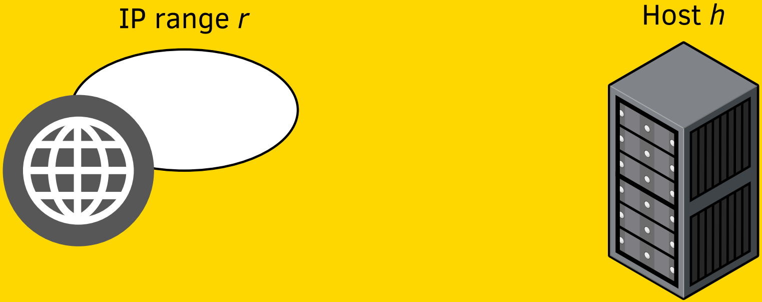
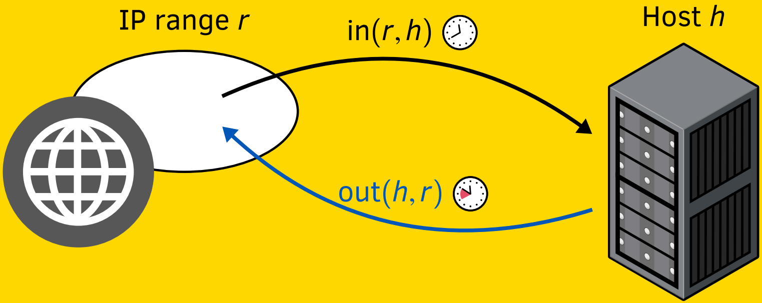
Verified First-Order Monitoring with Recursive Rules

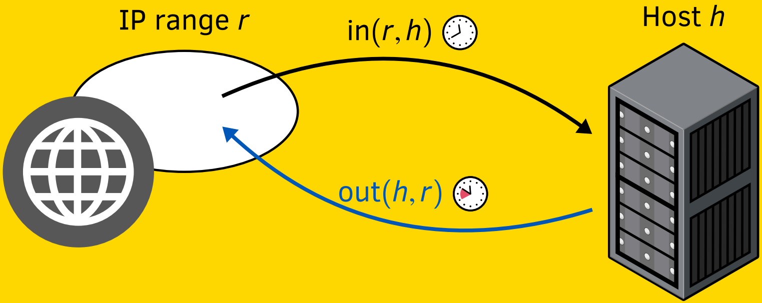
Sheila Zingg Srđan Krstić Martin Raszyk **Joshua Schneider** **Dmitriy Traytel**

ETH zürich

UNIVERSITY OF COPENHAGEN


IP range *r*
Host *h*


IP range r
in(r, h)
Host h
out(h, r)



previous | once/eventually in the past (non-strict)

$$\mathrm{out}(h,r) \wedge \underbrace{\bullet}_{\text{previous}} \underbrace{\blacklozenge}_{\text{once/eventually in the past (non-strict)}} \mathrm{in}(r,h)$$

Monitoring

Log:

```
10:29   in (DE, Webserver)
10:29   out(Laptop1, US)
10:31   in (CH, VPN)
10:33   out(Webserver, DE)
```

Monitoring

Policy: $\neg \exists h, r.\ \mathrm{out}(h, r) \wedge \bullet \blacklozenge\, \mathrm{in}(r, h)$

Log:

```
10:29   in (DE, Webserver)
10:29   out(Laptop1, US)
10:31   in (CH, VPN)
10:33   out(Webserver, DE)
```

VeriMon

DejaVu

MONPOLY

...

Monitoring

Policy: $\neg\exists h, r.\ \mathrm{out}(h, r) \wedge \bullet \blacklozenge \mathrm{in}(r, h)$

Log:

10:29	in (DE, Webserver)
10:29	out(Laptop1, US)
10:31	in (CH, VPN)
10:33	out(Webserver, DE)

VeriMon

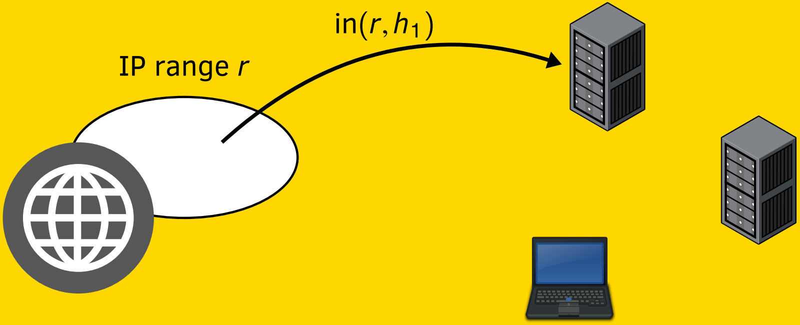
DejaVu

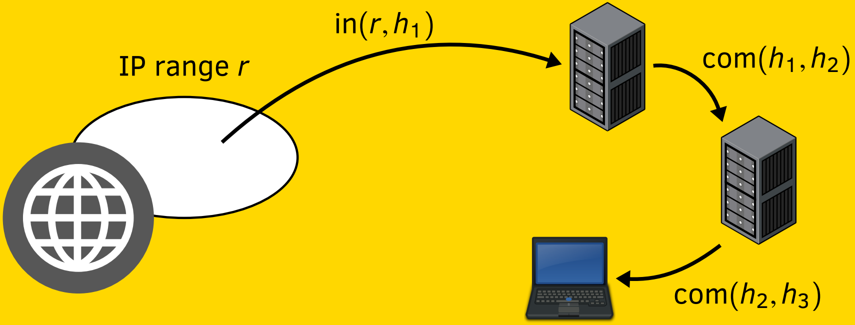
MONPOLY

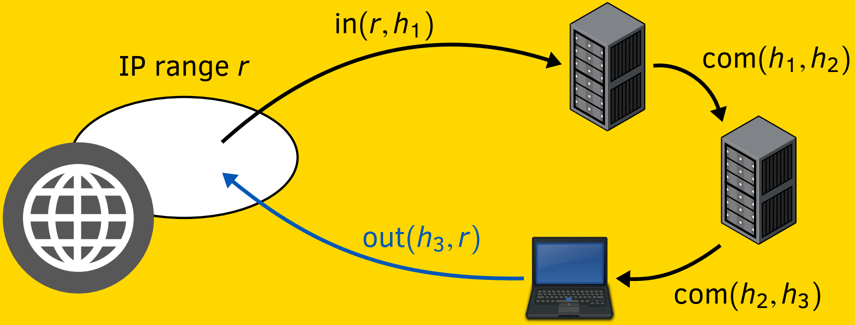
...

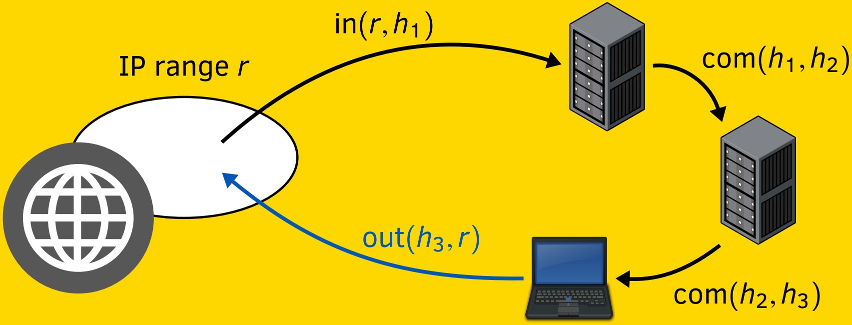
Output:

10:29	✓
10:29	✓
10:31	✓
10:33	✗

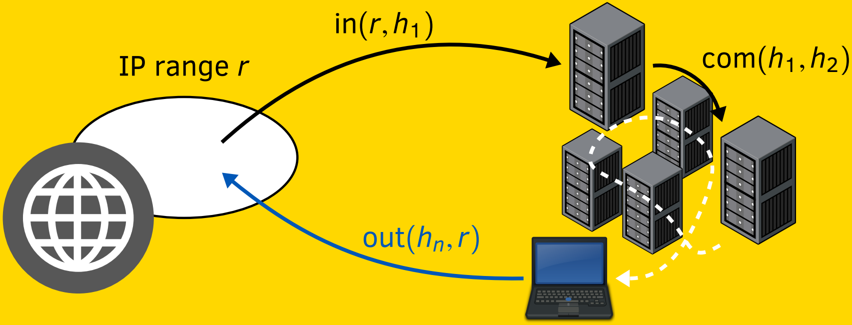

$\mathrm{in}(r, h_1)$
IP range r


IP range r
$\mathrm{in}(r, h_1)$
$\mathrm{com}(h_1, h_2)$
$\mathrm{com}(h_2, h_3)$

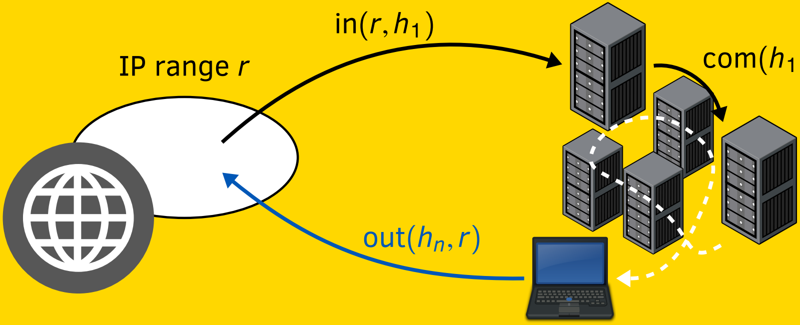

in(r, h_1)
IP range r
com(h_1, h_2)
out(h_3, r)
com(h_2, h_3)



$$\neg \exists h_1, h_2, h_3, r.\ \mathrm{out}(h_3, r) \land \CIRCLE \blacklozenge \big(\mathrm{com}(h_2, h_3) \land \CIRCLE \blacklozenge (\mathrm{com}(h_1, h_2) \land \CIRCLE \blacklozenge \mathrm{in}(r, h_1))\big)$$



in(r, h_1)
IP range r
com(h_1, h_2)
out(h_n, r)



$$
\begin{aligned}
\neg\exists h_n, r.\quad & (\mathrm{out}(h_n, r) \land \blacklozenge\!\!\bullet\, \mathrm{in}(r, h_n)) \\
& \lor (\exists h_1.\, \mathrm{out}(h_n, r) \land \bullet\blacklozenge(\mathrm{com}(h_1, h_n) \land \bullet\blacklozenge\, \mathrm{in}(r, h_1))) \\
& \lor \cdots \\
& \lor (\exists h_1 \ldots h_{n-1}.\, \mathrm{out}(h_n, r) \land \bullet\blacklozenge(\cdots(\mathrm{com}(h_1, h_2) \land \bullet\blacklozenge\, \mathrm{in}(r, h_1))\cdots)
\end{aligned}
$$

PFLTL with rules to the rescue!

Past-time First-order Linear Temporal Logic, implemented by DejaVu

PFLTL with rules to the rescue!

Past-time First-order Linear Temporal Logic, implemented by DejaVu

$$\neg\exists h, r.\ \mathrm{out}(h, r) \land \mathrm{taint}(r, h)$$

where

$$\begin{aligned}\mathrm{taint}(r, h) \coloneqq\ & \mathrm{in}(r, h) \\ & \lor (\bullet\, \mathrm{taint}(r, h)) \\ & \lor (\exists h'.\ (\bullet\, \mathrm{taint}(r, h')) \land \mathrm{com}(h', h))\end{aligned}$$

PFLTL with rules to the rescue!

Past-time First-order Linear Temporal Logic, implemented by DejaVu

$$\neg \exists h, r.\ \mathrm{out}(h, r) \wedge \mathrm{taint}(r, h)$$

where

$$\begin{aligned} \mathrm{taint}(r, h) \coloneqq{} & \mathrm{in}(r, h) \\ & \vee (\bullet\, \mathrm{taint}(r, h)) \\ & \vee (\exists h'.\ (\bullet\, \mathrm{taint}(r, h')) \wedge \mathrm{com}(h', h)) \end{aligned}$$

“Temporally-directed transitive closure”

PFLTL with rules to the rescue?

Past-time First-order Linear Temporal Logic

$\neg \exists h, r.\ \mathrm{out}(h, r) \wedge \mathrm{taint}(r, h)$

eventually in the future

where

$$\begin{aligned}\mathrm{taint}(r,h) \coloneqq\ & \mathrm{in}(r,h) \wedge (\Diamond_{[0,1h]}\mathrm{ids}(h)) \\ & \vee (\bullet\, \mathrm{taint}(r,h)) \\ & \vee (\exists h'.\ (\bullet\, \mathrm{taint}(r,h')) \wedge \mathrm{com}(h',h)) \wedge (\Diamond_{[0,1h]}\mathrm{ids}(h))\end{aligned}$$

Our paper: MFOTL meets recursive rules

Metric First-Order Temporal Logic = PFLTL + future operators + aggregations

Our paper: MFOTL meets recursive rules

Metric First-Order Temporal Logic = PFLTL + future operators + aggregations

Recursive let operator in MFOTL

- Semantics only considers recursive occurrences strictly in the past
- Allows efficient evaluation (no fixpoint iteration)
- Syntactic fragment: every recursive occurrence has a past guard

Our paper: MFOTL meets recursive rules

Metric First-Order Temporal Logic = PFLTL + future operators + aggregations

Recursive let operator in MFOTL

- Semantics only considers recursive occurrences strictly in the past
- Allows efficient evaluation (no fixpoint iteration)
- Syntactic fragment: every recursive occurrence has a past guard

Implementation and Evaluation

- Part of the VeriMon monitor
- Formally verified in Isabelle HOL
- Isabelle (40 kLOC) ⟶ OCaml (11 kLOC)
- Compared with DejaVu on PFLTL

Semantics

$$\sigma, v, i \models \varphi$$

Semantics

formula

$$\sigma, v, i \models \varphi$$

Semantics

formula

$$\sigma, v, i \models \varphi$$

database stream

Semantics

variable assignment

formula

$$\sigma, v, i \models \varphi$$

database stream

Semantics

variable assignment

formula

$$\sigma, v, i \models \varphi$$

database stream

time-point

Semantics

variable assignment

formula

$$\sigma, v, i \models \varphi$$

database stream

time-point

$$\text{eval}\ \sigma\ j\ \varphi \approx \{v \mid \sigma, v, j \models \varphi\}$$

Semantics

variable assignment   formula

$$\sigma, v, i \models \varphi$$

database stream   time-point

$$\text{eval } \sigma\ j\ \varphi \approx \{v \mid \sigma, v, j \models \varphi\}$$

$$\sigma, v, i \models p(x_1, \ldots, x_n) \quad \Longleftrightarrow \quad (v(x_1), \ldots, v(x_n)) \in \sigma_i(p)$$

table p in ith database

Semantics

variable assignment

formula

$$\sigma, v, i \vDash \varphi$$

database stream

time-point

eval $\sigma\ j\ \varphi \approx \{v \mid \sigma, v, j \vDash \varphi\}$

$$\sigma, v, i \vDash p(x_1, \ldots, x_n) \iff (v(x_1), \ldots, v(x_n)) \in \sigma_i(p)$$

$$\vdots$$

$$\sigma, v, i \vDash \bullet\, \alpha \iff i > 0 \;\wedge\; \sigma, v, i-1 \vDash \alpha$$

$$\sigma, v, i \vDash \alpha \,\mathsf{S}\, \beta \iff \exists j \leq i.\ \sigma, v, j \vDash \beta \;\wedge\; (\forall k \in \{j <.. i\}.\ \sigma, v, k \vDash \alpha)$$

$$\vdots$$

Semantics

variable assignment

formula

$$\sigma, v, i \models \varphi$$

database stream

time-point

eval $\sigma\ j\ \varphi \approx \{v \mid \sigma, v, j \models \varphi\}$

$$\sigma, v, i \models p(x_1, \ldots, x_n) \iff (v(x_1), \ldots, v(x_n)) \in \sigma_i(p)$$

$$\vdots$$

$$\sigma, v, i \models \bullet\, \alpha \iff i > 0 \ \wedge\ \sigma, v, i-1 \models \alpha$$

$$\sigma, v, i \models \alpha \mathrel{\mathsf{S}} \beta \iff \exists j \leq i.\ \sigma, v, j \models \beta \ \wedge\ (\forall k \in \{j <.. i\}.\ \sigma, v, k \models \alpha)$$

$$\vdots$$

$$\sigma, v, i \models \textbf{LetPast}\ p := \alpha\ \textbf{in}\ \beta \iff \qquad , v, i \models \beta$$

Semantics

variable assignment

formula

$$\sigma, v, i \models \varphi$$

database stream

time-point

eval $\sigma\ j\ \varphi \approx \{v \mid \sigma, v, j \models \varphi\}$

$$\sigma, v, i \models p(x_1, \dots, x_n) \iff (v(x_1), \dots, v(x_n)) \in \sigma_i(p)$$

$$\vdots$$

$$\sigma, v, i \models \bullet\, \alpha \iff i > 0 \ \wedge\ \sigma, v, i-1 \models \alpha$$

$$\sigma, v, i \models \alpha \,\mathsf{S}\, \beta \iff \exists j \leq i.\ \sigma, v, j \models \beta \ \wedge\ (\forall k \in \{j <.. i\}.\ \sigma, v, k \models \alpha)$$

$$\vdots$$

$$\sigma, v, i \models \textbf{LetPast}\ p := \alpha\ \textbf{in}\ \beta \iff \sigma[p \Rrightarrow \quad], v, i \models \beta$$

Semantics

variable assignment

formula

$$\sigma, v, i \models \varphi$$

database stream

time-point

$\text{eval}\ \sigma\ j\ \varphi \approx \{v \mid \sigma, v, j \models \varphi\}$

$$\sigma, v, i \models p(x_1, \ldots, x_n) \iff (v(x_1), \ldots, v(x_n)) \in \sigma_i(p)$$

$$\vdots$$

$$\sigma, v, i \models \bullet\, \alpha \iff i > 0 \ \land\ \sigma, v, i-1 \models \alpha$$

$$\sigma, v, i \models \alpha\ \mathsf{S}\ \beta \iff \exists j \le i.\ \sigma, v, j \models \beta \ \land\ (\forall k \in \{j <.. i\}.\ \sigma, v, k \models \alpha)$$

$$\vdots$$

$$\sigma, v, i \models \textbf{LetPast}\ p := \alpha\ \textbf{in}\ \beta \iff \sigma[p \Rrightarrow \lambda j.\ \text{eval}\ \sigma\ j\ \alpha], v, i \models \beta$$

Semantics

variable assignment

formula

$$\sigma, v, i \models \varphi$$

database stream

time-point

$$\text{eval}\ \sigma\ j\ \varphi \approx \{v \mid \sigma, v, j \models \varphi\}$$

$$\sigma, v, i \models p(x_1, \ldots, x_n) \iff (v(x_1), \ldots, v(x_n)) \in \sigma_i(p)$$

$$\vdots$$

$$\sigma, v, i \models \bullet\, \alpha \iff i > 0 \ \wedge\ \sigma, v, i - 1 \models \alpha$$

$$\sigma, v, i \models \alpha\ \mathsf{S}\ \beta \iff \exists j \leq i.\ \sigma, v, j \models \beta \ \wedge\ (\forall k \in \{j <.. i\}.\ \sigma, v, k \models \alpha)$$

$$\vdots$$

$$\sigma, v, i \models \textbf{LetPast}\ p \coloneqq \alpha\ \textbf{in}\ \beta \iff \sigma[p \Rrightarrow \text{recp}\ (\lambda R\, j.\ \text{eval}\ (\sigma[p \Rrightarrow R])\ j\ \alpha)], v, i \models \beta$$

Semantics

variable assignment

formula

$$\sigma, v, i \models \varphi$$

database stream

time-point

$$\text{eval}\ \sigma\ j\ \varphi \approx \{v \mid \sigma, v, j \models \varphi\}$$

$$\sigma, v, i \models p(x_1, \ldots, x_n) \iff (v(x_1), \ldots, v(x_n)) \in \sigma_i(p)$$

$$\vdots$$

$$\sigma, v, i \models \bullet\, \alpha \iff i > 0 \ \wedge\ \sigma, v, i-1 \models \alpha$$

$$\sigma, v, i \models \alpha\ \mathsf{S}\ \beta \iff \exists j \le i.\ \sigma, v, j \models \beta \ \wedge\ (\forall k \in \{j <.. i\}.\ \sigma, v, k \models \alpha)$$

$$\vdots$$

$$\sigma, v, i \models \textbf{LetPast}\ p := \alpha\ \textbf{in}\ \beta \iff \sigma[p \Rrightarrow \text{recp}\ (\lambda R\ j.\ \text{eval}\ (\sigma[p \Rrightarrow R])\ j\ \alpha)], v, i \models \beta$$

$$\text{recp}\ f\ i = f\ (\lambda j.\ \underline{\text{if}}\ j < i\ \underline{\text{then}}\ \text{recp}\ f\ j\ \underline{\text{else}}\ \{\})\ i$$

A picture for

$\varphi \triangleq$ **LetPast** $s \coloneqq b \lor (a \land \bullet s)$ **in** s

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
φ					?

A picture for

$\varphi \triangleq$ **LetPast** $s \coloneqq b \lor (a \land \bullet s)$ **in** s

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
φ					

$\text{eval}\ (\sigma[s \Rrightarrow \text{recp}\ f])\ 4\ s$

$\lambda R\ j.\ \text{eval}\ (\sigma[s \Rrightarrow R])\ j\ (b \lor (a \land \bullet s))$

A picture for

$$\varphi \triangleq \textbf{LetPast}\ s \coloneqq b \vee (a \wedge \bullet s)\ \textbf{in}\ s$$

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	recp f 0	recp f 1	recp f 2	recp f 3	recp f 4
φ					eval $(\sigma[s \Rrightarrow \text{recp } f])$ 4 s

$\lambda R\ j.\ \text{eval}\ (\sigma[s \Rrightarrow R])\ j\ (b \vee (a \wedge \bullet s))$

A picture for

$\varphi \triangleq$ **LetPast** $s := b \vee (a \wedge \bullet s)$ **in** s

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	recp f 0	recp f 1	recp f 2	recp f 3	recp f 4
φ					eval $(\sigma[s \Rrightarrow \text{recp } f])$ 4 s

$\lambda R\, j.$ eval $(\sigma[s \Rrightarrow R])\ j\ (b \vee (a \wedge \bullet s))$

$$\text{recp } f\ i = f\ (\lambda j.\ \underline{\text{if}}\ j < i\ \underline{\text{then}}\ \text{recp } f\ j\ \underline{\text{else}}\ \{\,\})\ i$$

A picture for

$\varphi \triangleq$ **LetPast** $s \coloneqq b \vee (a \wedge \bullet s)$ **in** s

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	recp f 0	recp f 1	recp f 2	recp f 3	recp f 4
φ					eval $(\sigma[s \Rrightarrow \text{recp } f])$ 4 s

$\lambda R\, j.$ eval $(\sigma[s \Rrightarrow R])\; j\; (b \vee (a \wedge \bullet s))$

recp $f\; i = f\; (\lambda j.\ \underline{\text{if}}\ j < i\ \underline{\text{then}}\ \text{recp } f\ j\ \underline{\text{else}}\ ✗\ )\; i$

A picture for

$\varphi \triangleq$ **LetPast** $s \coloneqq b \lor (a \land \bullet\, s)$ **in** s

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	$\mathsf{recp}\ f\ 0$	$\mathsf{recp}\ f\ 1$	$\mathsf{recp}\ f\ 2$	$\mathsf{recp}\ f\ 3$	$\mathsf{recp}\ f\ 4$

$\mathsf{eval}\ (\sigma[s \Rrightarrow \ldots])\ \boxed{0}\ (b \lor (a \land \bullet\, s))$

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	✗	✗	✗	✗	✗

$\mathsf{eval}\ (\sigma[s \Rrightarrow \mathsf{recp}\ f])\ 4\ s$

$\lambda R\, j.\ \mathsf{eval}\ (\sigma[s \Rrightarrow R])\ j\ (b \lor (a \land \bullet\, s))$

$\mathsf{recp}\ f\ i = f\ (\lambda j.\ \underline{\mathsf{if}}\ j < i\ \underline{\mathsf{then}}\ \mathsf{recp}\ f\ j\ \underline{\mathsf{else}}\ ✗\ )\ i$

A picture for

$\varphi \triangleq$ **LetPast** $s := b \vee (a \wedge \bullet s)$ **in** s

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	✓	recp f 1	recp f 2	recp f 3	recp f 4

eval $(\sigma[s \Rrightarrow \ldots])$ $\boxed{0}$ $(b \vee (a \wedge \bullet s))$

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	✗	✗	✗	✗	✗

eval $(\sigma[s \Rrightarrow \text{recp } f])$ 4 s

$\lambda R\, j.$ eval $(\sigma[s \Rrightarrow R])$ j $(b \vee (a \wedge \bullet s))$

recp f $i = f$ $(\lambda j.$ $\underline{\text{if}}$ $j < i$ $\underline{\text{then}}$ recp f j $\underline{\text{else}}$ ✗ $)$ i

A picture for

$\varphi \triangleq$ **LetPast** $s \coloneqq b \lor (a \land \bullet s)$ **in** s

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	✓	recp f 1	recp f 2	recp f 3	recp f 4

eval $(\sigma[s \Rrightarrow \ldots])$ 1 $(b \lor (a \land \bullet s))$

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	recp f 0	✗	✗	✗	✗

eval $(\sigma[s \Rrightarrow \text{recp } f])$ 4 s

$\lambda R\, j.$ eval $(\sigma[s \Rrightarrow R])\ j\ (b \lor (a \land \bullet s))$

recp $f\ i = f\ (\lambda j.\ \underline{\text{if}}\ j < i\ \underline{\text{then}}\ \text{recp}\ f\ j\ \underline{\text{else}}$ ✗ $)\ i$

A picture for
$\varphi \triangleq$ **LetPast** $s \coloneqq b \vee (a \wedge \bullet s)$ **in** s

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	✓	recp f 1	recp f 2	recp f 3	recp f 4

eval $(\sigma[s \Rrightarrow \ldots])$ $\boxed{1}$ $(b \vee (a \wedge \bullet s))$

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	✓	✗	✗	✗	✗

eval $(\sigma[s \Rrightarrow$ recp $f])$ 4 s

$\lambda R\, j.$ eval $(\sigma[s \Rrightarrow R])\; j\; (b \vee (a \wedge \bullet s))$

recp $f\ i = f\ (\lambda j.\ \underline{\text{if}}\ j < i\ \underline{\text{then}}$ recp $f\ j\ \underline{\text{else}}$ ✗ $)\ i$

A picture for

$\varphi \triangleq \textbf{LetPast}\ s \coloneqq b \lor (a \land \bullet s)\ \textbf{in}\ s$

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	✓	✗	recp f 2	recp f 3	recp f 4

$\text{eval}\ (\sigma[s \Rrightarrow \ldots])\ \boxed{1}\ (b \lor (a \land \bullet s))$

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	✓	✗	✗	✗	✗

$\text{eval}\ (\sigma[s \Rrightarrow \text{recp}\ f])\ 4\ s$

$\lambda R\, j.\ \text{eval}\ (\sigma[s \Rrightarrow R])\ j\ (b \lor (a \land \bullet s))$

$\text{recp}\ f\ i = f\ (\lambda j.\ \underline{\text{if}}\ j < i\ \underline{\text{then}}\ \text{recp}\ f\ j\ \underline{\text{else}}\ ✗\ )\ i$

A picture for

$\varphi \triangleq$ **LetPast** $s \coloneqq b \lor (a \land \bullet s)$ **in** s

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	✓	✗	recp f 2	recp f 3	recp f 4
φ					

eval $(\sigma[s \Rrightarrow \ldots])$ $\boxed{2}$ $(b \lor (a \land \bullet s))$

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	recp f 0	recp f 1	✗	✗	✗

eval $(\sigma[s \Rrightarrow \text{recp } f])$ 4 s

$\lambda R\, j.$ eval $(\sigma[s \Rrightarrow R])$ j $(b \lor (a \land \bullet s))$

recp f $i = f$ $(\lambda j.$ $\underline{\text{if}}$ $j < i$ $\underline{\text{then}}$ recp f j $\underline{\text{else}}$ ✗ $)$ i

A picture for

$\varphi \triangleq \textbf{LetPast}\ s := b \lor (a \land \bullet\, s)\ \textbf{in}\ s$

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	✓	✗	recp f 2	recp f 3	recp f 4
φ					

eval $(\sigma[s \Rrightarrow \ldots])$ $\boxed{2}$ $(b \lor (a \land \bullet\, s))$

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	✓	✗	✗	✗	✗

eval $(\sigma[s \Rrightarrow \text{recp } f])\ 4\ s$

$\lambda R\, j.\ \text{eval}\ (\sigma[s \Rrightarrow R])\ j\ (b \lor (a \land \bullet\, s))$

recp $f\ i = f\ (\lambda j.\ \underline{\text{if}}\ j < i\ \underline{\text{then}}\ \text{recp}\ f\ j\ \underline{\text{else}}$ ✗ $)\ i$

A picture for

$\varphi \triangleq$ **LetPast** $s \coloneqq b \vee (a \wedge \bullet s)$ **in** s

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	✓	✗	✓	recp f 3	recp f 4
φ					

eval $(\sigma[s \Rrightarrow \ldots])$ $\boxed{2}$ $(b \vee (a \wedge \bullet s))$

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	✓	✗	✗	✗	✗

eval $(\sigma[s \Rrightarrow \text{recp } f])$ 4 s

$\lambda R\, j.$ eval $(\sigma[s \Rrightarrow R])\ j\ (b \vee (a \wedge \bullet s))$

recp $f\ i = f\ (\lambda j.\ \underline{\text{if}}\ j < i\ \underline{\text{then}}\ \text{recp } f\ j\ \underline{\text{else}}$ ✗ $)\ i$

A picture for

$\varphi \triangleq$ **LetPast** $s := b \vee (a \wedge \bullet s)$ **in** s

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	✓	✗	✓	recp f 3	recp f 4
φ					

eval $(\sigma[s \Rrightarrow \ldots])$ 3 $(b \vee (a \wedge \bullet s))$

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	recp f 0	recp f 1	recp f 2	✗	✗

eval $(\sigma[s \Rrightarrow \text{recp } f])$ 4 s

$\Rrightarrow R])\ j\ (b \vee (a \wedge \bullet s))$

recp f i = f $(\lambda j.$ $\underline{\text{if}}$ $j < i$ $\underline{\text{then}}$ recp f j $\underline{\text{else}}$ ✗ $)$ i

A picture for

$\varphi \triangleq$ **LetPast** $s \coloneqq b \vee (a \wedge \bullet s)$ **in** s

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	✓	✗	✓	recp f 3	recp f 4
φ					eval $(\sigma[s \Rrightarrow \text{recp } f])$ 4 s

eval $(\sigma[s \Rrightarrow \ldots])$ 3 $(b \vee (a \wedge \bullet s))$

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	✓	✗	✓	✗	✗

$\Rrightarrow R])\ j\ (b \vee (a \wedge \bullet s))$

recp f i $=$ f $(\lambda j.$ <u>if</u> $j < i$ <u>then</u> recp f j <u>else</u> ✗ $)$ i

A picture for

$\varphi \triangleq$ **LetPast** $s := b \vee (a \wedge \bullet s)$ **in** s

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	✓	✗	✓	✓	recp f 4
φ					

eval $(\sigma[s \Rrightarrow \text{recp } f])$ 4 s

$\Rrightarrow R])\ j\ (b \vee (a \wedge \bullet s))$

eval $(\sigma[s \Rrightarrow \ldots])$ 3 $(b \vee (a \wedge \bullet s))$

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	✓	✗	✓	✗	✗

recp $f\ i = f\ (\lambda j.$ <u>if</u> $j < i$ <u>then</u> recp $f\ j$ <u>else</u> ✗ $)\ i$

A picture for

$\varphi \triangleq \textbf{LetPast}\ s \coloneqq b \lor (a \land \bullet s)\ \textbf{in}\ s$

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	✓	✗	✓	✓	recp f 4
φ					

ecp f]) 4 s

$\land \bullet s$))

eval $(\sigma[s \Rrightarrow \ldots])$ $\boxed{4}$ $(b \lor (a \land \bullet s))$

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	recp f 0	recp f 1	recp f 2	recp f 3	✗

recp f i = f $(\lambda j.$ <u>if</u> $j < i$ <u>then</u> recp f j <u>else</u> ✗ $)$ i

A picture for

$\varphi \triangleq \textbf{LetPast}\ s \coloneqq b \lor (a \land \bullet s)\ \textbf{in}\ s$

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	✓	✗	✓	✓	recp f 4
φ					

…ecp f]) 4 s

… $\land \bullet s$))

eval $(\sigma[s \Rrightarrow \ldots])$ 4 $(b \lor (a \land \bullet s))$

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	✓	✗	✓	✓	✗

recp f i = f $(\lambda j.$ <u>if</u> $j < i$ <u>then</u> recp f j <u>else</u> ✗ $)$ i

A picture for

$\varphi \triangleq$ **LetPast** $s \coloneqq b \lor (a \land \bullet s)$ **in** s

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	✓	✗	✓	✓	✓
φ					

eval $(\sigma[s \Rrightarrow \ldots])$ $\boxed{4}$ $(b \lor (a \land \bullet s))$

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	✓	✗	✓	✓	✗

recp f]) 4 s

$\land \bullet s))$

recp $f\ i = f\ (\lambda j.$ <u>if</u> $j < i$ <u>then</u> recp $f\ j$ <u>else</u> ✗ $)\ i$

A picture for

$\varphi \triangleq$ **LetPast** $s \coloneqq b \vee (a \wedge \bullet s)$ **in** s

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	✓	✗	✓	✓	✓
φ					

$\mathsf{eval}\ (\sigma[s \Rrightarrow \mathsf{recp}\ f])\ 4\ s$

$\lambda R\ j.\ \mathsf{eval}\ (\sigma[s \Rrightarrow R])\ j\ (b \vee (a \wedge \bullet s))$

$\mathsf{recp}\ f\ i = f\ (\lambda j.\ \underline{\mathsf{if}}\ j < i\ \underline{\mathsf{then}}\ \mathsf{recp}\ f\ j\ \underline{\mathsf{else}}$ ✗ $)\ i$

A picture for

$\varphi \triangleq$ **LetPast** $s \coloneqq b \vee (a \wedge \bullet s)$ **in** s

	0	1	2	3	4
a	✗	✗	✗	✓	✓
b	✓	✗	✓	✗	✗
s	✓	✗	✓	✓	✓
φ					✓

Past guards I

Our semantics for **LetPast** and the evaluation algorithm are well-behaved only for a subset of formulas.

Well-behaved $=$ can unfold the recursive predicate's definition iteratively;
evaluation makes progress.

Past guards I

Our semantics for **LetPast** and the evaluation algorithm are well-behaved only for a subset of formulas.

Well-behaved = can unfold the recursive predicate's definition iteratively;
evaluation makes progress.

	Formula	equivalent to	evaluation
Good:	$\textbf{LetPast}\ r(x) := p(x) \lor \bullet r(x)\ \textbf{in}\ r(x)$	$\blacklozenge p(x)$	works
Bad:	$\textbf{LetPast}\ r(x) := p(x) \lor \circ r(x)\ \textbf{in}\ r(x)$	$p(x)$	gets stuck

Well-behavedness is a semantic property that we approximate syntactically.

Past guards II

Approximation lattice: $\mathit{Unused} < \mathit{Past} < \mathit{NonFuture} < \mathit{Any}$

Past guards II

Approximation lattice: $\mathit{Unused} < \mathit{Past} < \mathit{NonFuture} < \mathit{Any}$

Status $⧗_p \, \varphi$ of predicate symbol p in formula φ:

Past guards II

Approximation lattice: $\mathit{Unused} < \mathit{Past} < \mathit{NonFuture} < \mathit{Any}$

Status $⌛_p\,\varphi$ of predicate symbol p in formula φ:

$$\begin{aligned} ⌛_p\, q(\ldots) &= (\textbf{if } p = q \textbf{ then } \mathit{NonFuture} \textbf{ else } \mathit{Unused}) \\ ⌛_p(\neg\alpha) &= ⌛_p\,\alpha \\ ⌛_p(\alpha \vee \beta) &= (⌛_p\,\alpha) \sqcup (⌛_p\,\beta) \qquad ⌛_p(\alpha \wedge \beta) = (⌛_p\,\alpha) \sqcup (⌛_p\,\beta) \end{aligned}$$

Past guards II

Approximation lattice: $\mathit{Unused} < \mathit{Past} < \mathit{NonFuture} < \mathit{Any}$

Status $⌛_p\,\varphi$ of predicate symbol p in formula φ:

$$
\begin{aligned}
⌛_p\, q(\ldots) &= (\textbf{if } p = q \textbf{ then } \mathit{NonFuture} \textbf{ else } \mathit{Unused}) \\
⌛_p(\neg\alpha) &= ⌛_p\,\alpha \\
⌛_p(\alpha \vee \beta) &= (⌛_p\,\alpha) \sqcup (⌛_p\,\beta) \qquad ⌛_p(\alpha \wedge \beta) = (⌛_p\,\alpha) \sqcup (⌛_p\,\beta) \\
⌛_p(\bullet_I\,\alpha) &= \mathit{Past} * ⌛_p\,\alpha
\end{aligned}
$$

$$
\begin{array}{cccll}
\mathit{Past} & * & \mathit{Unused} & = & \mathit{Unused} \\
\mathit{Past} & * & \mathit{Past} & = & \mathit{Past} \\
\mathit{Past} & * & \mathit{NonFuture} & = & \mathit{Past} \\
\mathit{Past} & * & \mathit{Any} & = & \mathit{Any}
\end{array}
$$

Past guards II

Approximation lattice: $\mathit{Unused} < \mathit{Past} < \mathit{NonFuture} < \mathit{Any}$

Status $\hourglass_p\, \varphi$ of predicate symbol p in formula φ:

$$\begin{aligned}
\hourglass_p\, q(\ldots) &= (\textbf{if } p = q \textbf{ then } \mathit{NonFuture} \textbf{ else } \mathit{Unused}) \\
\hourglass_p(\neg\alpha) &= \hourglass_p\, \alpha \\
\hourglass_p(\alpha \vee \beta) &= (\hourglass_p\, \alpha) \sqcup (\hourglass_p\, \beta) \qquad \hourglass_p(\alpha \wedge \beta) = (\hourglass_p\, \alpha) \sqcup (\hourglass_p\, \beta) \\
\hourglass_p(\bullet_I\, \alpha) &= \mathit{Past} * \hourglass_p\, \alpha \\
\hourglass_p(\bigcirc_I\, \alpha) &= \mathit{Any} * \hourglass_p\, \alpha
\end{aligned}$$

$$\begin{matrix}
\mathit{Any} & * & \mathit{Unused} & = & \mathit{Unused} \\
\mathit{Any} & * & _ & = & \mathit{Any}
\end{matrix}$$

Past guards II

Approximation lattice: $\quad \mathit{Unused} < \mathit{Past} < \mathit{NonFuture} < \mathit{Any}$

Status $\hourglass_p\,\varphi$ of predicate symbol p in formula φ:

$$
\begin{aligned}
\hourglass_p\, q(\ldots) &= (\textbf{if } p = q \textbf{ then } \mathit{NonFuture} \textbf{ else } \mathit{Unused}) \\
\hourglass_p(\neg\alpha) &= \hourglass_p\,\alpha \\
\hourglass_p(\alpha \vee \beta) &= (\hourglass_p\,\alpha) \sqcup (\hourglass_p\,\beta) \qquad \hourglass_p(\alpha \wedge \beta) = (\hourglass_p\,\alpha) \sqcup (\hourglass_p\,\beta) \\
\hourglass_p(\bullet_I\,\alpha) &= \mathit{Past} * \hourglass_p\,\alpha \\
\hourglass_p(\bigcirc_I\,\alpha) &= \mathit{Any} * \hourglass_p\,\alpha
\end{aligned}
$$

$\mathit{NonFuture} * x = x$

$$
\hourglass_p(\alpha\ \mathsf{S}_I\ \beta) = (\hourglass_p\,\alpha) \sqcup ((\textbf{if } 0 \in I \textbf{ then } \mathit{NonFuture} \textbf{ else } \mathit{Past}) * \hourglass_p\,\beta)
$$

$$\vdots$$

Past guards II

Approximation lattice: $\mathit{Unused} < \mathit{Past} < \mathit{NonFuture} < \mathit{Any}$

Status $⧖_p\,\varphi$ of predicate symbol p in formula φ:

$$
\begin{aligned}
⧖_p\, q(\ldots) &= (\textbf{if } p = q \textbf{ then } \mathit{NonFuture} \textbf{ else } \mathit{Unused})\\
⧖_p(\neg\alpha) &= ⧖_p\,\alpha\\
⧖_p(\alpha \vee \beta) &= (⧖_p\,\alpha) \sqcup (⧖_p\,\beta) \qquad ⧖_p(\alpha \wedge \beta) = (⧖_p\,\alpha) \sqcup (⧖_p\,\beta)\\
⧖_p(\bullet_I\,\alpha) &= \mathit{Past} * ⧖_p\,\alpha\\
⧖_p(\bigcirc_I\,\alpha) &= \mathit{Any} * ⧖_p\,\alpha\\
⧖_p(\alpha\ \mathsf{S}_I\ \beta) &= (⧖_p\,\alpha) \sqcup ((\textbf{if } 0 \in I \textbf{ then } \mathit{NonFuture} \textbf{ else } \mathit{Past}) * ⧖_p\,\beta)\\
&\vdots
\end{aligned}
$$

In subformulas $\textbf{LetPast}\ p \coloneqq \alpha \textbf{ in } \beta$, predicate p must be past guarded: $⧖_p\,\alpha \leq \mathit{Past}$.

Past guards II

Approximation lattice: $\quad$ *Unused* $<$ *Past* $<$ *NonFuture* $<$ *Any*

Status $⌛_p\,\varphi$ of predicate symbol p in formula φ:

$$
\begin{aligned}
⌛_p\, q(\dots) &= (\textbf{if } p = q \textbf{ then } \mathit{NonFuture} \textbf{ else } \mathit{Unused})\\
⌛_p(\neg\alpha) &= ⌛_p\,\alpha\\
⌛_p(\alpha \vee \beta) &= (⌛_p\,\alpha) \sqcup (⌛_p\,\beta) \qquad ⌛_p(\alpha \wedge \beta) = (⌛_p\,\alpha) \sqcup (⌛_p\,\beta)\\
⌛_p(\bullet_I\,\alpha) &= \mathit{Past} * ⌛_p\,\alpha\\
⌛_p(\bigcirc_I\,\alpha) &= \mathit{Any} * ⌛_p\,\alpha\\
⌛_p(\alpha\ \mathsf{S}_I\ \beta) &= (⌛_p\,\alpha) \sqcup ((\textbf{if } 0 \in I \textbf{ then } \mathit{NonFuture} \textbf{ else } \mathit{Past}) * ⌛_p\,\beta\,)\\
&\vdots
\end{aligned}
$$

In subformulas **LetPast** $p \coloneqq \alpha$ **in** β, predicate p must be past guarded: $⌛_p\,\alpha \leq \mathit{Past}$.

Example: $\quad$ **LetPast** $p \coloneqq (a\ \mathsf{S}_{[1,\infty]}\ p) \vee \Diamond_{[0,3]}\, b$ **in** p

Past guards II

Approximation lattice: $\mathit{Unused} < \mathit{Past} < \mathit{NonFuture} < \mathit{Any}$

Status $⧖_p\,\varphi$ of predicate symbol p in formula φ:

$$
\begin{aligned}
⧖_p\, q(\ldots) &= (\textbf{if } p = q \textbf{ then } \mathit{NonFuture} \textbf{ else } \mathit{Unused})\\
⧖_p(\neg\alpha) &= ⧖_p\,\alpha\\
⧖_p(\alpha \vee \beta) &= (⧖_p\,\alpha) \sqcup (⧖_p\,\beta) \qquad ⧖_p(\alpha \wedge \beta) = (⧖_p\,\alpha) \sqcup (⧖_p\,\beta)\\
⧖_p(\bullet_I\,\alpha) &= \mathit{Past} * ⧖_p\,\alpha\\
⧖_p(\bigcirc_I\,\alpha) &= \mathit{Any} * ⧖_p\,\alpha\\
⧖_p(\alpha\ \mathsf{S}_I\ \beta) &= (⧖_p\,\alpha) \sqcup ((\textbf{if } 0 \in I \textbf{ then } \mathit{NonFuture} \textbf{ else } \boxed{\mathit{Past}) * ⧖_p\,\beta})\\
&\vdots
\end{aligned}
$$

In subformulas $\textbf{LetPast}\ p \coloneqq \alpha \textbf{ in } \beta$, predicate p must be past guarded: $⧖_p\,\alpha \leq \mathit{Past}$.

Example: $\textbf{LetPast}\ p \coloneqq (a\ \boxed{\mathsf{S}_{[1,\infty]}\ p}\ ) \vee \Diamond_{[0,3]}\, b \textbf{ in } p$

Advanced Examples

Tainted hosts from the introduction

$$\begin{aligned}\textbf{LetPast}\ \mathit{taint}(r,h) := {} & \big(\big(\mathit{in}(r,h) \lor \exists h'.\ (\bullet\, \mathit{taint}(r,h')) \land \mathit{com}(h',h)\big) \land \Diamond_{[0,1\mathrm{h}]}\, \mathit{ids}(h)\big) \lor \\ & (\bullet\, \mathit{taint}(r,h)) \\ \textbf{in}\ & \mathit{taint}(r,h) \land \mathit{out}(h,r)\end{aligned}$$

Advanced Examples

Tainted hosts from the introduction

$$\textbf{LetPast}\ \mathit{taint}(r,h) := \Big(\big(\mathit{in}(r,h) \lor \exists h'.\ (\bullet\, \mathit{taint}(r,h')) \land \mathit{com}(h',h)\big) \land \Diamond_{[0,1\mathrm{h}]}\, \mathit{ids}(h)\Big) \lor (\bullet\, \mathit{taint}(r,h))$$
$$\textbf{in}\ \mathit{taint}(r,h) \land \mathit{out}(h,r)$$

Periodic beacon $b(x)$ activations with period $t > 0$ and error $\varepsilon < t$

$$\textbf{LetPast}\ \mathit{periodic}(x) := \mathit{start}(x) \lor \Big(b(x) \land \big((\blacklozenge_{[0,t+\varepsilon]}\, \mathit{start}(x)) \lor (\blacklozenge_{[t-\varepsilon,t+\varepsilon]}\, \mathit{periodic}(x))\big)\Big)$$
$$\textbf{in}\ \mathit{stop}(x) \land \blacklozenge_{[0,t+\varepsilon]}\, \mathit{periodic}(x)$$

Advanced Examples

Tainted hosts from the introduction

$$\textbf{LetPast}\ \mathit{taint}(r,h) := \big(\big(\mathit{in}(r,h) \lor \exists h'.\ (\bullet\,\mathit{taint}(r,h')) \land \mathit{com}(h',h)\big) \land \lozenge_{[0,1\mathrm{h}]}\,\mathit{ids}(h)\big) \lor (\bullet\,\mathit{taint}(r,h))\ \textbf{in}\ \mathit{taint}(r,h) \land \mathit{out}(h,r)$$

Periodic beacon $b(x)$ activations with period $t > 0$ and error $\varepsilon < t$

$$\textbf{LetPast}\ \mathit{periodic}(x) := \mathit{start}(x) \lor \big(b(x) \land \big((\blacklozenge_{[0,t+\varepsilon]}\,\mathit{start}(x)) \lor (\blacklozenge_{[t-\varepsilon,t+\varepsilon]}\,\mathit{periodic}(x))\big)\big)\ \textbf{in}\ \mathit{stop}(x) \land \blacklozenge_{[0,t+\varepsilon]}\,\mathit{periodic}(x)$$

Shortest Paths in a weighted graph (time-point = edge)

Turing Machine simulation (time-point 0 = input, other time-points = steps)

Monitor performance I

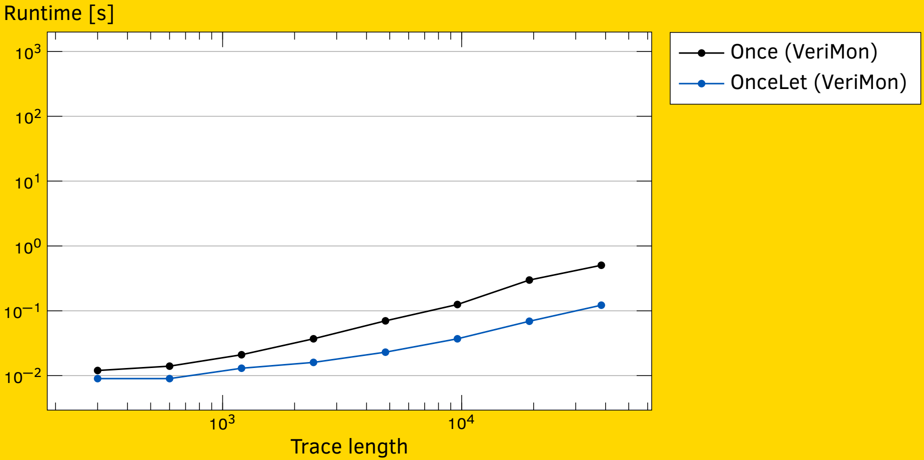
Once: $\mathit{filter}(x,y) \land \neg \blacklozenge s(x,y)$

OnceLet: $\textbf{LetPast}\ o(u,v) \coloneqq s(u,v) \lor \bullet\, o(u,v)\ \textbf{in}\ \mathit{filter}(x,y) \land \neg o(x,y)$

Monitor performance I

Once: $\mathit{filter}(x,y) \wedge \neg \blacklozenge s(x,y)$

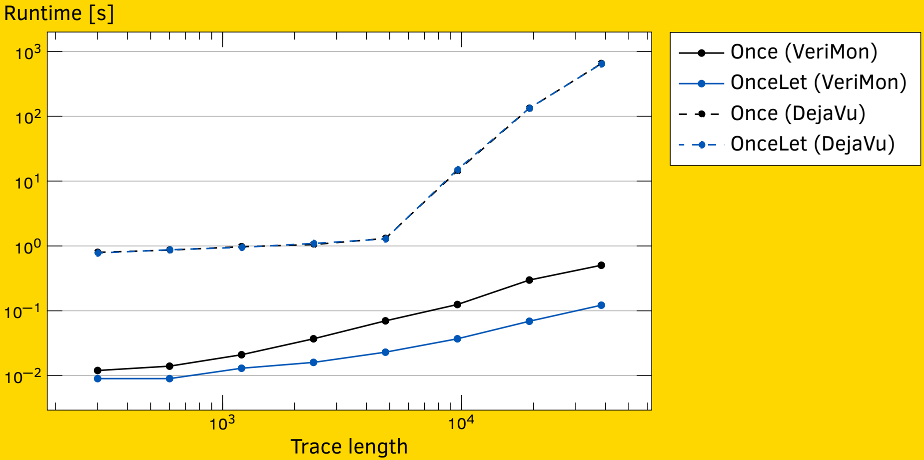
OnceLet: $\textbf{LetPast } o(u,v) \coloneqq s(u,v) \vee \bullet o(u,v) \textbf{ in } \mathit{filter}(x,y) \wedge \neg o(x,y)$



Monitor performance I

Once: $\mathit{filter}(x,y) \land \neg \blacklozenge\, s(x,y)$

OnceLet: $\textbf{LetPast}\ o(u,v) \coloneqq s(u,v) \lor \bullet\, o(u,v)\ \textbf{in}\ \mathit{filter}(x,y) \land \neg o(x,y)$

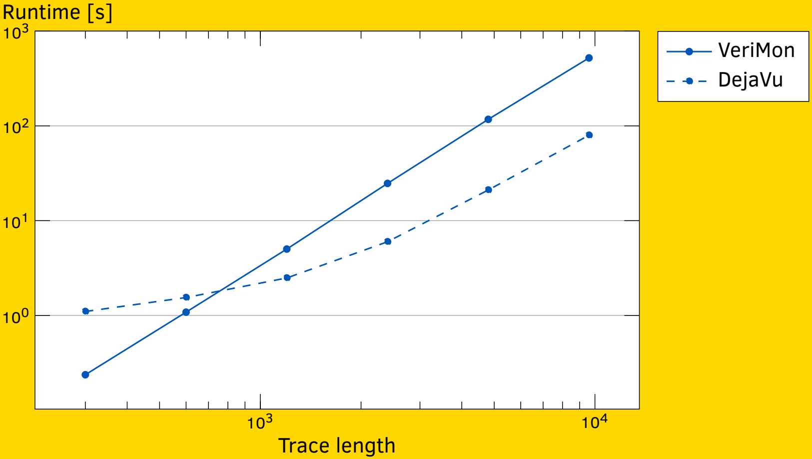


Monitor performance II

Spawn: $\textbf{LetPast}\ p(x,y) \coloneqq s(x,y) \lor (\bullet\, p(x,y)) \lor (\exists t.\ (\bullet\, p(x,t)) \land s(t,y))$
$\textbf{in}\ r(y,x,d) \land \neg p(x,y)$

Monitor performance II

Spawn: **LetPast** $p(x,y) \coloneqq s(x,y) \vee (\bullet\, p(x,y)) \vee (\exists t.\ (\bullet\, p(x,t)) \wedge s(t,y))$
in $r(y,x,d) \wedge \neg p(x,y)$



Future work

1. More precise approximation of past guards
   - For example, p is not considered past guarded in $\blacklozenge_{[10,20]}(\alpha \; \mathsf{U}_{[0,5]} \; p)$
   - Refine *Past* status with minimum time difference

Future work

1. More precise approximation of past guards
   - For example, p is not considered past guarded in $\blacklozenge_{[10,20]}(\alpha\ \mathsf{U}_{[0,5]}\ p)$
   - Refine *Past* status with minimum time difference

2. Non-temporal recursion
   - Needed for ordinary transitive closure
   - Standard restriction: recursion only in positive positions
   - Additional restriction needed to ensure finite results

Future work

1. More precise approximation of past guards
   - For example, p is not considered past guarded in $\blacklozenge_{[10,20]}(\alpha \; \mathsf{U}_{[0,5]} \; p)$
   - Refine *Past* status with minimum time difference

2. Non-temporal recursion
   - Needed for ordinary transitive closure
   - Standard restriction: recursion only in positive positions
   - Additional restriction needed to ensure finite results
     E.g., **LetRec** $N(x) \coloneqq x = 0 \lor N(x - 1)$ **in** $N(x)$ is the set of natural numbers!

Future work

1. More precise approximation of past guards
   - For example, p is not considered past guarded in $\blacklozenge_{[10,20]}(\alpha \, \mathsf{U}_{[0,5]} \, p)$
   - Refine *Past* status with minimum time difference
2. Non-temporal recursion
   - Needed for ordinary transitive closure
   - Standard restriction: recursion only in positive positions
   - Additional restriction needed to ensure finite results
     E.g., $\mathbf{LetRec}\ N(x) \coloneqq x = 0 \vee N(x - 1)\ \mathbf{in}\ N(x)$ is the set of natural numbers!
3. Recursion only through future operators

Future work

1. More precise approximation of past guards
   - For example, p is not considered past guarded in $\blacklozenge_{[10,20]}(\alpha \; \mathsf{U}_{[0,5]} \; p)$
   - Refine *Past* status with minimum time difference

2. Non-temporal recursion
   - Needed for ordinary transitive closure
   - Standard restriction: recursion only in positive positions
   - Additional restriction needed to ensure finite results
     E.g., **LetRec** $N(x) \coloneqq x = 0 \vee N(x - 1)$ **in** $N(x)$ is the set of natural numbers!

3. Recursion only through future operators

4. Optimizing the evaluation algorithm
   - E.g., index datastructures to speed up relational joins

Our paper: MFOTL meets recursive rules

Recursive let operator in MFOTL

- Semantics only considers recursive occurrences strictly in the past
- Allows efficient evaluation (no fixpoint iteration)
- Syntactic fragment: every recursive occurrence has a past guard

Implementation and Evaluation

- Part of the VeriMon monitor
- Formally verified in Isabelle/HOL
- Isabelle (40 kLOC) ⟶ OCaml (11 kLOC)
- Compared with DejaVu on PFLTL

Our paper: MFOTL meets recursive rules

Recursive let operator in MFOTL

- Semantics only considers recursive occurrences strictly in the past
- Allows efficient evaluation (no fixpoint iteration)
- Syntactic fragment: every rec… occurrence has a past gua…

Implementation and Evaluation

- Part of the VeriMon monitor
- Formally verified in Isabelle/HOL
- Isabelle (40 kLOC) ⟶ OCaml (11 kLOC)
- …ared with DejaVu on PFLTL

Danke!
Fragen?