Proactive Real-Time First-Order Enforcement

François Hublet
francois.hublet@inf.ethz.ch

Leonardo Lima
leonardo@di.ku.dk

David Basin
basin@inf.ethz.ch

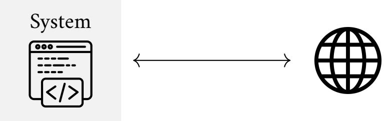
Srđan Krstić
srdan.krstic@inf.ethz.ch

Dmitriy Traytel
traytel@di.ku.dk

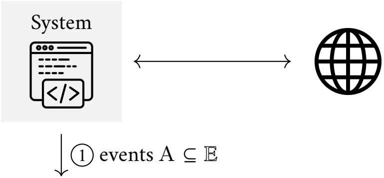
International Conference on Computer-Aided Verification — Montréal, July 26, 2024

ETH zürich

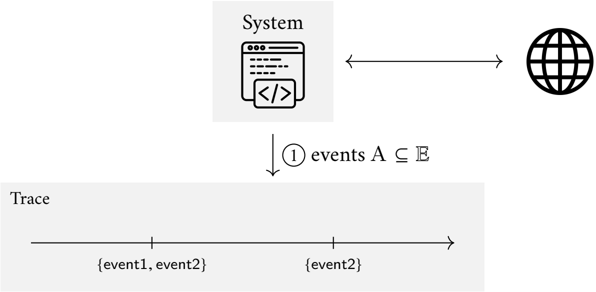
Runtime Enforcement



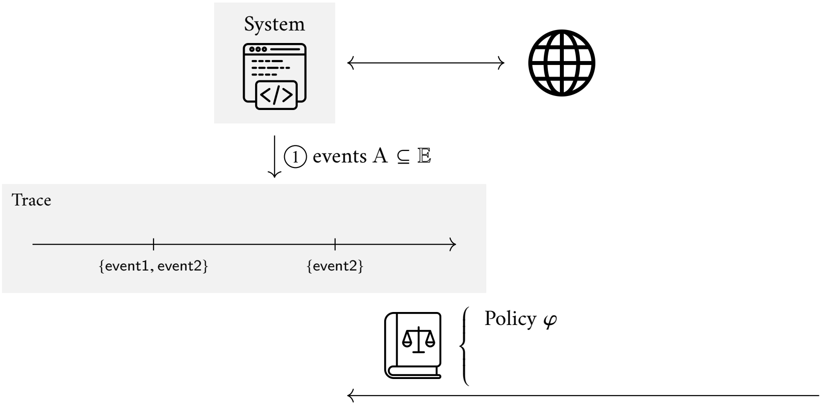
Runtime Enforcement



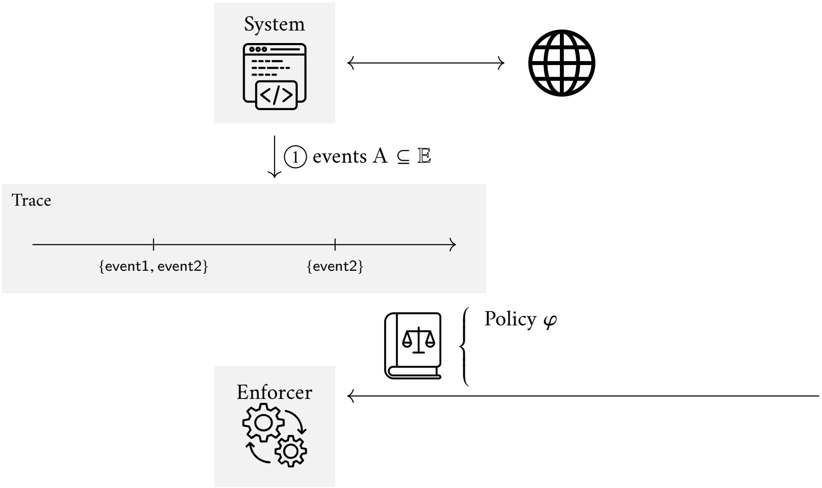
Runtime Enforcement



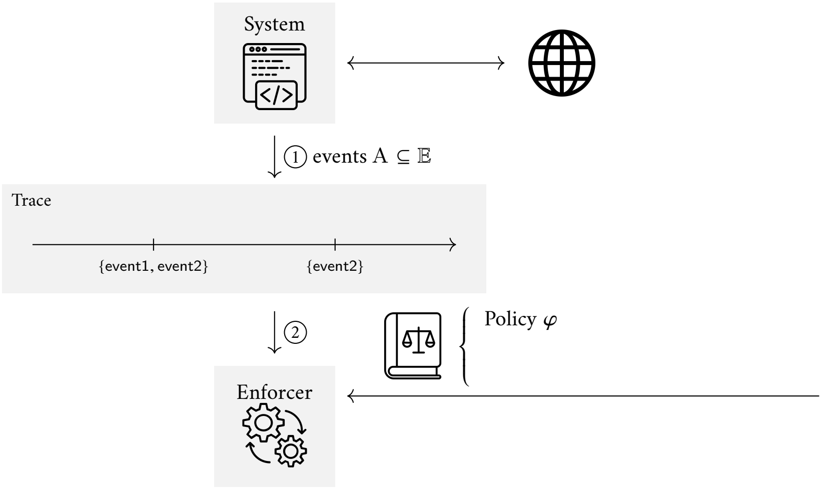
Runtime Enforcement



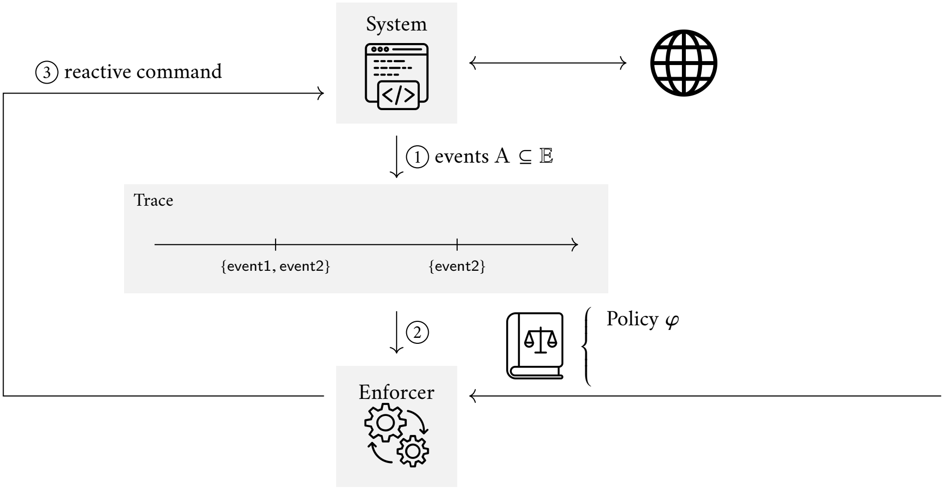
Runtime Enforcement



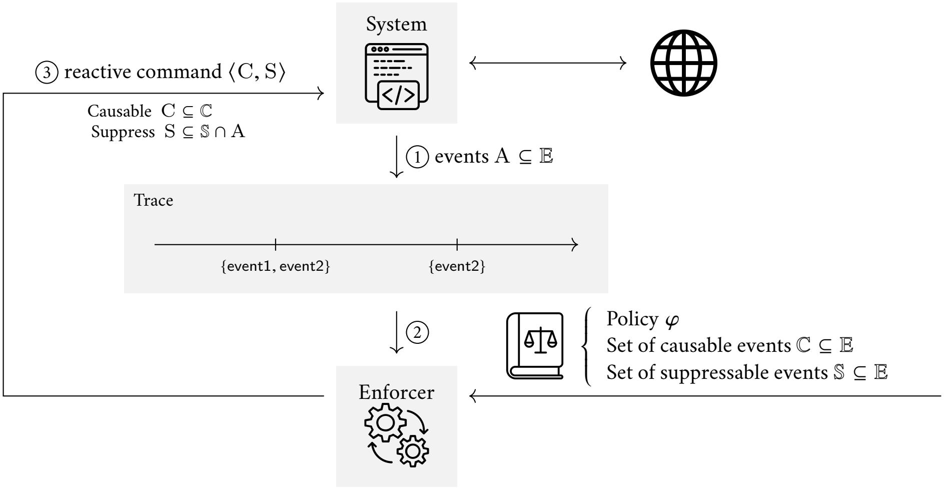
Runtime Enforcement



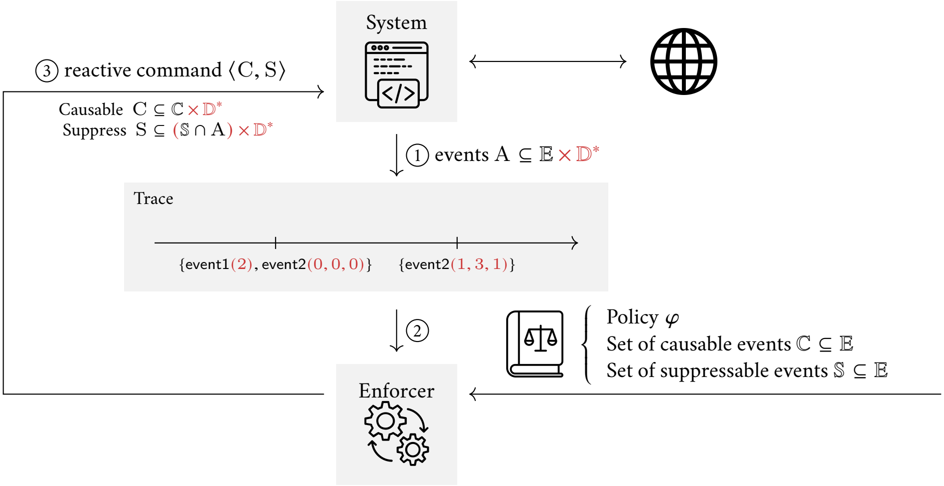
Runtime Enforcement



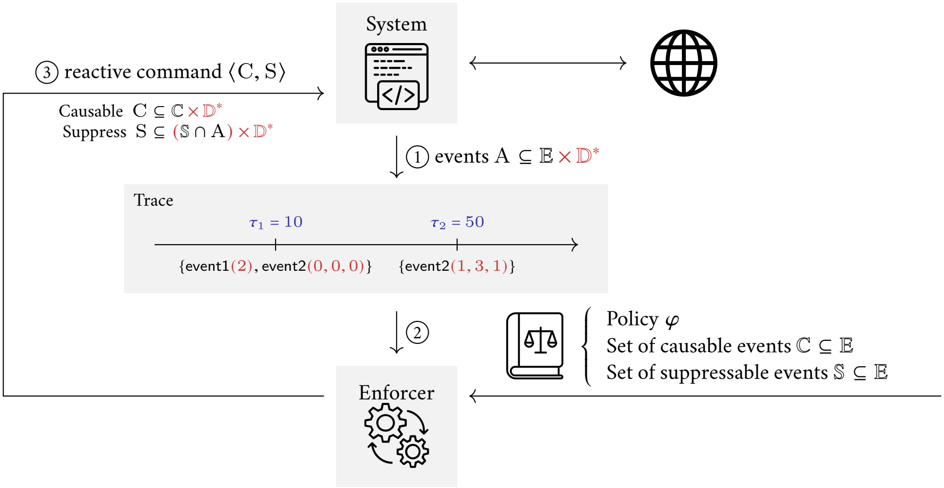
Runtime Enforcement



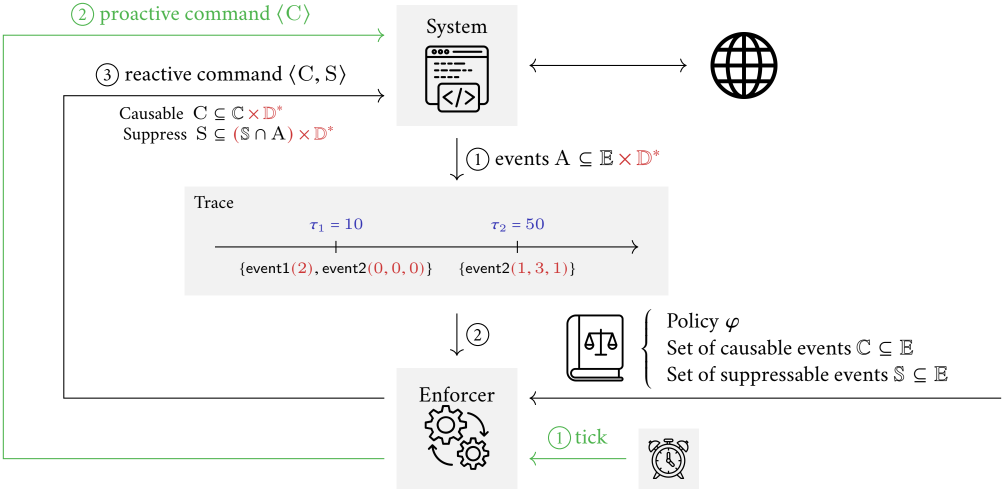
First-Order Runtime Enforcement



Real-Time First-Order Runtime Enforcement



Proactive Real-Time First-Order Runtime Enforcement



Motivating example

> *The data subject shall have the right to* ***obtain from the controller the erasure of personal data concerning him or her without undue delay****.*
>
> — *General Data Protection Regulation, Article 17*

Motivating example

> *The data subject shall have the right to* ***obtain from the controller the erasure of personal data concerning him or her without undue delay****.*
>
> — *General Data Protection Regulation, Article 17*

To enforce this, we need to set a bound for 'undue delay' and:

Motivating example

> *The data subject shall have the right to* ***obtain from the controller the erasure of personal data concerning him or her without undue delay****.*
>
> — *General Data Protection Regulation, Article 17*

To enforce this, we need to set a bound for 'undue delay' and:

- Distinguish between different users' requests $\longrightarrow$ First-Order

Motivating example

The data subject shall have the right to ***obtain from the controller the erasure of personal data concerning him or her without undue delay****.*

— General Data Protection Regulation, Article 17

To enforce this, we need to set a bound for 'undue delay' and:

- Distinguish between different users' requests → First-Order
- Capture real-time delays → Real-Time

Motivating example

> *The data subject shall have the right to* ***obtain from the controller the erasure of personal data concerning him or her without undue delay****.*
>
> — *General Data Protection Regulation, Article 17*

To enforce this, we need to set a bound for 'undue delay' and:

- Distinguish between different users' requests ⟶ First-Order
- Capture real-time delays ⟶ Real-Time
- Erase data when an 'undue delay' is about to elapse ⟶ Proactive

Motivating example

> *The data subject shall have the right to **obtain from the controller the erasure of personal data concerning him or her without undue delay**.*
>
> — *General Data Protection Regulation, Article 17*

To enforce this, we need to set a bound for 'undue delay' and:

- Distinguish between different users' requests ⟶ First-Order
- Capture real-time delays ⟶ Real-Time
- Erase data when an 'undue delay' is about to elapse ⟶ Proactive

Similarly for other data protection requirements, kernels, firewalls...

Related work

- Runtime enforcement: 'Enforceable security policies' [Schneider, 2000]
  + Enforces policies by blocking the system, automata as policies

Related work

- Runtime enforcement: ‘Enforceable security policies’ [Schneider, 2000]
  + Enforces policies by blocking the system, automata as policies
- Later extensions:
  + Causation vs. suppression [Bauer et al., 2002; Ligatti et al., 2005]

Related work

- Runtime enforcement: ‘Enforceable security policies’ [Schneider, 2000]
  - + Enforces policies by blocking the system, automata as policies
- Later extensions:
  - + Causation vs. suppression [Bauer et al., 2002; Ligatti et al., 2005]
  - + Suppressable vs. only-observable events [Basin et al., 2013]

Related work

- Runtime enforcement: ‘Enforceable security policies’ [Schneider, 2000]
  - + Enforces policies by blocking the system, automata as policies
- Later extensions:
  - + Causation vs. suppression [Bauer et al., 2002; Ligatti et al., 2005]
  - + Suppressable vs. only-observable events [Basin et al., 2013]
  - + Proactive enforcement [Basin et al., 2016; 2024]

Related work

- Runtime enforcement: 'Enforceable security policies' [Schneider, 2000]
  - \+ Enforces policies by blocking the system, automata as policies
- Later extensions:
  - \+ Causation vs. suppression [Bauer et al., 2002; Ligatti et al., 2005]
  - \+ Suppressable vs. only-observable events [Basin et al., 2013]
  - \+ Proactive enforcement [Basin et al., 2016; 2024]
- Few tools for enforcement of first-order temporal logic
  - BeepBeep [Hallé and Villemaire, 2009]: (future) LTL-FO, suppression only
  - EnfPoly [Hublet et al., 2022]: Restricted fragment, mostly past-only
  - [Aceto et al., 2018; 2021; 2023]: HML-FO, suppression only

Related work

- Runtime enforcement: 'Enforceable security policies' [Schneider, 2000]
  - + Enforces policies by blocking the system, automata as policies
- Later extensions:
  - + Causation vs. suppression [Bauer et al., 2002; Ligatti et al., 2005]
  - + Suppressable vs. only-observable events [Basin et al., 2013]
  - + Proactive enforcement [Basin et al., 2016; 2024]
- Few tools for enforcement of first-order temporal logic
  - BeepBeep [Hallé and Villemaire, 2009]: (future) LTL-FO, suppression only
  - EnfPoly [Hublet et al., 2022]: Restricted fragment, mostly past-only
  - [Aceto et al., 2018; 2021; 2023]: HML-FO, suppression only
- Monitoring of first-order temporal logic: MonPoly, Verimon, DejaVu, WhyMon...

Contributions

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

Policy language: Metric First-Order Temporal Logic (MFOTL)

1. New **system model** for real-time proactive enforcement of first-order policies
2. EMFOTL, an expressive **enforceable fragment** of MFOTL
3. **Enforcement algorithm** for EMFOTL
4. **WhyEnf** enforcement tool

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

$$
\begin{aligned}
t \quad &::= \quad x \mid c \\
\varphi \quad &::= \quad \mathsf{e}(t_1, \ldots, t_n) \mid \top \mid \bot \mid \neg\varphi \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \\
&\quad\;\; \varphi \rightarrow \varphi \mid \exists x.\ \varphi \mid \forall x.\ \varphi \mid \bullet_I \varphi \mid \bigcirc_I \varphi \mid \blacklozenge_I \varphi \mid \lozenge_I \varphi \mid \\
&\quad\;\; \blacksquare_I \varphi \mid \square_I \varphi \mid \varphi\ \mathcal{S}_I\ \varphi \mid \varphi\ \mathcal{U}_I\ \varphi
\end{aligned}
$$

Metric First-Order Temporal Logic (MFOTL)

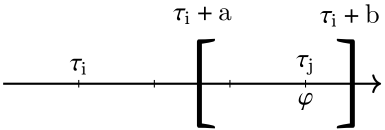
Let $x \in \mathbb{V}$ be a variable, $c \in \mathbb{C}$ be a constant, $e \in \mathbb{E}$ be an event and $I \in \mathbb{N} \times \mathbb{N}$ be an interval,

- Syntax

$$
\begin{aligned}
t &::= x \mid c \\
\varphi &::= e(t_1, \ldots, t_n) \mid \top \mid \bot \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \\
&\quad \varphi \rightarrow \varphi \mid \exists x.\ \varphi \mid \forall x.\ \varphi \mid \bullet_I \varphi \mid \bigcirc_I \varphi \mid \blacklozenge_I \varphi \mid \Diamond_I \varphi \mid \\
&\quad \blacksquare_I \varphi \mid \Box_I \varphi \mid \varphi\ \mathcal{S}_I\ \varphi \mid \varphi\ \mathcal{U}_I\ \varphi
\end{aligned}
$$

- Semantics (for a fixed trace $\langle(\tau_0, D_0), (\tau_1, D_1), \ldots\rangle$ and valuation $v : \mathbb{V} \mapsto \mathbb{D}$)

$$v, i \models \Diamond_{[a,b]}\varphi \iff v, j \models \varphi \textbf{ for some } j \geq i \textbf{ with } \tau_j - \tau_i \in [a, b]$$



MFOTL: Example

> *The data subject shall have the right to* ***obtain from the controller the erasure of personal data concerning him or her without undue delay****.*
>
> *— General Data Protection Regulation, Article 17*

MFOTL: Example

> *The data subject shall have the right to* ***obtain from the controller the erasure of personal data concerning him or her without undue delay****.*
>
> — *General Data Protection Regulation, Article 17*

- Policy (informal, ‘undue delay’ $\sim$ > 30 days)

WHENEVER: user **u** requests the deletion of data **d**

ENSURE: **d** is deleted within 30 days

MFOTL: Example

> *The data subject shall have the right to* ***obtain from the controller the erasure of personal data concerning him or her without undue delay****.*
>
> — *General Data Protection Regulation, Article 17*

- Policy (informal, 'undue delay' ~ > 30 days)

WHENEVER: user **u** requests the deletion of data **d**

ENSURE: **d** is deleted within 30 days

- Policy (MFOTL)

$$\square\left(\forall d, u.\ \mathsf{deletion\_request}(u, d) \longrightarrow \Diamond_{[0,30]}\mathsf{delete}(d)\right)$$

EMFOTL: an enforceable MFOTL fragment

- Restriction: require that every event kind is only caused or only suppressed [Hublet et al., 2022]
  - + Avoids degenerate cases such as $e \wedge \neg e$ with e both causable and suppressable
- $\phi \in$ EMFOTL iff there exists Γ such that $\Gamma \vdash \phi : \mathbb{C}$
  - + The typing context Γ is a mapping: $\mathbb{E} \longrightarrow \{\mathbb{C}, \mathbb{S}\}$
- (Selected) rules

$$\frac{\Gamma \vdash \phi : \mathbb{S}}{\Gamma \vdash \phi \rightarrow \psi : \mathbb{C}} \rightarrow^{\mathbb{C}L} \qquad \frac{\Gamma \vdash \psi : \mathbb{C}}{\Gamma \vdash \phi \rightarrow \psi : \mathbb{C}} \rightarrow^{\mathbb{C}R}$$

$$\frac{\Gamma \vdash \phi : \mathbb{C} \quad \sup I < \infty}{\Gamma \vdash \Diamond_I \phi : \mathbb{C}} \Diamond^{\mathbb{C}} \qquad \frac{\vdash \phi : \mathsf{PG}(x)^{-} \quad \Gamma \vdash \phi : \mathbb{C}}{\Gamma \vdash \forall x.\, \phi : \mathbb{C}} \forall^{\mathbb{C}}$$

EMFOTL: Example typing

If delete is causable, then $\forall \mathrm{d}, \mathrm{u}.\ \mathsf{deletion\_request}\,(\mathrm{u}, \mathrm{d}) \rightarrow \Diamond_{[0,30]}\mathsf{delete}\,(\mathrm{d}) \in \mathrm{EMFOTL}$

EMFOTL: Example typing

If delete is causable, then $\forall d, u.\, \mathsf{deletion\_request}\,(u, d) \rightarrow \Diamond_{[0,30]}\mathsf{delete}\,(d) \in \text{EMFOTL}$

$$\frac{}{\{\mathsf{delete} \mapsto \mathbb{C}\} \vdash \forall d, u.\, \mathsf{deletion\_request}\,(d) \rightarrow \Diamond_{[0,30]}\mathsf{delete}\,(d) : \mathbb{C}}$$

Ξ_i: "past-guardedness" proofs – see paper for details

EMFOTL: Example typing

If delete is causable, then $\forall d, u.\ \mathsf{deletion\_request}(u, d) \rightarrow \Diamond_{[0,30]}\mathsf{delete}(d) \in \text{EMFOTL}$

$$\dfrac{\Xi_1 \qquad \overline{\{\mathsf{delete} \mapsto \mathbb{C}\} \vdash \forall u.\ \mathsf{deletion\_request}(u, d) \rightarrow \Diamond_{[0,30]}\mathsf{delete}(d) : \mathbb{C}}}{\{\mathsf{delete} \mapsto \mathbb{C}\} \vdash \forall d, u.\ \mathsf{deletion\_request}(d) \rightarrow \Diamond_{[0,30]}\mathsf{delete}(d) : \mathbb{C}}\ \forall^{\mathbb{C}}$$

Ξ_i: "past-guardedness" proofs – see paper for details

EMFOTL: Example typing

If delete is causable, then $\forall d, u.\ \mathsf{deletion\_request}\,(u, d) \longrightarrow \Diamond_{[0,30]}\mathsf{delete}\,(d) \in \text{EMFOTL}$

$$\dfrac{\Xi_1 \qquad \dfrac{\Xi_2 \qquad \overline{\{\mathsf{delete} \mapsto \mathbb{C}\} \vdash \mathsf{deletion\_request}\,(u, d) \longrightarrow \Diamond_{[0,30]}\mathsf{delete}\,(d) : \mathbb{C}}}{\{\mathsf{delete} \mapsto \mathbb{C}\} \vdash \forall u.\ \mathsf{deletion\_request}\,(u, d) \longrightarrow \Diamond_{[0,30]}\mathsf{delete}\,(d) : \mathbb{C}}\ \forall^{\mathbb{C}}}{\{\mathsf{delete} \mapsto \mathbb{C}\} \vdash \forall d, u.\ \mathsf{deletion\_request}\,(d) \longrightarrow \Diamond_{[0,30]}\mathsf{delete}\,(d) : \mathbb{C}}\ \forall^{\mathbb{C}}$$

Ξ_i: “past-guardedness” proofs – see paper for details

EMFOTL: Example typing

If delete is causable, then $\forall \mathrm{d}, \mathrm{u}.\ \mathsf{deletion\_request}\,(\mathrm{u}, \mathrm{d}) \rightarrow \Diamond_{[0,30]} \mathsf{delete}\,(\mathrm{d}) \in \text{EMFOTL}$

$$
\dfrac{\Xi_1 \quad \dfrac{\Xi_2 \quad \dfrac{\dfrac{}{\{\mathsf{delete} \mapsto \mathbb{C}\} \vdash \Diamond_{[0,30]}\mathsf{delete}(\mathrm{d}) : \mathbb{C}}}{\{\mathsf{delete} \mapsto \mathbb{C}\} \vdash \mathsf{deletion\_request}\,(\mathrm{u}, \mathrm{d}) \rightarrow \Diamond_{[0,30]}\mathsf{delete}\,(\mathrm{d}) : \mathbb{C}} \rightarrow^{\mathbb{C}\mathrm{R}}}{\{\mathsf{delete} \mapsto \mathbb{C}\} \vdash \forall \mathrm{u}.\ \mathsf{deletion\_request}\,(\mathrm{u}, \mathrm{d}) \rightarrow \Diamond_{[0,30]}\mathsf{delete}\,(\mathrm{d}) : \mathbb{C}} \forall^{\mathbb{C}}}{\{\mathsf{delete} \mapsto \mathbb{C}\} \vdash \forall \mathrm{d}, \mathrm{u}.\ \mathsf{deletion\_request}\,(\mathrm{d}) \rightarrow \Diamond_{[0,30]}\mathsf{delete}\,(\mathrm{d}) : \mathbb{C}} \forall^{\mathbb{C}}
$$

Ξ_i: “past-guardedness” proofs – see paper for details

EMFOTL: Example typing

If delete is causable, then $\forall d, u.\ \mathsf{deletion\_request}\,(u, d) \longrightarrow \Diamond_{[0,30]}\mathsf{delete}\,(d) \in \mathrm{EMFOTL}$

$$
\dfrac{\Xi_1 \qquad \dfrac{\Xi_2 \qquad \dfrac{\dfrac{\{\mathsf{delete} \mapsto \mathbb{C}\} \vdash \mathsf{delete}(d) : \mathbb{C} \qquad 30 < \infty}{\{\mathsf{delete} \mapsto \mathbb{C}\} \vdash \Diamond_{[0,30]}\mathsf{delete}(d) : \mathbb{C}}\,\Diamond^{\mathbb{C}}}{\{\mathsf{delete} \mapsto \mathbb{C}\} \vdash \mathsf{deletion\_request}\,(u, d) \longrightarrow \Diamond_{[0,30]}\mathsf{delete}\,(d) : \mathbb{C}}\,\longrightarrow^{\mathbb{C}R}}{\{\mathsf{delete} \mapsto \mathbb{C}\} \vdash \forall u.\ \mathsf{deletion\_request}\,(u, d) \longrightarrow \Diamond_{[0,30]}\mathsf{delete}\,(d) : \mathbb{C}}\,\forall^{\mathbb{C}}}{\{\mathsf{delete} \mapsto \mathbb{C}\} \vdash \forall d, u.\ \mathsf{deletion\_request}\,(d) \longrightarrow \Diamond_{[0,30]}\mathsf{delete}\,(d) : \mathbb{C}}\,\forall^{\mathbb{C}}
$$

Ξ_i: "past-guardedness" proofs – see paper for details

EMFOTL: Example typing

If delete is causable, then $\forall d, u.\ \mathsf{deletion\_request}\,(u, d) \rightarrow \Diamond_{[0,30]}\mathsf{delete}\,(d) \in \mathrm{EMFOTL}$

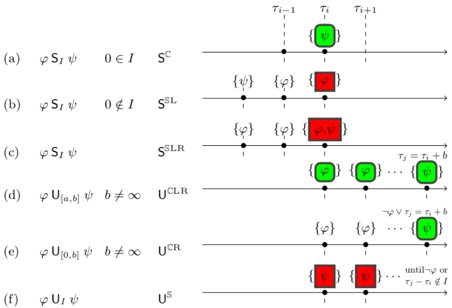
$$\dfrac{\Xi_1 \qquad \dfrac{\Xi_2 \qquad \dfrac{\dfrac{\dfrac{\mathsf{delete} \in \mathbb{C}}{\{\mathsf{delete} \mapsto \mathbb{C}\} \vdash \mathsf{delete}(d) : \mathbb{C}}\,\mathbb{E}^{\mathbb{C}} \qquad 30 < \infty}{\{\mathsf{delete} \mapsto \mathbb{C}\} \vdash \Diamond_{[0,30]}\mathsf{delete}(d) : \mathbb{C}}\,\Diamond^{\mathbb{C}}}{\{\mathsf{delete} \mapsto \mathbb{C}\} \vdash \mathsf{deletion\_request}\,(u, d) \rightarrow \Diamond_{[0,30]}\mathsf{delete}\,(d) : \mathbb{C}}\,\rightarrow^{\mathbb{C}R}}{\{\mathsf{delete} \mapsto \mathbb{C}\} \vdash \forall u.\ \mathsf{deletion\_request}\,(u, d) \rightarrow \Diamond_{[0,30]}\mathsf{delete}\,(d) : \mathbb{C}}\,\forall^{\mathbb{C}}}{\{\mathsf{delete} \mapsto \mathbb{C}\} \vdash \forall d, u.\ \mathsf{deletion\_request}\,(d) \rightarrow \Diamond_{[0,30]}\mathsf{delete}\,(d) : \mathbb{C}}\,\forall^{\mathbb{C}}$$

Ξ_i: "past-guardedness" proofs – see paper for details

Enforcement algorithm

In every step i with timestamp τ:

- The algorithm starts with a goal Φ
  - If i = 0, this goal is just the target policy
- By decomposing the goal into simpler goals, it computes:
  - $C \subseteq \mathbb{C} \times \mathbb{D}^*$: events to cause
  - $S \subseteq \mathbb{S} \times \mathbb{D}^*$: events to suppress
  - X': new set of obligations for the next time-point



Enforcement algorithm: Correctness

An enforcer $\mathcal{E}$ is *sound* with respect to a formula φ iff for any trace σ, we have $\models_{\mathcal{E}(\sigma)} \varphi$.
It is *transparent* with respect to φ iff for all σ with $\models_\sigma \varphi$, then $\mathcal{E}(\sigma) = \sigma$.

Theorem: Soundness

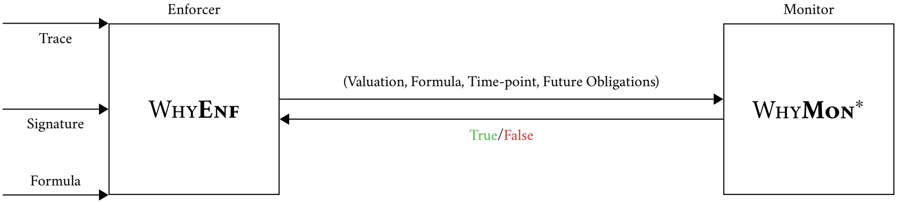
If $\varphi \in \mathrm{EMFOTL}$*, the enforcer* $\mathcal{E}_\varphi$ *is sound.*

Theorem: Transparency

If $\varphi \in \mathrm{TEMFOTL}^*$*, the enforcer* $\mathcal{E}_\varphi$ *is transparent.*

* see definition in the extended version of our paper

Implementation



where WhyMon* is a modified version of WhyMon [Lima et al., TACAS 2024] which

- returns Boolean verdicts instead of explanations
- includes a function Sat that checks if a valuation satisfies a formula on a trace prefix given some future obligations

Evaluation

Dataset:

- MFOTL formalization of core GDPR provisions [Arfelt et al., 2019]
- Traces produced by a real-world system [Debois et al., 2015] with ~4,000 time-points
- Random synthetic traces with length 100-25600 and time-point sizes 1–256

Research questions:

RQ1. Is EMFOTL expressive enough to formalize real-world policies?
Is manual formula rewriting necessary?

RQ2. At what maximum event rate can WhyEnf perform real-time enforcement?

RQ3. Do WhyEnf's performance improve upon the state-of-the-art?

Evaluation – RQ1 (Expressiveness)

Minimization: $\square(\forall c, d, u.\ \mathsf{collect}(c, d, u) \rightarrow \Diamond \mathsf{use}(c, d, u))$

Limitation: $\square(\forall c, d, u.\ \mathsf{collect}(c, d, u) \rightarrow \Diamond_{[0,b]} \mathsf{delete}(c, d, u))$

Lawfulness: $\square(\forall c, d, u.\ \mathsf{use}(c, d, u) \rightarrow \blacklozenge(\mathsf{consent}(u, c) \vee \mathsf{legal\_grounds}(u, d)))$

Consent: $\square(\forall c, d, u.\ \mathsf{use}(c, d, u) \rightarrow (\blacklozenge \mathsf{legal\_grounds}(u, d)) \vee (\neg \mathsf{revoke}(u, c)\ \mathcal{S}\ \mathsf{consent}(u, c)))$

Information: $\square(\forall c, d, u.\ \mathsf{collect}(c, d, u) \rightarrow ((\bigcirc \mathsf{inform}(u)) \vee (\blacklozenge \mathsf{inform}(u))))$

Deletion: $\square(\forall c, d, u.\ \mathsf{deletion\_request}(c, d, u) \rightarrow \Diamond_{[0,30]} \mathsf{delete}(c, d, u))$

Sharing: $\square(\forall c, d, u, p.\ \mathsf{deletion\_request}(c, d, u) \wedge (\blacklozenge \mathsf{share}(p, d)) \rightarrow \Diamond_{[0,30]} \mathsf{notify}(p, d))$

Evaluation – RQ1 (Expressiveness)

✗ **Minimization**: $\square(\forall c, d, u.\, \mathsf{collect}(c, d, u) \to \Diamond \mathsf{use}(c, d, u))$ **inherently not enforceable!**

Limitation: $\square(\forall c, d, u.\, \mathsf{collect}(c, d, u) \to \Diamond_{[0,b]} \mathsf{delete}(c, d, u))$

Lawfulness: $\square(\forall c, d, u.\, \mathsf{use}(c, d, u) \to \blacklozenge(\mathsf{consent}(u, c) \lor \mathsf{legal\_grounds}(u, d)))$

Consent: $\square(\forall c, d, u.\, \mathsf{use}(c, d, u) \to (\blacklozenge \mathsf{legal\_grounds}(u, d)) \lor (\neg \mathsf{revoke}(u, c)\ \mathcal{S}\ \mathsf{consent}(u, c)))$

Information: $\square(\forall c, d, u.\, \mathsf{collect}(c, d, u) \to ((\bigcirc \mathsf{inform}(u)) \lor (\blacklozenge \mathsf{inform}(u))))$

Deletion: $\square(\forall c, d, u.\, \mathsf{deletion\_request}(c, d, u) \to \Diamond_{[0,30]} \mathsf{delete}(c, d, u))$

Sharing: $\square(\forall c, d, u, p.\, \mathsf{deletion\_request}(c, d, u) \land (\blacklozenge \mathsf{share}(p, d)) \to \Diamond_{[0,30]} \mathsf{notify}(p, d))$

Evaluation – RQ1 (Expressiveness)

✗ **Minimization**: $\square(\forall c, d, u.\ \mathsf{collect}(c, d, u) \rightarrow \lozenge \mathsf{use}(c, d, u))$

✓ **Limitation**: $\square(\forall c, d, u.\ \mathsf{collect}(c, d, u) \rightarrow \lozenge_{[0,b]} \mathsf{delete}(c, d, u))$

✓ **Lawfulness**: $\square(\forall c, d, u.\ \mathsf{use}(c, d, u) \rightarrow \blacklozenge(\mathsf{consent}(u, c) \vee \mathsf{legal\_grounds}(u, d)))$

✓ **Consent**: $\square(\forall c, d, u.\ \mathsf{use}(c, d, u) \rightarrow (\blacklozenge \mathsf{legal\_grounds}(u, d)) \vee (\neg \mathsf{revoke}(u, c)\ \mathcal{S}\ \mathsf{consent}(u, c)))$

✓ **Information**: $\square(\forall c, d, u.\ \mathsf{collect}(c, d, u) \rightarrow ((\bigcirc \mathsf{inform}(u)) \vee (\blacklozenge \mathsf{inform}(u))))$

✓ **Deletion**: $\square(\forall c, d, u.\ \mathsf{deletion\_request}(c, d, u) \rightarrow \lozenge_{[0,30]} \mathsf{delete}(c, d, u))$

✓ **Sharing**: $\square(\forall c, d, u, p.\ \mathsf{deletion\_request}(c, d, u) \wedge (\blacklozenge \mathsf{share}(p, d)) \rightarrow \lozenge_{[0,30]} \mathsf{notify}(p, d))$

transparently enforceable, no policy rewriting needed!

Evaluation – RQ2+3 (Performance)

Real-time condition: $\mathsf{max}_\ell(a) \leq 1/a$.

Event rate (avg_{er}, s^{-1}) and maximum latency (max_ℓ, ms) for the largest real-time acceleration a.

	WhyEnf Enforcer		WhyMon* Monitor		EnfPoly Enforcer	
Policy	avg_{er}	max_ℓ	avg_{er}	max_ℓ	avg_{er}	max_ℓ
Limitation	**632**	14	not supported		not supported	
Lawfulness	405	15	405	12	**6479**	1.0
Consent	51	96	101	51	**6479**	1.0
Information	202	13	**405**	16	not supported	
Deletion	**632**	19	13	434	not supported	
Sharing	**202**	26	13	289	not supported	

Consistent findings on synthetic traces

- WhyEnf2
  - Language extensions: let bindings, complex terms, aggregations
  - Performance optimizations

- Using WhyEnf as a backend for enforcing legal requirements in software
  - Instrument of web applications
  - Domain-specific language for legal specs

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`
Leonardo Lima	`leonardo@di.ku.dk`

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,