Gray-box Monitoring of Hyperproperties

Sandro Stucki[1] César Sánchez[2]
Gerardo Schneider[1] Borzoo Bonakdarpour[3]

[1]GU | Chalmers, Sweden [2]IMDEA SW, Spain [2]ISU, USA

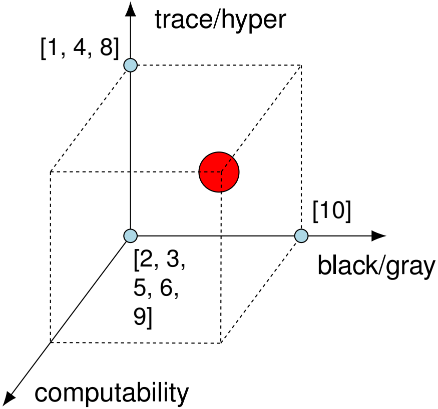
FM '19, Porto, Portugal, 11 October 2019

sandro.stucki@gu.se @stuckintheory

UNIVERSITY OF GOTHENBURG

CHALMERS UNIVERSITY OF TECHNOLOGY

The monitorability cube



Motivation: distributed data minimality

- Distributed data minimality (DDM)
  - privacy property (GDPR)
  - generalization of data minimality to a multi-input setting

Motivation: distributed data minimality

- Distributed data minimality (DDM)
  - privacy property (GDPR)
  - generalization of data minimality to a multi-input setting
  - $\forall\forall\exists\exists$-hyperproperty

$$\varphi_i \;=\; \forall\pi.\forall\pi'.\exists\tau.\exists\tau'.\,\neg\,\mathrm{same}_i(\pi,\pi') \rightarrow \begin{pmatrix} \mathrm{same}_i(\pi,\tau) \land \mathrm{same}_i(\pi',\tau') \land \\ \mathrm{almost}_i(\tau,\tau') \land \neg\,\mathrm{output}(\tau,\tau') \end{pmatrix}$$

Motivation: distributed data minimality

- Distributed data minimality (DDM)
  - privacy property (GDPR)
  - generalization of data minimality to a multi-input setting
  - $\forall\forall\exists\exists$-hyperproperty

$$\varphi_i \;=\; \forall\pi.\forall\pi'.\exists\tau.\exists\tau'.\ \neg\,\mathrm{same}_i(\pi,\pi') \rightarrow \begin{pmatrix} \mathrm{same}_i(\pi,\tau) \wedge \mathrm{same}_i(\pi',\tau') \wedge \\ \mathrm{almost}_i(\tau,\tau') \wedge \neg\,\mathrm{output}(\tau,\tau') \end{pmatrix}$$

- Challenges:
  - Not black-box monitorable.
  - Undecidable.
  - Defined over arbitrary domains/datatypes.

Motivation: distributed data minimality

- Distributed data minimality (DDM)
  - privacy property (GDPR)
  - generalization of data minimality to a multi-input setting
  - $\forall\forall\exists\exists$-hyperproperty

$$\varphi_i = \forall\pi.\forall\pi'.\exists\tau.\exists\tau'.\ \neg\,\mathrm{same}_i(\pi,\pi') \rightarrow \begin{pmatrix} \mathrm{same}_i(\pi,\tau) \wedge \mathrm{same}_i(\pi',\tau') \wedge \\ \mathrm{almost}_i(\tau,\tau') \wedge \neg\,\mathrm{output}(\tau,\tau') \end{pmatrix}$$

- Challenges:
  - Not black-box monitorable.
  - Undecidable.
  - Defined over arbitrary domains/datatypes.

Yet, we have a monitor…

Motivation: distributed data minimality

- Distributed data minimality (DDM)
  - privacy property (GDPR)
  - generalization of data minimality to a multi-input setting
  - $\forall\forall\exists\exists$-hyperproperty

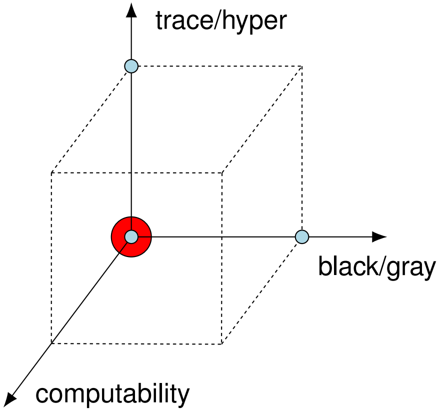
$$\varphi_i = \forall\pi.\forall\pi'.\exists\tau.\exists\tau'.\ \neg\,\mathrm{same}_i(\pi,\pi') \rightarrow \begin{pmatrix} \mathrm{same}_i(\pi,\tau) \wedge \mathrm{same}_i(\pi',\tau') \wedge \\ \mathrm{almost}_i(\tau,\tau') \wedge \neg\,\mathrm{output}(\tau,\tau') \end{pmatrix}$$

- Challenges:
  - Not black-box monitorable.
  - Undecidable.
  - Defined over arbitrary domains/datatypes.

Yet, we have a monitor...

what's going on here?

Trace properties – LTL

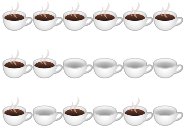


Trace properties – LTL

$\varphi_s = \square$☕ $\qquad \varphi_l = \Diamond$☕ $\qquad \varphi_r = \square\Diamond$☕

Trace properties – LTL

$\varphi_s = \Box$☕ $\qquad \varphi_l = \Diamond$☕ $\qquad \varphi_r = \Box\Diamond$☕



$t_1 = \cdots$	$t_1 \models \varphi_s$	$t_1 \models \varphi_l$	$t_1 \models \varphi_r$
$t_2 = \cdots$	$t_2 \not\models \varphi_s$	$t_2 \models \varphi_l$	$t_2 \not\models \varphi_r$
$t_3 = \cdots$	$t_3 \not\models \varphi_s$	$t_3 \models \varphi_l$	$t_3 \models \varphi_r$

Trace properties – LTL

$\varphi_s = \Box$☕ $\qquad \varphi_l = \Diamond$☕ $\qquad \varphi_r = \Box\Diamond$☕

$t_1 \models \varphi_s$	$t_1 \models \varphi_l$	$t_1 \models \varphi_r$
$t_2 \not\models \varphi_s$	$t_2 \models \varphi_l$	$t_2 \not\models \varphi_r$
$t_3 \not\models \varphi_s$	$t_3 \models \varphi_l$	$t_3 \models \varphi_r$

$\varphi ::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid \bigcirc\varphi \mid \varphi\,\mathcal{U}\,\varphi \qquad \Diamond\varphi \equiv \text{true}\ \mathcal{U}\ \varphi \qquad \Box\varphi \equiv \neg\Diamond\neg\varphi$

$t \models p$	iff	$p \in t[0]$
$t \models \neg\varphi$	iff	$t \not\models \varphi$
$t \models \varphi_1 \vee \varphi_2$	iff	$t \models \varphi_1$ or $t \models \varphi_2$
$t \models \bigcirc\varphi$	iff	$t[1,..] \models \varphi$
$t \models \varphi_1\,\mathcal{U}\,\varphi_2$	iff	for some i, $t[i,..] \models \varphi_2$ and for all $j < i$, $t[j,..] \models \varphi_1$

Monitoring LTL

$\varphi_s = \square$☕ $\qquad$ $\varphi_l = \lozenge$☕ $\qquad$ $\varphi_r = \square\lozenge$☕

Monitoring LTL

$\varphi_s = \Box$☕ $\qquad$ $\varphi_l = \Diamond$☕ $\qquad$ $\varphi_r = \Box\Diamond$☕

- Observation: the world today at 10am

$u_{10} =$ ☕☕☕☕

Monitoring LTL

$\varphi_s = \Box$☕ $\quad \varphi_l = \Diamond$☕ $\quad \varphi_r = \Box\Diamond$☕

- **Observation:** the world today at 10am

$u_{10} =$ ☕☕☕☕

- **Update:** the world at 11am

$u_{11} =$ ☕☕☕☕☕☕☕☕

Monitoring LTL

$\varphi_s = \Box$☕ $\varphi_l = \Diamond$☕ $\varphi_r = \Box\Diamond$☕

- Observation: the world today at 10am

  $u_{10} =$ ☕☕☕☕

- Update: the world at 11am

  $u_{11} =$ ☕☕☕☕☕☕☕☕

φ_s Is there always coffee?

Monitoring LTL

$\varphi_s = \Box$☕ $\qquad \varphi_l = \Diamond$☕ $\qquad \varphi_r = \Box\Diamond$☕

- Observation: the world today at 10am

$u_{10} =$ ☕☕☕☕

- Update: the world at 11am

$u_{11} =$ ☕☕☕☕☕☕☕☕

φ_s Is there always coffee? $\qquad u_{10} \rightarrow$ **?**

Monitoring LTL

$\varphi_s = \Box$☕ $\qquad \varphi_l = \Diamond$☕ $\qquad \varphi_r = \Box\Diamond$☕

- Observation: the world today at 10am

  $u_{10} =$ ☕☕☕☕

- Update: the world at 11am

  $u_{11} =$ ☕☕☕☕☕☕☕☕

φ_s Is there always coffee? $\qquad u_{10} \rightarrow$ **?**, $u_{11} \rightarrow$ ✗

Monitoring LTL

$\varphi_s = \Box$☕ $\qquad \varphi_l = \Diamond$☕ $\qquad \varphi_r = \Box\Diamond$☕

- Observation: the world today at 10am

  $u_{10} =$ ☕☕☕☕

- Update: the world at 11am

  $u_{11} =$ 

φ_s	Is there always coffee?	$u_{10} \rightarrow$ **?**, $u_{11} \rightarrow$ ✗
φ_l	Is there eventually coffee?	$u_{10} \rightarrow$ ✓, $u_{11} \rightarrow$ ✓

Monitoring LTL

$\varphi_s = \Box$☕ $\qquad \varphi_l = \Diamond$☕ $\qquad \varphi_r = \Box\Diamond$☕

- Observation: the world today at 10am

$u_{10} =$ 

- Update: the world at 11am

$u_{11} =$ 

φ_s	Is there always coffee?	$u_{10} \rightarrow$ **?**, $u_{11} \rightarrow$ ✗
φ_l	Is there eventually coffee?	$u_{10} \rightarrow$ ✓, $u_{11} \rightarrow$ ✓
φ_r	Is there always eventually coffee?	$u_{10} \rightarrow$ **?**, $u_{11} \rightarrow$ **?**

Monitoring LTL

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), at runtime.

Monitoring LTL

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation u.

Monitoring LTL

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation u.

Definition

A finite observation u permanently satisfies (resp. violates) φ, if every infinite extension of u satisfies (resp. violates) φ:

$$u \models^s \varphi \quad \text{iff} \quad \text{for all } t \in \Sigma^\omega \text{ such that } u \preceq t,\ t \models \varphi$$

$$u \models^v \varphi \quad \text{iff} \quad \text{for all } t \in \Sigma^\omega \text{ such that } u \preceq t,\ t \not\models \varphi$$

Monitoring LTL

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation u.

Definition

A finite observation u permanently satisfies (resp. violates) φ, if every infinite extension of u satisfies (resp. violates) φ:

$$u \models^s \varphi \quad \text{iff} \quad \text{for all } t \in \Sigma^\omega \text{ such that } u \preceq t,\ t \models \varphi$$

$$u \models^v \varphi \quad \text{iff} \quad \text{for all } t \in \Sigma^\omega \text{ such that } u \preceq t,\ t \not\models \varphi$$

$$u_{11} = \text{☕☕☕☕☕☕☕☕}$$

$u_{11} \not\models^s \Box$☕ $\qquad u_{11} \models^s \Diamond$☕ $\qquad u_{11} \not\models^s \Box\Diamond$☕

$u_{11} \models^v \Box$☕ $\qquad u_{11} \not\models^v \Diamond$☕ $\qquad u_{11} \not\models^v \Box\Diamond$☕

Monitors for LTL

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation u.

Monitors for LTL

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation u.

A monitor for a property φ is a computable function $M_\varphi \colon \Sigma^* \to \{✓, ✗, \mathbf{?}\}$ that decides a verdict for φ given a finite u.

Monitors for LTL

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation u.

A monitor for a property φ is a computable function $M_\varphi \colon \Sigma^* \to \{✓, ✗, \mathbf{?}\}$ that decides a verdict for φ given a finite u.

The monitor M_φ is sound if

$$u \models^s \varphi \quad \text{if} \quad M_\varphi(u) = ✓, \qquad u \models^v \varphi \quad \text{if} \quad M_\varphi(u) = ✗$$

Monitors for LTL

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation u.

A monitor for a property φ is a computable function $M_\varphi \colon \Sigma^* \to \{✓, ✗, \textbf{?}\}$ that decides a verdict for φ given a finite u.

The monitor M_φ is sound if

$$u \models^s \varphi \quad \text{if} \quad M_\varphi(u) = ✓, \qquad u \models^v \varphi \quad \text{if} \quad M_\varphi(u) = ✗$$

The monitor M_φ is complete if

$$M_\varphi(u) = ✓ \text{ if } u \models^s \varphi, \quad M_\varphi(u) = ✗ \text{ if } u \models^v \varphi, \quad M_\varphi(u) = \textbf{?} \text{ o/w.}$$

Monitors for LTL

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation u.

A monitor for a property φ is a computable function $M_\varphi \colon \Sigma^* \to \{✓, ✗, \mathbf{?}\}$ that decides a verdict for φ given a finite u.

The monitor M_φ is sound if

$$u \models^s \varphi \quad \text{if} \quad M_\varphi(u) = ✓, \qquad u \models^v \varphi \quad \text{if} \quad M_\varphi(u) = ✗$$

The monitor M_φ is complete if

$$M_\varphi(u) = ✓ \text{ if } u \models^s \varphi, \quad M_\varphi(u) = ✗ \text{ if } u \models^v \varphi, \quad M_\varphi(u) = \mathbf{?} \text{ o/w.}$$

Fact: every LTL formula has a sound and complete monitor.

Monitorability of LTL formulas

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation u.

$\varphi_r = \Box\Diamond$ 

$u_{11} =$ 

$u_{11} \not\models^{s} \varphi_r$ $\qquad$ $u_{11} \not\models^{v} \varphi_r$

Monitorability of LTL formulas

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation u.

$$\varphi_r = \Box\Diamond ☕ \qquad u_{11} =$$ 

$$u_{11} \not\models^s \varphi_r \qquad u_{11} \not\models^v \varphi_r$$

Observation: $u \not\models^s \varphi_r$ and $u \not\models^v \varphi_r$ for any u.

Monitorability of LTL formulas

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation u.

$\varphi_r = \Box\Diamond$☕ $u_{11} =$ 

$u_{11} \not\models^{s} \varphi_r$ $u_{11} \not\models^{v} \varphi_r$

Observation: $u \not\models^{s} \varphi_r$ and $u \not\models^{v} \varphi_r$ for any u.

There's no point in monitoring φ_r!

Monitorability of LTL formulas

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation u.

$$\varphi_r = \Box\Diamond\text{☕} \qquad\qquad u_{11} = \text{☕☕☕☕☕☕☕☕}$$

$$u_{11} \not\models^s \varphi_r \qquad\qquad u_{11} \not\models^v \varphi_r$$

Observation: $u \not\models^s \varphi_r$ and $u \not\models^v \varphi_r$ for any u.

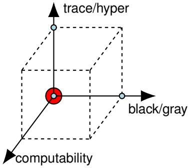
There's no point in monitoring φ_r!

Definition (Pnueli & Zaks 2006)

A formula φ is *(semantically) monitorable* if every observation u has an extension $v \succeq u$, such that either $v \models^s \varphi$ or $v \models^v \varphi$.

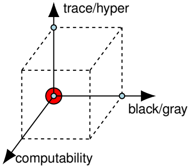
LTL – Summary

- Properties defined over individual traces.
  $\Rightarrow$ Properties describe sets of traces.
- Sound and complete monitors can be constructed for any formula.
- Not every formula is monitorable. For example,
  - safety and liveness properties are monitorable,
  - recurrence properties ($\square\lozenge$) are not.



LTL – Summary

- Properties defined over individual traces.
  $\Rightarrow$ Properties describe sets of traces.
- Sound and complete monitors can be constructed for any formula.
- Not every formula is monitorable. For example,
  - safety and liveness properties are monitorable,
  - recurrence properties ($\Box\Diamond$) are not.



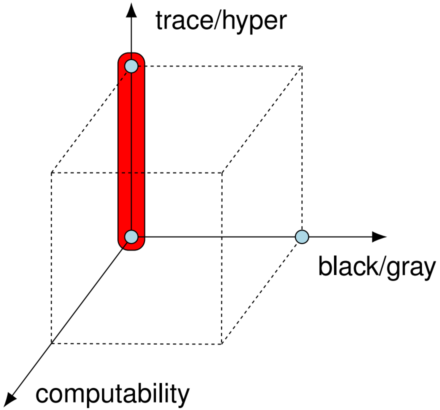
[9] A. Pnueli and A. Zaks. *PSL Model Checking and Run-time Verification via Testers.*, FM'06, Springer, 2006.

[5] Y. Falcone, J-C. Fernandez, and L. Mounier. *What can you verify and enforce at runtime?*, STTT 14(3), 2012.

[8] K. Havelund and D. Peled. *Runtime Verification: From Propositional to First-Order Temporal Logic.* RV'18, Springer, 2018.

. . . and many more!

Hyperproperties – HyperLTL



Hyperproperties – HyperLTL

$\varphi_u = \forall\pi.\forall\tau.\Box(\text{☕}_\pi \rightarrow \text{☕}_\tau)$

$\varphi_a = \forall\pi.\exists\tau.\Box($$_\pi \rightarrow \text{☕}_\tau)$

Hyperproperties – HyperLTL

$\varphi_u = \forall\pi.\forall\tau.\square(☕_\pi \rightarrow ☕_\tau)$ $\qquad$ $\varphi_a = \forall\pi.\exists\tau.\square(○_\pi \rightarrow ☕_\tau)$

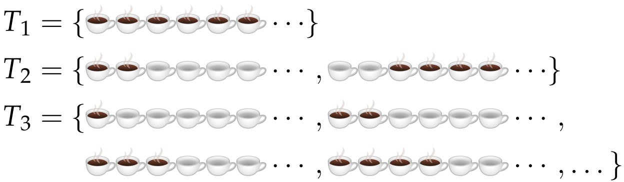
$T_1 = \{☕☕☕☕☕☕\cdots\}$ $\qquad$ $T_1 \models \varphi_u$ $\qquad$ $T_1 \models \varphi_a$

$T_2 = \{☕☕○○○○\cdots, ○○☕☕☕☕\cdots\}$ $\qquad$ $T_2 \not\models \varphi_u$ $\qquad$ $T_2 \models \varphi_a$

$T_3 = \{☕○○○○○\cdots, ☕☕○○○○\cdots,$ $\qquad$ $T_3 \not\models \varphi_u$ $\qquad$ $T_3 \not\models \varphi_a$

$☕☕☕○○○\cdots, ☕☕☕☕○○\cdots, \ldots\}$

Hyperproperties – HyperLTL



$T_1 \models \varphi_u$	$T_1 \models \varphi_a$
$T_2 \not\models \varphi_u$	$T_2 \models \varphi_a$
$T_3 \not\models \varphi_u$	$T_3 \not\models \varphi_a$

$$\varphi ::= \forall \pi . \varphi \mid \exists \pi . \varphi \mid \psi \qquad \psi ::= a_\pi \mid \neg \psi \mid \psi \vee \psi \mid \bigcirc \psi \mid \psi \,\mathcal{U}\, \psi$$

$\Pi \models a_\pi$	iff	$a \in \Pi(\pi)[0]$
$\Pi \models \psi_1 \vee \psi_2$	iff	$\Pi \models \psi_1$ or $\Pi \models \psi_2$
$\Pi \models \neg \psi$	iff	$\Pi \not\models \psi$
$\Pi \models \bigcirc \psi$	iff	$\Pi[1..] \models \psi$
$\Pi \models \psi_1 \,\mathcal{U}\, \psi_2$	iff	for some i, $\Pi[i,..] \models \psi_2$, and for all $j < i$ $T, \Pi[j,..] \models \psi_1$

$T, \Pi \models \forall \pi . \varphi$	iff	$T, \Pi[\pi \to t] \models \varphi$ for all $t \in T$
$T, \Pi \models \exists \pi . \varphi$	iff	$T, \Pi[\pi \to t] \models \varphi$ for some $t \in T$
$T, \Pi \models \psi$	iff	$\Pi \models \psi$

Monitoring HyperLTL

$$\varphi_u = \forall \pi . \forall \tau . \Box (\text{☕}_\pi \rightarrow \text{☕}_\tau) \qquad \varphi_a = \forall \pi . \exists \tau . \Box (\text{☕}_\pi \rightarrow \text{☕}_\tau)$$

Monitoring HyperLTL

$$\varphi_u = \forall\pi.\forall\tau.\Box(\text{☕}_\pi \rightarrow \text{☕}_\tau) \qquad \varphi_a = \forall\pi.\exists\tau.\Box(\text{☕}_\pi \rightarrow \text{☕}_\tau)$$

- **Observation:** the world today at 10am

  $U_{10} = \{\text{☕☕☕☕}\}$

Monitoring HyperLTL

$$\varphi_u = \forall\pi.\forall\tau.\Box(\text{☕}_\pi \rightarrow \text{☕}_\tau) \qquad \varphi_a = \forall\pi.\exists\tau.\Box(\text{○}_\pi \rightarrow \text{☕}_\tau)$$

- Observation: the world today at 10am

  $U_{10} = \{\text{☕☕☕☕}\}$

- Update: the world at 11am

  $U_{11} = \{\text{☕☕☕☕☕○○○},\ \text{○○○○☕☕☕☕}\}$

Monitoring HyperLTL

$$\varphi_u = \forall \pi. \forall \tau. \Box (☕_\pi \rightarrow ☕_\tau) \qquad \varphi_a = \forall \pi. \exists \tau. \Box (◌_\pi \rightarrow ☕_\tau)$$

- Observation: the world today at 10am

  $U_{10} = \{☕☕☕☕\}$

- Update: the world at 11am

  $U_{11} = \{☕☕☕☕☕◌◌◌,\ ◌◌◌◌☕☕☕☕,\ ☕☕◌\}$

Monitoring HyperLTL

$$\varphi_u = \forall \pi . \forall \tau . \Box (\text{☕}_\pi \rightarrow \text{☕}_\tau) \qquad \varphi_a = \forall \pi . \exists \tau . \Box (\text{○}_\pi \rightarrow \text{☕}_\tau)$$

- Observation: the world today at 10am

  $U_{10} = \{\text{☕☕☕☕}\}$

- Update: the world at 11am

  $U_{11} = \{\text{☕☕☕☕☕○○○},\ \text{○○○○☕☕☕☕},\ \text{☕☕○}\}$

φ_u Is there always coffee everywhere at the same time?

Monitoring HyperLTL

$$\varphi_u = \forall\pi.\forall\tau.\square(\text{☕}_\pi \rightarrow \text{☕}_\tau) \qquad \varphi_a = \forall\pi.\exists\tau.\square(\text{◯}_\pi \rightarrow \text{☕}_\tau)$$

- Observation: the world today at 10am

  $U_{10} = \{\text{☕☕☕☕}\}$

- Update: the world at 11am

  $U_{11} = \{\text{☕☕☕☕☕◯◯◯},\ \text{◯◯◯◯☕☕☕☕},\ \text{☕☕◯}\}$

φ_u Is there always coffee everywhere at the same time? $U_{10} \rightarrow$ **?**,

Monitoring HyperLTL

$$\varphi_u = \forall\pi.\forall\tau.\Box(☕_\pi \rightarrow ☕_\tau) \qquad \varphi_a = \forall\pi.\exists\tau.\Box(☕_\pi \rightarrow ☕_\tau)$$

- Observation: the world today at 10am

  $U_{10} = \{☕☕☕☕\}$

- Update: the world at 11am

  $U_{11} = \{☕☕☕☕☕☕☕☕,\ ☕☕☕☕☕☕☕☕,\ ☕☕☕\}$

φ_u Is there always coffee everywhere at the same time? $U_{10} \rightarrow$ **?**, $U_{11} \rightarrow$ ✗

Monitoring HyperLTL

$$\varphi_u = \forall\pi.\forall\tau.\Box(☕_\pi \rightarrow ☕_\tau) \qquad \varphi_a = \forall\pi.\exists\tau.\Box(◡_\pi \rightarrow ☕_\tau)$$

- **Observation:** the world today at 10am

  $U_{10} = \{☕☕☕☕\}$

- **Update:** the world at 11am

  $U_{11} = \{☕☕☕☕☕◡◡◡,\ ◡◡◡◡☕☕☕☕,\ ☕☕◡\}$

φ_u Is there always coffee everywhere at the same time? $U_{10} \rightarrow$ **?**, $U_{11} \rightarrow$ ✗

φ_a Is there always coffee somewhere? $U_{10} \rightarrow$ **?**, $U_{11} \rightarrow$ **?**

Monitoring HyperLTL

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation U.

Monitoring HyperLTL

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation U.

Definition

A finite observation $U \in \mathcal{P}_{\mathit{fin}}(\Sigma^*)$ permanently satisfies (resp. violates) φ, if every infinite extension of U satisfies (resp. violates) φ:

$$U \models^s \varphi \quad \text{iff} \quad \text{for all } T \in \mathcal{P}(\Sigma^\omega) \text{ such that } U \preceq T,\ T \models \varphi$$
$$U \models^v \varphi \quad \text{iff} \quad \text{for all } T \in \mathcal{P}(\Sigma^\omega) \text{ such that } U \preceq T,\ T \not\models \varphi$$

Monitoring HyperLTL

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation U.

Definition

A finite observation $U \in \mathcal{P}_{fin}(\Sigma^*)$ permanently satisfies (resp. violates) φ, if every infinite extension of U satisfies (resp. violates) φ:

$$U \models^s \varphi \quad \text{iff} \quad \text{for all } T \in \mathcal{P}(\Sigma^\omega) \text{ such that } U \preceq T,\ T \models \varphi$$

$$U \models^v \varphi \quad \text{iff} \quad \text{for all } T \in \mathcal{P}(\Sigma^\omega) \text{ such that } U \preceq T,\ T \not\models \varphi$$

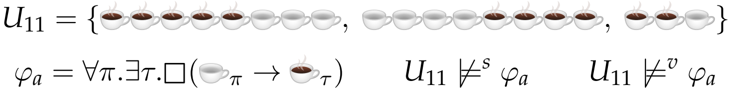
$$U_{11} = \{☕☕☕☕☕○○○,\ ○○○○☕☕☕☕,\ ☕☕○\}$$

$$U_{11} \not\models^s \forall\pi.\forall\tau.\Box(☕_\pi \rightarrow ☕_\tau) \qquad U_{11} \not\models^s \forall\pi.\exists\tau.\Box(○_\pi \rightarrow ☕_\tau)$$

$$U_{11} \models^v \forall\pi.\forall\tau.\Box(☕_\pi \rightarrow ☕_\tau) \qquad U_{11} \not\models^v \forall\pi.\exists\tau.\Box(○_\pi \rightarrow ☕_\tau)$$

Monitorability of HyperLTL formulas

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation U.



$U_{11} \not\models^{s} \varphi_a$ $\qquad$ $U_{11} \not\models^{v} \varphi_a$

Monitorability of HyperLTL formulas

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation U.

$$U_{11} = \{☕☕☕☕☕○○○,\ ○○○○☕☕☕☕,\ ☕☕○\}$$

$$\varphi_a = \forall\pi.\exists\tau.\Box(○_\pi \rightarrow ☕_\tau) \qquad U_{11} \not\models^s \varphi_a \qquad U_{11} \not\models^v \varphi_a$$

Observation: $U \not\models^s \varphi_a$ and $U \not\models^v \varphi_a$ for any U.

Monitorability of HyperLTL formulas

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation U.

$$U_{11} = \{☕☕☕☕☕🥛🥛🥛,\ 🥛🥛🥛🥛☕☕☕☕,\ ☕☕🥛\}$$

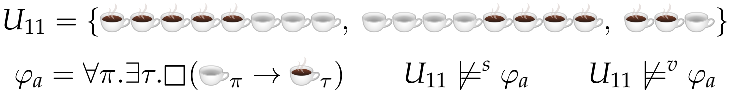
$$\varphi_a = \forall\pi.\exists\tau.\Box(🥛_\pi \rightarrow ☕_\tau) \qquad U_{11} \not\models^s \varphi_a \qquad U_{11} \not\models^v \varphi_a$$

Observation: $U \not\models^s \varphi_a$ and $U \not\models^v \varphi_a$ for any U.

There's no point in monitoring φ_a!

Monitorability of HyperLTL formulas

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation U.



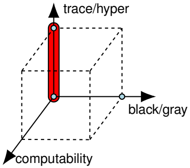
Observation: $U \not\models^s \varphi_a$ and $U \not\models^v \varphi_a$ for any U.

There's no point in monitoring φ_a!

Definition (Agrawal & Bonakdarpour 2016)

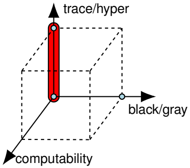
A formula φ is *(semantically) monitorable* if every observation U has an extension $V \succeq U$, such that $V \models^s \varphi$ or $V \models^v \varphi$.

HyperLTL – Summary



- Properties defined over sets of traces.
  $\Rightarrow$ Properties describe sets of sets of traces.
- Sound and complete monitors can be constructed for some formulas.
  - For example, for formulas without quantifier alternations.
  - But what about formulas with alternations?
- Most formulas are not monitorable.
  - For example, $\forall^+\exists^+$-properties are not!

HyperLTL – Summary



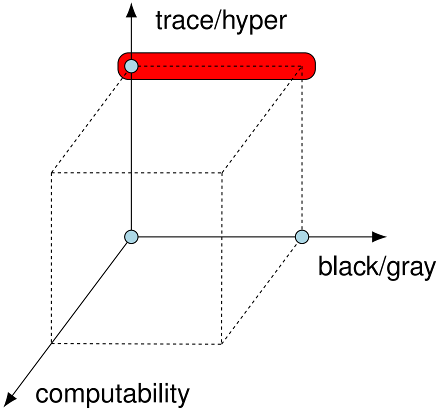
- Properties defined over sets of traces.
  $\Rightarrow$ Properties describe sets of sets of traces.
- Sound and complete monitors can be constructed for some formulas.
  - For example, for formulas without quantifier alternations.
  - But what about formulas with alternations?
- Most formulas are not monitorable.
  - For example, $\forall^+\exists^+$-properties are not!

[1] S. Agrawal and B. Bonakdarpour. *Runtime Verification of k-Safety Hyperproperties in HyperLTL.* CSF'16, IEEE CS Press, 2016.

[8] K. Havelund and D. Peled. *Runtime Verification: From Propositional to First-Order Temporal Logic.* RV'18, Springer, 2018.

[7] C. Hahn. *Algorithms for Monitoring Hyperproperties.* RV'19, Springer, 2019.

Gray-box monitoring (of hyperproperties)



Why is φ_a not monitorable?

Theorem

Let $\varphi_a = \forall\pi.\exists\tau.\square($$_\pi \to$ $_\tau)$*. Then* $U \not\models^s \varphi_a$ *and* $U \not\models^v \varphi_a$ *for all* $U \in \mathcal{P}_{\mathit{fin}}(\Sigma^*)$*.*

Why is φ_a not monitorable?

Theorem

Let $\varphi_a = \forall\pi.\exists\tau.\Box(☕_\pi \to ☕_\tau)$. *Then* $U \not\models^s \varphi_a$ *and* $U \not\models^v \varphi_a$ *for all* $U \in \mathcal{P}_{\mathit{fin}}(\Sigma^*)$.

Proof.

$U \not\models^v \varphi_a$ $\;U \preceq \Sigma^\omega$, and $\Sigma^\omega \models \varphi_a$ because ☕☕☕$\cdots \in \Sigma^\omega$;

Why is φ_a not monitorable?

Theorem

Let $\varphi_a = \forall\pi.\exists\tau.\Box(☕_\pi \rightarrow ☕_\tau)$. Then $U \not\models^s \varphi_a$ and $U \not\models^v \varphi_a$ for all $U \in \mathcal{P}_{\mathit{fin}}(\Sigma^*)$.

Proof.

$U \not\models^v \varphi_a$ $U \preceq \Sigma^\omega$, and $\Sigma^\omega \models \varphi_a$ because ☕☕☕ $\cdots \in \Sigma^\omega$;

$U \not\models^s \varphi_a$ define T as $T = \{t \in \Sigma^\omega \mid w = u$☕☕☕$\cdots$ for $u \in U\}$;
then $U \preceq T$ and $T \not\models \varphi_a$. $\Box$

Why is φ_a not monitorable?

Theorem

Let $\varphi_a = \forall\pi.\exists\tau.\square(\text{○}_\pi \rightarrow \text{☕}_\tau)$. Then $U \not\models^s \varphi_a$ and $U \not\models^v \varphi_a$ for all $U \in \mathcal{P}_{fin}(\Sigma^*)$.

Proof.

$U \not\models^v \varphi_a$ $U \preceq \Sigma^\omega$, and $\Sigma^\omega \models \varphi_a$ because ☕☕☕ $\cdots \in \Sigma^\omega$;

$U \not\models^s \varphi_a$ define T as $T = \{t \in \Sigma^\omega \mid w = u\text{○○○}\cdots$ for $u \in U\}$;
then $U \preceq T$ and $T \not\models \varphi_a$. $\square$

This theorem can be generalized to all formulas $\varphi = \forall\pi.\exists\tau.\square P(\pi, \tau)$ where P is

- a binary (non-temporal) predicate,
- serial,
- non-reflexive.

Why is φ_a not monitorable?

Theorem

Let $\varphi_a = \forall\pi.\exists\tau.\Box(☕_\pi \rightarrow ☕_\tau)$. Then $U \not\models^s \varphi_a$ and $U \not\models^v \varphi_a$ for all $U \in \mathcal{P}_{fin}(\Sigma^*)$.

Proof.

$U \not\models^v \varphi_a$ $U \preceq \Sigma^\omega$, and $\Sigma^\omega \models \varphi_a$ because ☕☕☕$\cdots \in \Sigma^\omega$;

$U \not\models^s \varphi_a$ define T as $T = \{t \in \Sigma^\omega \mid w = u$☕☕☕$\cdots$ for $u \in U\}$;
then $U \preceq T$ and $T \not\models \varphi_a$. $\square$

This theorem can be generalized to all formulas $\varphi = \forall\pi.\exists\tau.\Box P(\pi,\tau)$ where P is

- a binary (non-temporal) predicate,
- serial,
- non-reflexive.

OK, but let's have a closer look at this proof...

Why is φ_a not monitorable?

Theorem

Let $\varphi_a = \forall\pi.\exists\tau.\Box(☕_\pi \to ☕_\tau)$. *Then* $U \not\models^s \varphi_a$ *and* $U \not\models^v \varphi_a$ *for all* $U \in \mathcal{P}_{fin}(\Sigma^*)$.

Proof.

$U \not\models^v \varphi_a$ $U \preceq \Sigma^\omega$, and $\Sigma^\omega \models \varphi_a$ because ☕☕☕$\cdots \in \Sigma^\omega$;

...

This step is somewhat dubious.

Why is φ_a not monitorable?

Theorem

Let $\varphi_a = \forall\pi.\exists\tau.\Box(☕_\pi \rightarrow ☕_\tau)$. *Then* $U \not\models^s \varphi_a$ *and* $U \not\models^v \varphi_a$ *for all* $U \in \mathcal{P}_{fin}(\Sigma^*)$.

Proof.

$U \not\models^v \varphi_a$ $U \preceq \Sigma^\omega$, and $\Sigma^\omega \models \varphi_a$ because ☕☕☕$\cdots \in \Sigma^\omega$;

…

This step is somewhat dubious.

- Realistic systems don't realize every possible trace.

Why is φ_a not monitorable?

Theorem

Let $\varphi_a = \forall\pi.\exists\tau.\Box(\text{☕}_\pi \to \text{☕}_\tau)$. Then $U \not\models^s \varphi_a$ and $U \not\models^v \varphi_a$ for all $U \in \mathcal{P}_{fin}(\Sigma^*)$.

Proof.

$U \not\models^v \varphi_a$ $\quad U \preceq \Sigma^\omega$, and $\Sigma^\omega \models \varphi_a$ because ☕☕☕ $\cdots \in \Sigma^\omega$;

$\ldots$

This step is somewhat dubious.

- Realistic systems don't realize every possible trace.
- There is only a finite number of coffee dispensers in the world (sadly).

Why is φ_a not monitorable?

Theorem

Let $\varphi_a = \forall\pi.\exists\tau.\Box(\text{☕}_\pi \to \text{☕}_\tau)$. Then $U \not\models^s \varphi_a$ and $U \not\models^v \varphi_a$ for all $U \in \mathcal{P}_{\mathit{fin}}(\Sigma^*)$.

Proof.

$U \not\models^v \varphi_a$ $U \preceq \Sigma^\omega$, and $\Sigma^\omega \models \varphi_a$ because $\text{☕☕☕}\cdots \in \Sigma^\omega$;

. . .

This step is somewhat dubious.

- Realistic systems don't realize every possible trace.
- There is only a finite number of coffee dispensers in the world (sadly).

When monitoring hyperproperties, we'd like to take into account
some information about the system
(gray-box monitoring).

Gray-box monitoring of HyperLTL properties

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation U.

Definition

A finite observation $U \in \mathcal{P}_{\mathit{fin}}(\Sigma^*)$ permanently satisfies (resp. violates) φ, if every infinite extension of U satisfies (resp. violates) φ:

$$U \models^s \varphi \quad \text{iff} \quad \text{for all } T \in \mathcal{P}(\Sigma^\omega) \text{ such that } U \preceq T, T \models \varphi$$

$$U \models^v \varphi \quad \text{iff} \quad \text{for all } T \in \mathcal{P}(\Sigma^\omega) \text{ such that } U \preceq T, T \not\models \varphi$$

Gray-box monitoring of HyperLTL properties

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation U of a system $\mathcal{S}$.

Definition

A finite observation $U \in \mathcal{P}_{\mathit{fin}}(\Sigma^*)$ permanently satisfies (resp. violates) φ, if every infinite extension of U satisfies (resp. violates) φ:

$$U \models^s \varphi \quad \text{iff} \quad \text{for all } T \in \mathcal{P}(\Sigma^\omega) \text{ such that } U \preceq T,\ T \models \varphi$$

$$U \models^v \varphi \quad \text{iff} \quad \text{for all } T \in \mathcal{P}(\Sigma^\omega) \text{ such that } U \preceq T,\ T \not\models \varphi$$

Gray-box monitoring of HyperLTL properties

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation U of a system $\mathcal{S}$.

Definition

Given a set of system behaviors $\mathcal{S} \subseteq \mathcal{P}(\Sigma^\omega)$,
a finite observation $U \in \mathcal{P}_{\mathit{fin}}(\Sigma^*)$ permanently satisfies (resp. violates) φ,
if every infinite extension of U in $\mathcal{S}$ satisfies (resp. violates) φ:

$$U \models^s_{\mathcal{S}} \varphi \quad \text{iff} \quad \text{for all } T \in \mathcal{S} \text{ such that } U \preceq T,\ T \models \varphi$$

$$U \models^v_{\mathcal{S}} \varphi \quad \text{iff} \quad \text{for all } T \in \mathcal{S} \text{ such that } U \preceq T,\ T \not\models \varphi$$

Gray-box monitoring of HyperLTL properties

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation U of a system $\mathcal{S}$.

Definition

Given a set of system behaviors $\mathcal{S} \subseteq \mathcal{P}(\Sigma^\omega)$,
a finite observation $U \in \mathcal{P}_{fin}(\Sigma^*)$ permanently satisfies (resp. violates) φ,
if every infinite extension of U in $\mathcal{S}$ satisfies (resp. violates) φ:

$$U \models^s_{\mathcal{S}} \varphi \quad \text{iff} \quad \text{for all } T \in \mathcal{S} \text{ such that } U \preceq T,\ T \models \varphi$$
$$U \models^v_{\mathcal{S}} \varphi \quad \text{iff} \quad \text{for all } T \in \mathcal{S} \text{ such that } U \preceq T,\ T \not\models \varphi$$

$$\mathcal{S} = \{T \in \mathcal{P}(\Sigma^\omega) \mid |T| = 3\} \qquad U = \{☕☕◌◌◌,\ ◌◌☕☕◌,\ ☕☕☕☕◌\}$$

$$U \not\models^s \forall \pi. \exists \tau. \Box(◌_\pi \rightarrow ☕_\tau) \qquad U \models^v \forall \pi. \exists \tau. \Box(◌_\pi \rightarrow ☕_\tau)$$

Gray-box monitoring in general

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation O of a system $\mathcal{S}$.

Definition

Given a set of system behaviors $\mathcal{S} \subseteq \mathcal{B}$,
a finite observation $O \in \mathcal{O}$ permanently satisfies (resp. violates) φ,
if every infinite extension of O in $\mathcal{S}$ satisfies (resp. violates) φ:

$$
\begin{array}{lll}
O \models^{s}_{\mathcal{S}} \varphi & \text{iff} & \text{for all } B \in \mathcal{S} \text{ such that } O \preceq B,\ B \models \varphi \\
O \models^{v}_{\mathcal{S}} \varphi & \text{iff} & \text{for all } B \in \mathcal{S} \text{ such that } O \preceq B,\ B \not\models \varphi
\end{array}
$$

Gray-box monitoring in general

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (?), given a finite observation O of a system $\mathcal{S}$.

Definition

Given a set of system behaviors $\mathcal{S} \subseteq \mathcal{B}$,
a finite observation $O \in \mathcal{O}$ permanently satisfies (resp. violates) φ,
if every infinite extension of O in $\mathcal{S}$ satisfies (resp. violates) φ:

$$O \models^s_{\mathcal{S}} \varphi \quad \text{iff} \quad \text{for all } B \in \mathcal{S} \text{ such that } O \preceq B,\ B \models \varphi$$
$$O \models^v_{\mathcal{S}} \varphi \quad \text{iff} \quad \text{for all } B \in \mathcal{S} \text{ such that } O \preceq B,\ B \not\models \varphi$$

A formula φ is semantically gray-box monitorable for a system $\mathcal{S}$ if every observation O has an extension $P \succeq O$ in $\mathcal{S}$, such that $P \models^s_{\mathcal{S}} \varphi$ or $P \models^v_{\mathcal{S}} \varphi$.

Gray-box monitors for $\forall^+\exists^+$-properties

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation O of a system $\mathcal{S}$.

Gray-box monitors for $\forall^+\exists^+$-properties

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation O of a system $\mathcal{S}$.

A monitor for a property φ and a system $\mathcal{S}$ is a computable function $M_{\varphi,\mathcal{S}}\colon \mathcal{O} \to \{✓, ✗, \mathbf{?}\}$ that decides a verdict for φ given a finite O in $\mathcal{S}$.

Gray-box monitors for $\forall^+\exists^+$-properties

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation O of a system $\mathcal{S}$.

A monitor for a property φ and a system $\mathcal{S}$ is a computable function $M_{\varphi,\mathcal{S}}\colon \mathcal{O} \to \{✓, ✗, \textbf{?}\}$ that decides a verdict for φ given a finite O in $\mathcal{S}$.

Assuming $\varphi = \forall\pi.\exists\tau.\psi(\pi,\tau)$, and a sufficiently restrictive $\mathcal{S}$, we may be able to statically prove that all extensions $T \succeq U$ of a given U permanently violate φ.

Gray-box monitors for $\forall^+\exists^+$-properties

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation O of a system $\mathcal{S}$.

A monitor for a property φ and a system $\mathcal{S}$ is a computable function $M_{\varphi,\mathcal{S}} \colon \mathcal{O} \to \{✓, ✗, \textbf{?}\}$ that decides a verdict for φ given a finite O in $\mathcal{S}$.

Assuming $\varphi = \forall\pi.\exists\tau.\psi(\pi,\tau)$, and a sufficiently restrictive $\mathcal{S}$, we may be able to statically prove that all extensions $T \succeq U$ of a given U permanently violate φ.

Example: $\varphi_a = \forall\pi.\exists\tau.\Box(\text{☕}_\pi \to \text{☕}_\tau)$ $\qquad \mathcal{S} = \{T \in \mathcal{P}(\Sigma^\omega) \mid |T| = 3\}$

Gray-box monitors for $\forall^{+}\exists^{+}$-properties

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation O of a system $\mathcal{S}$.

A monitor for a property φ and a system $\mathcal{S}$ is a computable function $M_{\varphi,\mathcal{S}} \colon \mathcal{O} \to \{✓, ✗, \mathbf{?}\}$ that decides a verdict for φ given a finite O in $\mathcal{S}$.

Assuming $\varphi = \forall\pi.\exists\tau.\psi(\pi,\tau)$, and a sufficiently restrictive $\mathcal{S}$, we may be able to statically prove that all extensions $T \succeq U$ of a given U permanently violate φ.

Example: $\varphi_a = \forall\pi.\exists\tau.\square(\text{☕}_{\pi} \to \text{☕}_{\tau})$ $\qquad \mathcal{S} = \{T \in \mathcal{P}(\Sigma^{\omega}) \mid |T| = 3\}$

Negate φ_a: $\neg\varphi_a = \exists\pi.\neg\exists\tau.\square(\text{☕}_{\pi} \to \text{☕}_{\tau})$

Gray-box monitors for $\forall^+\exists^+$-properties

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation O of a system $\mathcal{S}$.

A monitor for a property φ and a system $\mathcal{S}$ is a computable function $M_{\varphi,\mathcal{S}} \colon \mathcal{O} \to \{✓, ✗, \textbf{?}\}$ that decides a verdict for φ given a finite O in $\mathcal{S}$.

Assuming $\varphi = \forall\pi.\exists\tau.\psi(\pi,\tau)$, and a sufficiently restrictive $\mathcal{S}$, we may be able to statically prove that all extensions $T \succeq U$ of a given U permanently violate φ.

Example: $\varphi_a = \forall\pi.\exists\tau.\square(\text{☕}_\pi \to \text{☕}_\tau)$ $\qquad \mathcal{S} = \{T \in \mathcal{P}(\Sigma^\omega) \mid |T| = 3\}$

Negate φ_a: $\neg\varphi_a = \exists\pi.\neg\exists\tau.\square(\text{☕}_\pi \to \text{☕}_\tau)$ $\qquad$ instantiate

Gray-box monitors for $\forall^+\exists^+$-properties

Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation O of a system $\mathcal{S}$.

A monitor for a property φ and a system $\mathcal{S}$ is a computable function $M_{\varphi,\mathcal{S}} \colon \mathcal{O} \to \{✓, ✗, \mathbf{?}\}$ that decides a verdict for φ given a finite O in $\mathcal{S}$.

Assuming $\varphi = \forall\pi.\exists\tau.\psi(\pi,\tau)$, and a sufficiently restrictive $\mathcal{S}$, we may be able to statically prove that all extensions $T \succeq U$ of a given U permanently violate φ.

Example: $\varphi_a = \forall\pi.\exists\tau.\Box(\text{☕}_\pi \to \text{☕}_\tau)$ $\qquad$ $\mathcal{S} = \{T \in \mathcal{P}(\Sigma^\omega) \mid |T| = 3\}$

Negate φ_a: $\neg\varphi_a = \exists\pi.\neg\exists\tau.\Box(\text{☕}_\pi \to \text{☕}_\tau)$ $\qquad$ instantiate $\qquad$ solve

Gray-box monitors for $\forall^+\exists^+$-properties

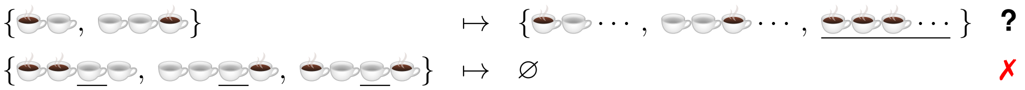
Monitoring: decide whether a given property φ is permanently satisfied (✓), violated (✗), or neither (**?**), given a finite observation O of a system $\mathcal{S}$.

A monitor for a property φ and a system $\mathcal{S}$ is a computable function $M_{\varphi,\mathcal{S}} \colon \mathcal{O} \to \{✓, ✗, \textbf{?}\}$ that decides a verdict for φ given a finite O in $\mathcal{S}$.

Assuming $\varphi = \forall\pi.\exists\tau.\psi(\pi,\tau)$, and a sufficiently restrictive $\mathcal{S}$, we may be able to statically prove that all extensions $T \succeq U$ of a given U permanently violate φ.

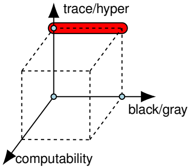
Example: $\varphi_a = \forall\pi.\exists\tau.\Box(\text{○}_\pi \to \text{☕}_\tau)$ $\qquad \mathcal{S} = \{T \in \mathcal{P}(\Sigma^\omega) \mid |T| = 3\}$

Negate φ_a: $\neg\varphi_a = \exists\pi.\neg\exists\tau.\Box(\text{○}_\pi \to \text{☕}_\tau)$ $\qquad$ instantiate $\qquad$ solve

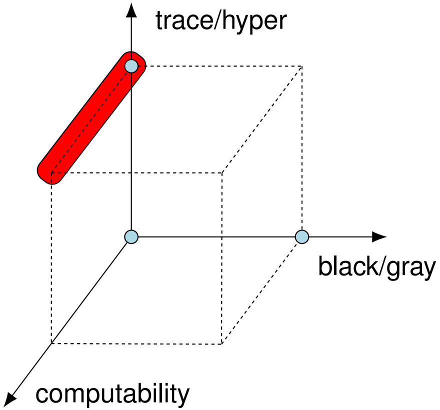


Gray-box monitoring – Summary

- Properties defined over observations (e.g. traces or sets of traces).
  $\Rightarrow$ Properties describe sets of observations.
- Sound and complete monitors can be constructed for some formulas.
  - For example, for formulas without quantifier alternations (as for black-box).
  - But also for $\forall^+\exists^+$-formulas when $\mathcal{S}$ imposes enough constraints.
- Monitorability of formulas depends on set of valid system behaviors $\mathcal{S}$.
  - For example, $\forall^+\exists^+$-properties are monitorable for some choices of $\mathcal{S}$.
  - We will see a more interesting example later...



Undecidable hyperproperties



Monitorability is not existence of monitors

A formula φ is semantically gray-box monitorable for a system $\mathcal{S}$ if every observation O has an extension $P \succeq O$ in $\mathcal{S}$, such that $P \models^{s}_{\mathcal{S}} \varphi$ or $P \models^{v}_{\mathcal{S}} \varphi$.

Monitorability is not existence of monitors

A formula φ is semantically gray-box monitorable for a system $\mathcal{S}$ if every observation O has an extension $P \succeq O$ in $\mathcal{S}$, such that $P \models^{s}_{\mathcal{S}} \varphi$ or $P \models^{v}_{\mathcal{S}} \varphi$.

A monitor for a property φ and a system $\mathcal{S}$ is a computable function $M_{\varphi,\mathcal{S}} \colon \mathcal{O}\{\checkmark, \text{✗}, \textbf{?}\}$ that decides a verdict for φ given a finite u.

Monitorability is not existence of monitors

A formula φ is semantically gray-box monitorable for a system $\mathcal{S}$ if every observation O has an extension $P \succeq O$ in $\mathcal{S}$, such that $P \models^{s}_{\mathcal{S}} \varphi$ or $P \models^{v}_{\mathcal{S}} \varphi$.

A monitor for a property φ and a system $\mathcal{S}$ is a computable function $M_{\varphi,\mathcal{S}} \colon \mathcal{O}\{✓, ✗, ?\}$ that decides a verdict for φ given a finite u.

Observation: Monitorability of φ in $\mathcal{S}$ does not guarantee the existence of a sound and complete monitor $M_{\varphi,\mathcal{S}}$.

Monitorability is not existence of monitors

A formula φ is semantically gray-box monitorable for a system $\mathcal{S}$ if every observation O has an extension $P \succeq O$ in $\mathcal{S}$, such that $P \models^s_{\mathcal{S}} \varphi$ or $P \models^v_{\mathcal{S}} \varphi$.

A monitor for a property φ and a system $\mathcal{S}$ is a computable function $M_{\varphi,\mathcal{S}} \colon \mathcal{O}\{✓, ✗, \textbf{?}\}$ that decides a verdict for φ given a finite u.

Observation: Monitorability of φ in $\mathcal{S}$ does not guarantee the existence of a sound and complete monitor $M_{\varphi,\mathcal{S}}$.

Example: Let T be some Turing machine.

$$\mathcal{S} = \{t \in \Sigma^\omega \mid t_i = \text{ the state of } T \text{ after } i \text{ steps}\}, \qquad \varphi = \Diamond\text{halt}.$$

Because T is deterministic, either $u \models^s_{\mathcal{S}} \varphi$ or $u \models^v_{\mathcal{S}} \varphi$, for any u in $\mathcal{S}$.

Monitorability is not existence of monitors

A formula φ is semantically gray-box monitorable for a system $\mathcal{S}$ if every observation O has an extension $P \succeq O$ in $\mathcal{S}$, such that $P \models^s_{\mathcal{S}} \varphi$ or $P \models^v_{\mathcal{S}} \varphi$.

A monitor for a property φ and a system $\mathcal{S}$ is a computable function $M_{\varphi,\mathcal{S}} \colon \mathcal{O}\{✓, ✗, ?\}$ that decides a verdict for φ given a finite u.

Observation: Monitorability of φ in $\mathcal{S}$ does not guarantee the existence of a sound and complete monitor $M_{\varphi,\mathcal{S}}$.

Example: Let T be some Turing machine.

$$\mathcal{S} = \{t \in \Sigma^\omega \mid t_i = \text{ the state of } T \text{ after } i \text{ steps}\}, \qquad \varphi = \Diamond\text{halt}.$$

Because T is deterministic, either $u \models^s_{\mathcal{S}} \varphi$ or $u \models^v_{\mathcal{S}} \varphi$, for any u in $\mathcal{S}$.

$\Rightarrow$ φ is monitorable in $\mathcal{S}$;

Monitorability is not existence of monitors

A formula φ is semantically gray-box monitorable for a system $\mathcal{S}$ if every observation O has an extension $P \succeq O$ in $\mathcal{S}$, such that $P \models^s_{\mathcal{S}} \varphi$ or $P \models^v_{\mathcal{S}} \varphi$.

A monitor for a property φ and a system $\mathcal{S}$ is a computable function $M_{\varphi,\mathcal{S}} \colon \mathcal{O}\{\checkmark, \text{✗}, \textbf{?}\}$ that decides a verdict for φ given a finite u.

Observation: Monitorability of φ in $\mathcal{S}$ does not guarantee the existence of a sound and complete monitor $M_{\varphi,\mathcal{S}}$.

Example: Let T be some Turing machine.

$$\mathcal{S} = \{t \in \Sigma^\omega \mid t_i = \text{ the state of } T \text{ after } i \text{ steps}\}, \qquad \varphi = \Diamond\text{halt}.$$

Because T is deterministic, either $u \models^s_{\mathcal{S}} \varphi$ or $u \models^v_{\mathcal{S}} \varphi$, for any u in $\mathcal{S}$.

$\Rightarrow$ φ is monitorable in $\mathcal{S}$;

$\Rightarrow$ but there is no sound and complete monitor $M_{\varphi,\mathcal{S}}$.

Monitorability is not existence of monitors

A formula φ is semantically gray-box monitorable for a system $\mathcal{S}$ if every observation O has an extension $P \succeq O$ in $\mathcal{S}$, such that $P \models^s_{\mathcal{S}} \varphi$ or $P \models^v_{\mathcal{S}} \varphi$.

A monitor for a property φ and a system $\mathcal{S}$ is a computable function $M_{\varphi,\mathcal{S}} : \mathcal{O}\{\checkmark, \text{✗}, \textbf{?}\}$ that decides a verdict for φ given a finite u.

Observation: Monitorability of φ in $\mathcal{S}$ does not guarantee the existence of a sound and complete monitor $M_{\varphi,\mathcal{S}}$.

Example: Let T be some Turing machine.

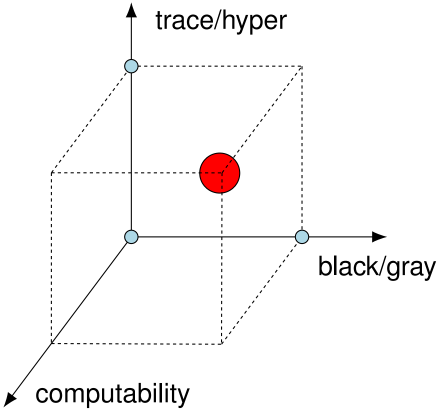
$$\mathcal{S} = \{t \in \Sigma^\omega \mid t_i = \text{ the state of } T \text{ after } i \text{ steps}\}, \qquad \varphi = \Diamond\text{halt}.$$

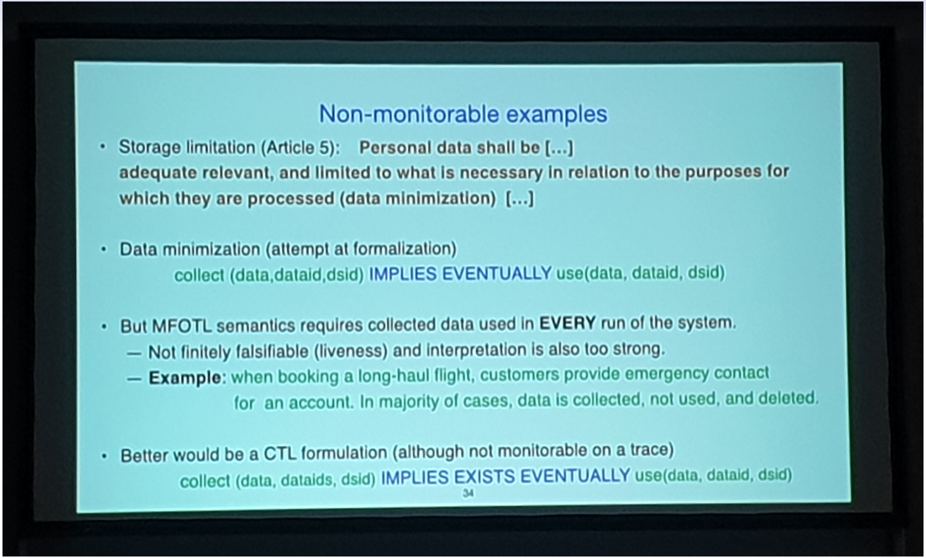
Because T is deterministic, either $u \models^s_{\mathcal{S}} \varphi$ or $u \models^v_{\mathcal{S}} \varphi$, for any u in $\mathcal{S}$.

$\Rightarrow$ φ is monitorable in $\mathcal{S}$;

$\Rightarrow$ *there is a sound monitor $M_{\varphi,\mathcal{S}}$ that only answers $\checkmark$ or* ***?****!*

Case study: distributed data minimality





Slide by David Basin, *Can we Verify GDPR Compliance?*, RV’19 keynote.

Case study: distributed data minimality

- Distributed data minimality (DDM)
  - privacy property (GDPR)
  - generalization of data minimality to a multi-input setting
  - $\forall\forall\exists\exists$-hyperproperty

$$\varphi_i \;=\; \forall\pi.\forall\pi'.\exists\tau.\exists\tau'.\, \neg\,\mathrm{same}_i(\pi,\pi') \rightarrow \begin{pmatrix} \mathrm{same}_i(\pi,\tau) \wedge \mathrm{same}_i(\pi',\tau') \wedge \\ \mathrm{almost}_i(\tau,\tau') \wedge \neg\,\mathrm{output}(\tau,\tau') \end{pmatrix}$$

Case study: distributed data minimality

- Distributed data minimality (DDM)
  - privacy property (GDPR)
  - generalization of data minimality to a multi-input setting
  - $\forall\forall\exists\exists$-hyperproperty

$$\varphi_i \;=\; \forall\pi.\forall\pi'.\exists\tau.\exists\tau'.\,\neg\,\mathrm{same}_i(\pi,\pi') \rightarrow \begin{pmatrix} \mathrm{same}_i(\pi,\tau) \wedge \mathrm{same}_i(\pi',\tau') \,\wedge \\ \mathrm{almost}_i(\tau,\tau') \wedge \neg\,\mathrm{output}(\tau,\tau') \end{pmatrix}$$

- Challenges:
  - Not black-box monitorable.
  - Undecidable.
  - Defined over arbitrary domains/datatypes.

Case study: distributed data minimality

- Distributed data minimality (DDM)
  - privacy property (GDPR)
  - generalization of data minimality to a multi-input setting
  - $\forall\forall\exists\exists$-hyperproperty

$$\varphi_i \;=\; \forall\pi.\forall\pi'.\exists\tau.\exists\tau'.\, \neg\,\mathrm{same}_i(\pi,\pi') \to \begin{pmatrix} \mathrm{same}_i(\pi,\tau) \wedge \mathrm{same}_i(\pi',\tau') \;\wedge \\ \mathrm{almost}_i(\tau,\tau') \wedge \neg\,\mathrm{output}(\tau,\tau') \end{pmatrix}$$

- Challenges:
  - Not black-box monitorable.
  - Undecidable.
  - Defined over arbitrary domains/datatypes.

Yet, we have a monitor. . .

Case study: distributed data minimality

- Distributed data minimality (DDM)
  - privacy property (GDPR)
  - generalization of data minimality to a multi-input setting
  - $\forall\forall\exists\exists$-hyperproperty

$$\varphi_i = \forall\pi.\forall\pi'.\exists\tau.\exists\tau'.\ \neg\,\mathrm{same}_i(\pi,\pi') \to \begin{pmatrix} \mathrm{same}_i(\pi,\tau) \wedge \mathrm{same}_i(\pi',\tau') \wedge \\ \mathrm{almost}_i(\tau,\tau') \wedge \neg\,\mathrm{output}(\tau,\tau') \end{pmatrix}$$

- Challenges:
  - Not black-box monitorable.
  - Undecidable.
  - Defined over arbitrary domains/datatypes.

Yet, we have a monitor...

here's how...

Distributed data minimality

Definition (Antignac, Sands & Schneider, 2017)

A function f is distributed data-minimal (DDM) if, for all input positions k and all $x, y \in I_k$ such that $x \neq y$, there is some $z \in I$, such that $f(z[k \mapsto x]) \neq f(z[k \mapsto y])$.

Distributed data minimality

$$\varphi_i \;=\; \forall\pi.\forall\pi'.\exists\tau.\exists\tau'.\, \neg\, \mathrm{same}_i(\pi,\pi') \rightarrow \begin{pmatrix} \mathrm{same}_i(\pi,\tau) \wedge \mathrm{same}_i(\pi',\tau') \,\wedge \\ \mathrm{almost}_i(\tau,\tau') \wedge \neg\, \mathrm{output}(\tau,\tau') \end{pmatrix}$$

$$\varphi_{\mathsf{dm}} \;=\; \bigwedge_{i=1}^{n} \varphi_i, \qquad \Sigma_f^{\#} = \{(x,y) \mid f(x) = y\}, \qquad \mathcal{S}_f = \mathcal{P}(\Sigma_f^{\#})$$

Distributed data minimality

$$\varphi_i = \forall \pi . \forall \pi' . \exists \tau . \exists \tau' . \neg \mathrm{same}_i(\pi, \pi') \rightarrow \begin{pmatrix} \mathrm{same}_i(\pi, \tau) \wedge \mathrm{same}_i(\pi', \tau') \wedge \\ \mathrm{almost}_i(\tau, \tau') \wedge \neg \mathrm{output}(\tau, \tau') \end{pmatrix}$$

$$\varphi_{\text{dm}} = \bigwedge_{i=1}^{n} \varphi_i, \qquad \Sigma_f^{\#} = \{(x, y) \mid f(x) = y\}, \qquad \mathcal{S}_f = \mathcal{P}(\Sigma_f^{\#})$$

Using the generalized framework

- Set of observable behaviors $\mathcal{O} = \Sigma_f^{\#}$ are valid function applications.

Distributed data minimality

$$\varphi_i \;=\; \forall\pi.\forall\pi'.\exists\tau.\exists\tau'.\;\neg\,\mathrm{same}_i(\pi,\pi') \rightarrow \begin{pmatrix} \mathrm{same}_i(\pi,\tau) \land \mathrm{same}_i(\pi',\tau') \land \\ \mathrm{almost}_i(\tau,\tau') \land \neg\,\mathrm{output}(\tau,\tau') \end{pmatrix}$$

$$\varphi_{\mathsf{dm}} \;=\; \bigwedge_{i=1}^{n} \varphi_i, \qquad \Sigma_f^{\#} = \{(x,y) \mid f(x) = y\}, \qquad \mathcal{S}_f = \mathcal{P}(\Sigma_f^{\#})$$

Using the generalized framework

- Set of observable behaviors $\mathcal{O} = \Sigma_f^{\#}$ are valid function applications.
- Not black-box monitorable.

Distributed data minimality

$$\varphi_i \;=\; \forall\pi.\forall\pi'.\exists\tau.\exists\tau'.\;\neg\,\mathrm{same}_i(\pi,\pi') \rightarrow \begin{pmatrix} \mathrm{same}_i(\pi,\tau) \wedge \mathrm{same}_i(\pi',\tau') \wedge \\ \mathrm{almost}_i(\tau,\tau') \wedge \neg\,\mathrm{output}(\tau,\tau') \end{pmatrix}$$

$$\varphi_{\text{dm}} \;=\; \bigwedge_{i=1}^{n}\varphi_i, \qquad \Sigma_f^{\#} = \{(x,y) \mid f(x) = y\}, \qquad \mathcal{S}_f = \mathcal{P}(\Sigma_f^{\#})$$

Using the generalized framework

- Set of observable behaviors $\mathcal{O} = \Sigma_f^{\#}$ are valid function applications.
- Not black-box monitorable, but gray-box monitorable (thanks to $\mathcal{S}$).

A sound monitor for distributed data minimality

$$\varphi_i \;=\; \forall\pi.\forall\pi'.\exists\tau.\exists\tau'.\ \neg\,\mathrm{same}_i(\pi,\pi') \rightarrow \begin{pmatrix} \mathrm{same}_i(\pi,\tau) \wedge \mathrm{same}_i(\pi',\tau') \wedge \\ \mathrm{almost}_i(\tau,\tau') \wedge \neg\,\mathrm{output}(\tau,\tau') \end{pmatrix}$$

$$\varphi_{\textsf{dm}} \;=\; \bigwedge_{i=1}^{n} \varphi_i, \qquad \Sigma_f^{\#} = \{(x,y) \mid f(x) = y\}, \qquad \mathcal{S}_f = \mathcal{P}(\Sigma_f^{\#})$$

A sound monitor for distributed data minimality

$$\varphi_i \;=\; \forall\pi.\forall\pi'.\exists\tau.\exists\tau'.\; \neg\,\mathrm{same}_i(\pi,\pi') \rightarrow \begin{pmatrix} \mathrm{same}_i(\pi,\tau) \wedge \mathrm{same}_i(\pi',\tau') \,\wedge \\ \mathrm{almost}_i(\tau,\tau') \wedge \neg\,\mathrm{output}(\tau,\tau') \end{pmatrix}$$

$$\varphi_{\text{dm}} \;=\; \bigwedge_{i=1}^{n} \varphi_i, \qquad \Sigma_f^{\#} = \{(x,y) \mid f(x) = y\}, \qquad \mathcal{S}_f = \mathcal{P}(\Sigma_f^{\#})$$

We build a monitor

$$M_{\text{dm}}(U) = \begin{cases} \textbf{?} & \text{if } f(u_{in}) \neq u_{out} \text{ for some } u \in U, \\ \textbf{?} & \text{if } \bigwedge_{i=1}^{n} \bigwedge_{u,u' \in U} N_{f,i}(\mathrm{proj}_i(u_{in}), \mathrm{proj}_i(u'_{in})) \neq \text{✗}, \\ \text{✗} & \text{otherwise.} \end{cases}$$

A sound monitor for distributed data minimality

$$\varphi_i \;=\; \forall\pi.\forall\pi'.\exists\tau.\exists\tau'.\ \neg\,\mathrm{same}_i(\pi,\pi') \rightarrow \begin{pmatrix} \mathrm{same}_i(\pi,\tau) \wedge \mathrm{same}_i(\pi',\tau') \wedge \\ \mathrm{almost}_i(\tau,\tau') \wedge \neg\,\mathrm{output}(\tau,\tau') \end{pmatrix}$$

$$\varphi_{\mathsf{dm}} \;=\; \bigwedge_{i=1}^{n} \varphi_i, \qquad \Sigma_f^{\#} = \{(x,y) \mid f(x) = y\}, \qquad \mathcal{S}_f = \mathcal{P}(\Sigma_f^{\#})$$

We build a monitor

$$M_{\mathsf{dm}}(U) = \begin{cases} \textbf{?} & \text{if } f(u_{in}) \neq u_{out} \text{ for some } u \in U, \\ \textbf{?} & \text{if } \bigwedge_{i=1}^{n} \bigwedge_{u,u' \in U} N_{f,i}(\mathrm{proj}_i(u_{in}), \mathrm{proj}_i(u'_{in})) \neq ✗, \\ ✗ & \text{otherwise.} \end{cases}$$

using an oracle $N_{f,i}(x,y)$ (implemented as symbolic execution + SMT solver):

$$N_{f,i}(x,y) = \begin{cases} ✓ \text{ or } \textbf{?} & \text{if } \exists z \in I. f(z[i \mapsto x]) \neq f(z[i \mapsto y]), \\ ✗ \text{ or } \textbf{?} & \text{otherwise.} \end{cases}$$

A sound monitor for distributed data minimality

We build a monitor

$$M_{\mathsf{dm}}(U) = \begin{cases} \textbf{?} & \text{if } f(u_{\mathit{in}}) \neq u_{\mathit{out}} \text{ for some } u \in U, \\ \textbf{?} & \text{if } \bigwedge_{i=1}^{n} \bigwedge_{u,u' \in U} N_{f,i}(\mathrm{proj}_i(u_{\mathit{in}}), \mathrm{proj}_i(u'_{\mathit{in}})) \neq \text{✗}, \\ \text{✗} & \text{otherwise.} \end{cases}$$

using an oracle $N_{f,i}(x,y)$ (implemented as symbolic execution + SMT solver):

$$N_{f,i}(x,y) = \begin{cases} \checkmark \text{ or } \textbf{?} & \text{if } \exists z \in I.f(z[i \mapsto x]) \neq f(z[i \mapsto y]), \\ \text{✗ or } \textbf{?} & \text{otherwise.} \end{cases}$$

The monitor is sound but not complete.

Try it out!

https://github.com/sstucki/minion/

Thank you!

Coauthors

- César Sánchez, IMDEA SW
- Borzoo Bonakdarpour, ISU
- Gerardo Schneider, GU/Chalmers

UNIVERSITY OF GOTHENBURG

CHALMERS UNIVERSITY OF TECHNOLOGY

institute imdea software

IOWA STATE UNIVERSITY

Checkout the `minion` monitor for data minimality

`https://github.com/sstucki/minion/`

Shreya Agrawal and Borzoo Bonakdarpour.
Runtime verification of k-safety hyperproperties in HyperLTL.
In *Proc. of the IEEE 29th Computer Security Foundations (CSF'16)*, pages 239–252. IEEE CS Press, 2016.

Andreas Bauer, Martin Leucker, and Chrisitan Schallhart.
Runtime verification for LTL and TLTL.
ACM T. Softw. Eng. Meth., 20(4):14, 2011.

Andreas Bauer, Martin Leucker, and Christian Schallhart.
The good, the bad, and the ugly—but how ugly is ugly?
In *Proc. of the 7th Int'l Workshop on Runtime Verification (RV'07)*, volume 4839 of *LNCS*, pages 126–138. Springer, 2007.

Borzoo Bonakdarpour, César Sánchez, and Gerardo Schneider.
Monitoring hyperproperties by combining static analysis and runtime verification.

In *Proc. of the 8th Int'l Symp. on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA'2018). Verification. Part II*, volume 11245 of *LNCS*, pages 8–27. Springer, 2018.

Yliès Falcone, Jean-Claude Fernandez, and Laurent Mounier.
What can you verify and enforce at runtime?
International Journal on Software Tools for Technology Transfer (STTT), 14(3):349–382, 2012.

Bernd Finkbeiner, Christopher Hahn, Marvin Stenger, and Leander Tentrup.
Monitoring hyperproperties.
In *Proc. of 17th Int'l Conf. on Runtime Verification (RV'17)*, volume 10548 of *LNCS*, pages 190–207. Springer, 2017.

Christopher Hahn.
Algorithms for monitoring hyperproperties.
In Bernd Finkbeiner and Leonardo Mariani, editors, *Runtime Verification*, pages 70–90, Cham, 2019. Springer International Publishing.

Klaus Havelund and Doron Peled.
Runtime verification: From propositional to first-order temporal logic.
In *Proc. of the 18th Int'l Conf. on Runtime Verification (RV'18)*, volume 11237 of *LNCS*, pages 90–112. Springer, 2018.

Amir Pnueli and Aleksandr Zaks.
PSL model checking and run-time verification via testers.
In *Proc. of the 14th Int'l Symp on Formal Methods (FM'06)*, volume 4085 of *LNCS*, pages 573–586. Springer, 2006.

Xian Zhang, Martin Leucker, and Wei Dong.
Runtime verification with predictive semantics.
In *Proc. of 4th NASA Int'l Symp on Formal Methods (NFM'12)*, volume 7226 of *LNCS*, pages 418–432. Springer, 2012.