\, \phi : \mathbb{C}}\ \forall^{\mathbb{C}}$$

Does this work with functions? Assume $A, B \in \mathbb{C}$.

$\Box(\forall x.\ A(x) \to B(x+1))$ ✓ Cause $B(x+1)$ for each $A(x)$

$\Box(\forall x.\ A(x) \to A(\lfloor x / 2 \rfloor))$

Function: Enforceability

$$\frac{\Gamma(e) = \mathbb{C} \quad e \in \mathbb{C}}{\Gamma \vdash e(t_1, \ldots, t_{a(e)}) : \mathbb{C}}\ \mathbb{E}^{\mathbb{C}} \qquad \frac{\Gamma \vdash \psi : \mathbb{C}}{\Gamma \vdash \phi \to \psi : \mathbb{C}} \to^{\mathbb{C}R}$$

$$\frac{\Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \Box\phi : \mathbb{C}}\ \Box^{\mathbb{C}} \qquad \frac{\vdash \phi : \mathsf{PG}(x)^{-} \quad \Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \forall x.\, \phi : \mathbb{C}}\ \forall^{\mathbb{C}}$$

Does this work with functions? Assume $A, B \in \mathbb{C}$.

$\Box(\forall x.\ A(x) \to B(x+1))$ ✓ Cause $B(x+1)$ for each $A(x)$

$\Box(\forall x.\ A(x) \to A(\lfloor x / 2 \rfloor))$ ✓

Function: Enforceability

$$\frac{\Gamma(e) = \mathbb{C} \quad e \in \mathbb{C}}{\Gamma \vdash e(t_1, \ldots, t_{a(e)}) : \mathbb{C}}\ \mathbb{E}^{\mathbb{C}} \qquad \frac{\Gamma \vdash \psi : \mathbb{C}}{\Gamma \vdash \phi \rightarrow \psi : \mathbb{C}}\ \rightarrow^{\mathbb{C}\mathrm{R}}$$

$$\frac{\Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \square\phi : \mathbb{C}}\ \square^{\mathbb{C}} \qquad \frac{\vdash \phi : \mathsf{PG}(x)^{-} \quad \Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \forall x.\ \phi : \mathbb{C}}\ \forall^{\mathbb{C}}$$

Does this work with functions? Assume $A, B \in \mathbb{C}$.

$\square(\forall x.\ A(x) \rightarrow B(x+1))$	✓	Cause $B(x+1)$ for each $A(x)$
$\square(\forall x.\ A(x) \rightarrow A(\lfloor x / 2 \rfloor))$	✓	Cause $A(\lfloor x/2 \rfloor), A(\lfloor x/4 \rfloor), \ldots, A(0)$ for each $A(x)$

Function: Enforceability

$$\frac{\Gamma(e) = \mathbb{C} \quad e \in \mathbb{C}}{\Gamma \vdash e(t_1, \ldots, t_{a(e)}) : \mathbb{C}}\ \mathbb{E}^{\mathbb{C}} \qquad \frac{\Gamma \vdash \psi : \mathbb{C}}{\Gamma \vdash \phi \rightarrow \psi : \mathbb{C}} \rightarrow^{\mathbb{C}\mathrm{R}}$$

$$\frac{\Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \Box\phi : \mathbb{C}}\ \Box^{\mathbb{C}} \qquad \frac{\vdash \phi : \mathsf{PG}(x)^{-} \quad \Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \forall x.\, \phi : \mathbb{C}}\ \forall^{\mathbb{C}}$$

Does this work with functions? Assume $A, B \in \mathbb{C}$.

$\Box(\forall x.\ A(x) \rightarrow B(x+1))$	✓	Cause $B(x+1)$ for each $A(x)$
$\Box(\forall x.\ A(x) \rightarrow A(\lfloor x/2 \rfloor))$	✓	Cause $A(\lfloor x/2 \rfloor), A(\lfloor x/4 \rfloor), \ldots, A(0)$ for each $A(x)$
$\Box(\forall x.\ A(x) \rightarrow A(x+1))$		

Function: Enforceability

$$\frac{\Gamma(e) = \mathbb{C} \quad e \in \mathbb{C}}{\Gamma \vdash e(t_1, \ldots, t_{a(e)}) : \mathbb{C}}\ \mathbb{E}^{\mathbb{C}} \qquad \frac{\Gamma \vdash \psi : \mathbb{C}}{\Gamma \vdash \phi \to \psi : \mathbb{C}} \to^{\mathbb{C}R}$$

$$\frac{\Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \Box\phi : \mathbb{C}}\ \Box^{\mathbb{C}} \qquad \frac{\vdash \phi : \mathsf{PG}(x)^- \quad \Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \forall x.\ \phi : \mathbb{C}}\ \forall^{\mathbb{C}}$$

Does this work with functions? Assume $A, B \in \mathbb{C}$.

$\Box(\forall x.\ A(x) \to B(x+1))$	✓	Cause $B(x+1)$ for each $A(x)$
$\Box(\forall x.\ A(x) \to A(\lfloor x / 2 \rfloor))$	✓	Cause $A(\lfloor x/2 \rfloor), A(\lfloor x/4 \rfloor), \ldots, A(0)$ for each $A(x)$
$\Box(\forall x.\ A(x) \to A(x+1))$	✗	

Function: Enforceability

$$\frac{\Gamma(e) = \mathbb{C} \quad e \in \mathbb{C}}{\Gamma \vdash e(t_1, \ldots, t_{a(e)}) : \mathbb{C}} \; \mathbb{E}^{\mathbb{C}} \qquad \frac{\Gamma \vdash \psi : \mathbb{C}}{\Gamma \vdash \phi \to \psi : \mathbb{C}} \; \to^{\mathbb{C}\mathrm{R}}$$

$$\frac{\Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \Box\phi : \mathbb{C}} \; \Box^{\mathbb{C}} \qquad \frac{\vdash \phi : \mathsf{PG}(x)^{-} \quad \Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \forall x.\, \phi : \mathbb{C}} \; \forall^{\mathbb{C}}$$

Does this work with functions? Assume $A, B \in \mathbb{C}$.

$\Box(\forall x.\, A(x) \to B(x+1))$	✓	Cause $B(x+1)$ for each $A(x)$
$\Box(\forall x.\, A(x) \to A(\lfloor x / 2 \rfloor))$	✓	Cause $A(\lfloor x/2 \rfloor), A(\lfloor x/4 \rfloor), \ldots, A(0)$ for each $A(x)$
$\Box(\forall x.\, A(x) \to A(x+1))$	✗	Must insert infinitely many A events

Function: Enforceability

$$\frac{\Gamma(e) = \mathbb{C} \quad e \in \mathbb{C}}{\Gamma \vdash e(t_1, \ldots, t_{a(e)}) : \mathbb{C}} \; \mathbb{E}^{\mathbb{C}} \qquad \frac{\Gamma \vdash \psi : \mathbb{C}}{\Gamma \vdash \phi \rightarrow \psi : \mathbb{C}} \rightarrow^{\mathbb{C}\mathrm{R}}$$

$$\frac{\Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \Box\phi : \mathbb{C}} \; \Box^{\mathbb{C}} \qquad \frac{\vdash \phi : \mathsf{PG}(x)^{-} \quad \Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \forall x.\, \phi : \mathbb{C}} \; \forall^{\mathbb{C}}$$

Does this work with functions? Assume $A, B \in \mathbb{C}$.

$\Box(\forall x.\, A(x) \rightarrow B(x+1))$	✓	Cause $B(x+1)$ for each $A(x)$
$\Box(\forall x.\, A(x) \rightarrow A(\lfloor x / 2 \rfloor))$	✓	Cause $A(\lfloor x/2 \rfloor), A(\lfloor x/4 \rfloor), \ldots, A(0)$ for each $A(x)$
$\Box(\forall x.\, A(x) \rightarrow A(x+1))$	✗	Must insert infinitely many A events

Observe: events that guard x, functions that can generate infinitely many values, functions that can generate only finitely many values

Function: Enforceability

$$\frac{\Gamma(e) = \mathbb{C} \quad e \in \mathbb{C}}{\Gamma \vdash e(t_1, \ldots, t_{a(e)}) : \mathbb{C}} \mathbb{E}^{\mathbb{C}} \qquad \frac{\Gamma \vdash \psi : \mathbb{C}}{\Gamma \vdash \phi \to \psi : \mathbb{C}} \to^{\mathbb{C}R}$$

$$\frac{\Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \Box\phi : \mathbb{C}} \Box^{\mathbb{C}} \qquad \frac{\vdash \phi : \mathsf{PG}(x)^- \quad \Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \forall x.\, \phi : \mathbb{C}} \forall^{\mathbb{C}}$$

Does this work with functions? Assume $A, B \in \mathbb{C}$.

$\Box(\forall x.\, A(x) \to B(x+1))$	✓	Cause $B(x+1)$ for each $A(x)$
$\Box(\forall x.\, A(x) \to A(\lfloor x / 2 \rfloor))$	✓	Cause $A(\lfloor x/2 \rfloor), A(\lfloor x/4 \rfloor), \ldots, A(0)$ for each $A(x)$
$\Box(\forall x.\, A(x) \to A(x+1))$	✗	Must insert infinitely many A events

Observe: events that guard x, functions that can generate infinitely many values, functions that can generate only finitely many values

Function: Enforceability

$$\frac{\Gamma(e) = \mathbb{C} \quad e \in \mathbb{C}}{\Gamma \vdash e(t_1, \ldots, t_{a(e)}) : \mathbb{C}} \ \mathbb{E}^{\mathbb{C}} \qquad \frac{\Gamma \vdash \psi : \mathbb{C}}{\Gamma \vdash \phi \rightarrow \psi : \mathbb{C}} \rightarrow^{\mathbb{CR}}$$

$$\frac{\Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \Box\phi : \mathbb{C}} \ \Box^{\mathbb{C}} \qquad \frac{\vdash \phi : \mathsf{PG}(x)^- \quad \Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \forall x.\, \phi : \mathbb{C}} \ \forall^{\mathbb{C}}$$

Does this work with functions? Assume $A, B \in \mathbb{C}$.

$\Box(\forall x.\, A(x) \rightarrow B(x+1))$	✓	Cause $B(x+1)$ for each $A(x)$
$\Box(\forall x.\, A(x) \rightarrow A(\lfloor x\,/\,2 \rfloor))$	✓	Cause $A(\lfloor x/2 \rfloor), A(\lfloor x/4 \rfloor), \ldots, A(0)$ for each $A(x)$
$\Box(\forall x.\, A(x) \rightarrow A(x+1))$	✗	Must insert infinitely many A events

Observe: events that guard x, functions that can generate infinitely many values, functions that can generate only finitely many values

Function: Enforceability

$$\frac{\Gamma(e) = \mathbb{C} \quad e \in \mathbb{C}}{\Gamma \vdash e(t_1, \ldots, t_{a(e)}) : \mathbb{C}}\ \mathbb{E}^{\mathbb{C}} \qquad \frac{\Gamma \vdash \psi : \mathbb{C}}{\Gamma \vdash \phi \rightarrow \psi : \mathbb{C}} \rightarrow^{\mathbb{C}R}$$

$$\frac{\Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \Box\phi : \mathbb{C}}\ \Box^{\mathbb{C}} \qquad \frac{\vdash \phi : \mathsf{PG}(x)^{-} \quad \Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \forall x.\, \phi : \mathbb{C}}\ \forall^{\mathbb{C}}$$

Does this work with functions? Assume $A, B \in \mathbb{C}$.

$\Box(\forall x.\, A(x) \rightarrow B(x+1))$	✓	Cause $B(x+1)$ for each $A(x)$
$\Box(\forall x.\, A(x) \rightarrow A(\lfloor x / 2 \rfloor))$	✓	Cause $A(\lfloor x/2 \rfloor), A(\lfloor x/4 \rfloor), \ldots, A(0)$ for each $A(x)$
$\Box(\forall x.\, A(x) \rightarrow A(x+1))$	✗	Must insert infinitely many A events

Observe: events that guard x, functions that can generate infinitely many values, functions that can generate only finitely many values → the problem is the interaction guard + 'unstable' functions

Functions: Enforceability (cont'd)

Solution: If $A(x)$ is used as a guard, prevent $A(t)$ to be caused if t contains 'unstable' functions.

Functions: Enforceability (cont'd)

Solution: If $A(x)$ is used as a guard, prevent $A(t)$ to be caused if t contains 'unstable' functions.

Definition: Stable function

Let $\preceq$ be a well-founded relation on $\mathbb{D}$. A function $f : \mathbb{D}^k \to \mathbb{D}$ is $\preceq$-*stable* iff there exists a finite $C_f \subseteq \mathbb{D}$ such that for any $d_{\mathsf{sup}} \in \mathbb{D}$ and $d_1, \ldots, d_{a(f)} \preceq d_{\mathsf{sup}}$, either $f(d_1, \ldots, d_{a(f)}) \preceq d_{\mathsf{sup}}$ or $f(d_1, \ldots, d_{a(f)}) \in C_f$.

Functions: Enforceability (cont'd)

Solution: If $A(x)$ is used as a guard, prevent $A(t)$ to be caused if t contains 'unstable' functions.

Definition: Stable function

Let $\preceq$ be a well-founded relation on $\mathbb{D}$. A function $f : \mathbb{D}^k \longrightarrow \mathbb{D}$ is $\preceq$-*stable* iff there exists a finite $C_f \subseteq \mathbb{D}$ such that for any $d_{\mathsf{sup}} \in \mathbb{D}$ and $d_1, \ldots, d_{a(f)} \preceq d_{\mathsf{sup}}$, either $f(d_1, \ldots, d_{a(f)}) \preceq d_{\mathsf{sup}}$ or $f(d_1, \ldots, d_{a(f)}) \in C_f$.

Now, $\Gamma : \mathbb{E} \longrightarrow \{\mathbb{C}_s, \mathbb{C}_n, \mathbb{S}\}$.

Functions: Enforceability (cont'd)

Solution: If $A(x)$ is used as a guard, prevent $A(t)$ to be caused if t contains 'unstable' functions.

Definition: Stable function

Let $\preceq$ be a well-founded relation on $\mathbb{D}$. A function $f : \mathbb{D}^k \longrightarrow \mathbb{D}$ is $\preceq$-*stable* iff there exists a finite $C_f \subseteq \mathbb{D}$ such that for any $d_{\mathsf{sup}} \in \mathbb{D}$ and $d_1, \ldots, d_{a(f)} \preceq d_{\mathsf{sup}}$, either $f(d_1, \ldots, d_{a(f)}) \preceq d_{\mathsf{sup}}$ or $f(d_1, \ldots, d_{a(f)}) \in C_f$.

Now, $\Gamma : \mathbb{E} \longrightarrow \{\mathbb{C}_s, \mathbb{C}_n, \mathbb{S}\}$.

- $\Gamma(e) = \mathbb{S}$: the enforcer can suppress e events

Functions: Enforceability (cont'd)

Solution: If $A(x)$ is used as a guard, prevent $A(t)$ to be caused if t contains 'unstable' functions.

Definition: Stable function

Let $\preceq$ be a well-founded relation on $\mathbb{D}$. A function $f : \mathbb{D}^k \longrightarrow \mathbb{D}$ is $\preceq$-*stable* iff there exists a finite $C_f \subseteq \mathbb{D}$ such that for any $d_{\mathsf{sup}} \in \mathbb{D}$ and $d_1, \ldots, d_{a(f)} \preceq d_{\mathsf{sup}}$, either $f(d_1, \ldots, d_{a(f)}) \preceq d_{\mathsf{sup}}$ or $f(d_1, \ldots, d_{a(f)}) \in C_f$.

Now, $\Gamma : \mathbb{E} \longrightarrow \{\mathbb{C}_s, \mathbb{C}_n, \mathbb{S}\}$.

- $\Gamma(e) = \mathbb{S}$: the enforcer can suppress e events
- $\Gamma(e) = \mathbb{C}_s$: the enforcer can use e events as guards + cause e events containing stable functions

Functions: Enforceability (cont'd)

Solution: If $A(x)$ is used as a guard, prevent $A(t)$ to be caused if t contains 'unstable' functions.

Definition: Stable function

Let $\preceq$ be a well-founded relation on $\mathbb{D}$. A function $f : \mathbb{D}^k \longrightarrow \mathbb{D}$ is $\preceq$-*stable* iff there exists a finite $C_f \subseteq \mathbb{D}$ such that for any $d_{\mathsf{sup}} \in \mathbb{D}$ and $d_1, \ldots, d_{a(f)} \preceq d_{\mathsf{sup}}$, either $f(d_1, \ldots, d_{a(f)}) \preceq d_{\mathsf{sup}}$ or $f(d_1, \ldots, d_{a(f)}) \in C_f$.

Now, $\Gamma : \mathbb{E} \longrightarrow \{\mathbb{C}_s, \mathbb{C}_n, \mathbb{S}\}$.

- $\Gamma(e) = \mathbb{S}$: the enforcer can suppress e events
- $\Gamma(e) = \mathbb{C}_s$: the enforcer can use e events as guards + cause e events containing stable functions
- $\Gamma(e) = \mathbb{C}_n$: the enforcer can generate e events containing arbitrary functions

Functions: Enforceability (cont'd)

Solution: If $A(x)$ is used as a guard, prevent $A(t)$ to be caused if t contains 'unstable' functions.

Definition: Stable function

Let $\preceq$ be a well-founded relation on $\mathbb{D}$. A function $f : \mathbb{D}^k \longrightarrow \mathbb{D}$ is $\preceq$-*stable* iff there exists a finite $C_f \subseteq \mathbb{D}$ such that for any $d_{\mathsf{sup}} \in \mathbb{D}$ and $d_1, \ldots, d_{a(f)} \preceq d_{\mathsf{sup}}$, either $f(d_1, \ldots, d_{a(f)}) \preceq d_{\mathsf{sup}}$ or $f(d_1, \ldots, d_{a(f)}) \in C_f$.

Now, $\Gamma : \mathbb{E} \longrightarrow \{\mathbb{C}_s, \mathbb{C}_n, \mathbb{S}\}$.

- $\Gamma(e) = \mathbb{S}$: the enforcer can suppress e events
- $\Gamma(e) = \mathbb{C}_s$: the enforcer can use e events as guards + cause e events containing stable functions
- $\Gamma(e) = \mathbb{C}_n$: the enforcer can generate e events containing arbitrary functions

$\longrightarrow$ details in the paper

Implementation

- New open-source tool

EnfGuard
proactive enforcer

- Code base: ~10k loc (OCaml)
- Includes optimizations → details in paper

Evaluation

RQ1. Can EnfGuard's EMFOTL fragment formalize real-world policies?
RQ2. At what maximum event rate can EnfGuard perform real-time enforcement?
RQ3. Does EnfGuard's performance improve upon the state-of-the-art?

	Log statistics				Formulae statistics					Tool support			
Name	Real	#logs	max |log|	max *er*	max $\|\varphi\|$	let bindings	Aggregations	Functions	#formulae	EnfGuard	WhyEnf	EnfPoly	MonPoly
GDPR	✓	1	5,631	10^{-4}	72				6	6	6	2	6
GPDR$^{\text{FUN}}$	✓	1	5,631	10^{-4}	108				6	6			
NOKIA	✓	1	9,458,824	109	44			✓	11	11	11	5	11
IC	✓	3	634,789	147	179	✓		✓	8	8			8
AGG		2	100,000		34		✓	✓	6	6			6
CLUSTER		1	5,000		42	✓		✓	2	2			
								Total:	39	39	17	7	31
					Rewriting required:					no	no	yes	yes

The largest benchmark suite for runtime enforcement!

Evaluation: Expressiveness (RQ1)

EnfGuard supports more policies (39/39) than the SOTA MFOTL monitor MonPoly [Basin et al., 2017] (31/39) and significantly more than WhyEnf (17/39).

Evaluation: Expressiveness (RQ1)

EnfGuard supports more policies (39/39) than the SOTA MFOTL monitor MonPoly [Basin et al., 2017] (31/39) and significantly more than WhyEnf (17/39).

"**Block validation latency**" (from IC benchmark [Basin et al., 2023])

```
LET node_added_to_subnet(node_id, node_addr, subnet) = ... IN
LET node_removed_from_subnet(node_id, node_addr) = ... IN
LET in_subnet(node_id, node_addr, subnet) = ... IN
LET subnet_size(subnet_id, n) = ... IN
LET block_added(node_id, subnet_id, block, t_add) = ... IN
LET validated(block, subnet_id, t_add) =
  EXISTS n_validated, n_subnet. (n_validated <- CNT (valid_node; block, subnet_id, t_add; ...)
    AND subnet_size(subnet_id, n_subnet)
    AND (float_of_int(n_validated) > 2. *. float_of_int(n_subnet) /. 3.) IN
LET time_per_block(block, subnet_id, time) = ... IN
LET subnet_type_assoc(subnet_id, subnet_type) = ... IN
LET subnet_type_map(subnet_id, subnet_type) = ... IN
FORALL block, subnet_id, time.
  time_per_block(block, subnet_id, time)
  AND ((subnet_type_map(subnet_id, "System") AND (time > 3000))
  OR ((subnet_type_map(subnet_id, "Application")
  OR subnet_type_map(subnet_id, "VerifiedApplication")) AND (time > 1000)))
IMPLIES alert_validation_latency(block, subnet_id, time)
```

Evaluation: Performance (RQ2–3)

- Event rate er: number of events **in the trace** per time unit (timestamp time)

Evaluation: Performance (RQ2–3)

- Event rate er: number of events **in the trace** per time unit (timestamp time)
- Acceleration a = trace timestamp step / real time step

Evaluation: Performance (RQ2–3)

- Event rate er: number of events **in the trace** per time unit (timestamp time)
- Acceleration a = trace timestamp step / real time step
- Latency ℓ: time difference (real time) between event input and command output

Evaluation: Performance (RQ2–3)

- Event rate er: number of events **in the trace** per time unit (timestamp time)
- Acceleration a = trace timestamp step / real time step
- Latency ℓ: time difference (real time) between event input and command output
- $\mathsf{max}_\ell(\mathrm{a})$: maximum latency at acceleration a

Evaluation: Performance (RQ2–3)

- Event rate er: number of events **in the trace** per time unit (timestamp time)
- Acceleration a = trace timestamp step / real time step
- Latency ℓ: time difference (real time) between event input and command output
- $\mathsf{max}_\ell(a)$: maximum latency at acceleration a
- Real-time condition: $\mathsf{max}_\ell(a) \leq 1/a$

Evaluation: Performance (RQ2–3)

- Event rate er: number of events **in the trace** per time unit (timestamp time)
- Acceleration a = trace timestamp step / real time step
- Latency ℓ: time difference (real time) between event input and command output
- $\mathsf{max}_\ell(\mathrm{a})$: maximum latency at acceleration a
- Real-time condition: $\mathsf{max}_\ell(\mathrm{a}) \leq 1/\mathrm{a}$
- We report $\mathsf{avg}_{\mathrm{er}}$ at the maximum latency fulfilling the real-time condition

Evaluation: Performance (RQ2–3)

Comparison with WhyEnf, EnfPoly [Hublet et al., 2022], MonPoly (**monitor**).
Note: EnfPoly covers only 7/39 formulae.

GPDR benchmarks:

		EnfGuard			WhyEnf			EnfPoly			MonPoly		
Policy ϕ	$\lvert\phi\rvert$	avg_{er}	avg_ℓ	max_ℓ	avg_{er}	avg_ℓ	max_ℓ	avg_{er}	avg_ℓ	max_ℓ	avg_{er}	avg_ℓ	max_ℓ
consent	22	1619	.39	2	101	7.6	30	6480	.17	1	6934	.20	1
deletion	14	3238	.28	2	3238	.20	1				6934	.20	1
gdpr	72	810	.87	3	25	33	110				3465	.13	1
information	16	1619	.33	2	810	1.1	5.2				6934	.15	1
lawfulness	17	1619	.35	2	810	1.3	4.4	6480	.17	1	6934	.15	1
sharing	19	1619	.32	2	405	3.0	15				6934	.20	1

Consistent findings on other benchmarks (not shown here):

- EnfGuard 2–10× faster than WhyEnf
- Slightly slower but much better coverage than EnfPoly
- Difference with MonPoly: PDT- rather than table-based

Thank you for your attention!

If you are interested in this work, feel free to drop us an e-mail:

François Hublet	`francois.hublet@inf.ethz.ch`
Srđan Krstić	`srdan.krstic@inf.ethz.ch`

Scaling Up Proactive Enforcement

François Hublet[1], Leonardo Lima[2], David Basin[1], Srđan Krstić[1], and Dmitriy Traytel[2]

[1] ETH Zürich, Zurich, Switzerland
{francois.hublet, basin, srdan.krstic}@inf.ethz.ch
[2] University of Copenhagen, Denmark
{leonardo, traytel}@di.ku.dk

Abstract. Runtime enforcers receive events from a system and output commands ensuring the system's policy compliance. Proactive enforcers extend traditional (reactive) enforcers by emitting commands at any time, rather only as a response to system actions. However, proactive enforcers have so far lacked support for many useful policy features. This, along with the existing tools' poor performance, hinders their adoption. We present a performance-optimized, proactive enforcement algorithm for a rich policy language: metric first-order temporal logic with function applications, aggregations, and let bindings. We have implemented this algorithm in EnfGuard, the first proactive enforcer tool that supports the above constructs. We evaluated our tool using a novel set of six benchmarks containing both real-world and synthetic policies and logs, demonstrating that it enforces realistic policies out-of-the-box and achieves the necessary performance to be used in real-time systems.

1 Introduction

Statically certifying the behavior of large, complex systems is often impossible. As an alternative, runtime enforcement [41] has emerged as a family of techniques aimed at observing and correcting the behavior of systems during their execution.

In runtime enforcement, an *enforcer* is a policy enforcement mechanism that observes the real-time execution of a system under enforcement (SuE) through the sequence of *events* that occur in it and sends *commands* to the SuE to ensure policy compliance (Figure 1). These commands instruct the system to suppress, cause, modify, or delay specific events. In *reactive* enforcement, the enforcer emits commands immediately upon receiving events (Figure 1, interactions 1.1–1.2). In *proactive* enforcement [4], the enforcer can additionally give commands at any time, rather than only after SuE events (Figure 1, interactions 2.1–2.2). This is crucial whenever policies require action to be taken before a deadline, even in the absence of SuE actions, as in common, e.g., in privacy regulations [24].

SuE
1.1: events
1.2: (reactive) command
2.2: (proactive) command
Enforcer
policy P
1.1, 2.1: time τ

Fig. 1: Communication diagram for enforcement. R-step: 1.1, 1.2; P-step: 2.1, 2.2

Thank you for your attention!

If you are interested in this work, feel free to drop us an e-mail:

François Hublet   francois.hublet@inf.ethz.ch
Srđan Krstić   srdan.krstic@inf.ethz.ch

Any questions?

Scaling Up Proactive Enforcement

François Hublet[1], Leonardo Lima[2], David Basin[1], Srđan Krstić[1], and Dmitriy Traytel[2]

[1] ETH Zürich, Zurich, Switzerland
{francois.hublet, basin, srdan.krstic}@inf.ethz.ch
[2] University of Copenhagen, Denmark
{leonardo, traytel}@di.ku.dk

Abstract. Runtime enforcers receive events from a system and output commands ensuring the system's policy compliance. Proactive enforcers extend traditional (reactive) enforcers by emitting commands at any time, rather only as a response to system actions. However, proactive enforcers have so far lacked support for many useful policy features. This, along with the existing tools' poor performance, hinders their adoption. We present a performance-optimized, proactive enforcement algorithm for a rich policy language: metric first-order temporal logic with function applications, aggregations, and let bindings. We have implemented this algorithm in EnfGuard, the first proactive enforcer tool that supports the above constructs. We evaluated our tool using a novel set of six benchmarks containing both real-world and synthetic policies and logs, demonstrating that it enforces realistic policies out-of-the-box and achieves the necessary performance to be used in real-time systems.

1 Introduction

Statically certifying the behavior of large, complex systems is often impossible. As an alternative, runtime enforcement [41] has emerged as a family of techniques aimed at observing and correcting the behavior of systems during their execution.

In runtime enforcement, an *enforcer* is a policy enforcement mechanism that observes the real-time execution of a system under enforcement (SuE) through the sequence of *events* that occur in it and sends *commands* to the SuE to ensure policy compliance (Figure 1). These commands instruct the system to suppress, cause, modify, or delay specific events. In *reactive* enforcement, the enforcer emits commands immediately upon receiving events (Figure 1, interactions 1.1–1.2). In *proactive* enforcement [4], the enforcer can additionally give commands at any time, rather than only after SuE events (Figure 1, interactions 2.1–2.2). This is crucial whenever policies require action to be taken before a deadline, even in the absence of SuE actions, as in common, e.g., in privacy regulations [24].

Fig. 1: Communication diagram for enforcement. R-step: 1.1, 1.2; P-step: 2.1, 2.2