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 \(^3\)ISU, USA

FM ’19, Porto, Portugal, 11 October 2019

sandro.stucki@gu.se  @stuckintheory
The monitorability cube

- Trace/hyper: [1, 4, 8]
- Black/gray: [2, 3, 5, 6, 9, 10]
- Computability
Motivation: distributed data minimality

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

\[
\forall \exists \forall\exists \pi. \forall \pi'. \exists \tau. \exists \tau'. \neg \text{same}_i(\pi, \pi') \rightarrow (\text{same}_i(\pi, \tau) \land \text{same}_i(\pi', \tau') \land \text{almost}_i(\tau, \tau') \land \neg \text{output}(\tau, \tau'))
\]

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

Yet, we have a monitor... what's going on here?
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

\[
\varphi_i = \forall \pi. \forall \pi'. \exists \tau. \exists \tau'. \neg \text{same}_i(\pi, \pi') \rightarrow \left( \begin{aligned}
\text{same}_i(\pi, \tau) & \land \text{same}_i(\pi', \tau') & \land \\
\text{almost}_i(\tau, \tau') & \land \neg \text{output}(\tau, \tau')
\end{aligned} \right)
\]
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

$$\varphi_i = \forall \pi. \forall \pi'. \exists \tau. \exists \tau'. \neg \text{same}_i(\pi, \pi') \rightarrow \left( \begin{array}{l}
\text{same}_i(\pi, \tau) \land \text{same}_i(\pi', \tau') \land \\
\text{almost}_i(\tau, \tau') \land \neg \text{output}(\tau, \tau')
\end{array} \right)$$

- 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

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

- 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

$$\varphi_i = \forall\pi. \forall\pi'. \exists\exists\tau. \exists\exists\tau'. \neg\text{same}_i(\pi, \pi') \rightarrow \left( \begin{array}{c}
\text{same}_i(\pi, \tau) \land \text{same}_i(\pi', \tau') \land \\
\text{almost}_i(\tau, \tau') \land \neg\text{output}(\tau, \tau')
\end{array} \right)$$

- 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

\[ \varphi_s = \square \text{coffee} \quad \varphi_l = \Diamond \text{coffee} \quad \varphi_r = \square \Diamond \text{coffee} \]
Trace properties – LTL

\[ \varphi_s = \square \] \hspace{1cm} \varphi_l = \lozenge \hspace{1cm} \varphi_r = \square \lozenge \]

\[ t_1 = \cdots \] \hspace{1cm} \models t_1 \varphi_s \hspace{1cm} \models t_1 \varphi_l \hspace{1cm} \models t_1 \varphi_r \\
\[ t_2 = \cdots \] \hspace{1cm} \not\models t_2 \varphi_s \hspace{1cm} \models t_2 \varphi_l \hspace{1cm} \not\models t_2 \varphi_r \\
\[ t_3 = \cdots \] \hspace{1cm} \not\models t_3 \varphi_s \hspace{1cm} \models t_3 \varphi_l \hspace{1cm} \models t_3 \varphi_r \]
Trace properties – LTL

ϕ_s = □
ϕ_l = ◇
ϕ_r = □◇

\begin{align*}
t_1 &= \cdots \\
t_2 &= \cdots \\
t_3 &= \cdots 
\end{align*}

\begin{align*}
t_1 \models \varphi_s & \quad t_1 \not\models \varphi_s \\
t_2 \not\models \varphi_s & \quad t_2 \models \varphi_l \\
t_3 \not\models \varphi_s & \quad t_3 \models \varphi_l 
\end{align*}

ϕ ::= a | \neg \varphi | \varphi \lor \varphi | \varphi U \varphi \\
\Diamond \varphi \equiv \text{true} U \varphi \\
\Box \varphi \equiv \neg \Diamond \neg \varphi \\

\begin{align*}
t \models p & \quad \text{iff} \quad p \in t[0] \\
t \models \neg \varphi & \quad \text{iff} \quad t \not\models \varphi \\
t \models \varphi_1 \lor \varphi_2 & \quad \text{iff} \quad t \models \varphi_1 \text{ or } t \models \varphi_2 \\
t \models \varphi & \quad \text{iff} \quad t[1,..] \models \varphi \\
t \models \varphi_1 U \varphi_2 & \quad \text{iff} \quad \text{for some } i, t[i,..] \models \varphi_2 \text{ and for all } j < i, t[j,..] \models \varphi_1
Monitoring LTL

\( \varphi_s = \square \)  \( \varphi_l = \Diamond \)  \( \varphi_r = \square \Diamond \)

Observation: the world today at 10am

\( u_{10} = \)

Update: the world at 11am

\( u_{11} = \varphi_s \)

Is there always coffee?

\( u_{10} \rightarrow ?, u_{11} \rightarrow \varphi_l \)

Is there eventually coffee?

\( u_{10} \rightarrow ?, u_{11} \rightarrow \varphi_r \)

Is there always eventually coffee?
Monitoring LTL

\( \varphi_s = \square \) \hspace{1cm} \( \varphi_l = \Diamond \) \hspace{1cm} \( \varphi_r = \square \Diamond \)

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

\( u_{10} = \)

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

\( u_{11} = \)

Is there always coffee?

\( u_{10} \rightarrow \)  ,  \( u_{11} \rightarrow \)

Is there eventually coffee?

\( u_{10} \rightarrow \)  ,  \( u_{11} \rightarrow \)

Is there always eventually coffee?

\( u_{10} \rightarrow \)  ,  \( u_{11} \rightarrow \)
Monitoring LTL

\[ \varphi_s = \square \quad \varphi_l = \Diamond \quad \varphi_r = \square \Diamond \]

- **Observation**: the world today at 10am
  \[ u_{10} = \square \square \square \square \square \]

- **Update**: the world at 11am
  \[ u_{11} = \square \square \square \square \square \square \square \square \square \square \]
Monitoring LTL

\[ \varphi_s = \square \quad \varphi_l = \Diamond \quad \varphi_r = \square \Diamond \]

- **Observation:** the world today at 10am
  
  \[ u_{10} = \text{coffee} \]

- **Update:** the world at 11am
  
  \[ u_{11} = \text{coffee} \]

\[ \varphi_s \text{ Is there always coffee?} \]
Monitoring LTL

\[ \varphi_s = \square \quad \varphi_l = \Diamond \quad \varphi_r = \square \Diamond \]

- **Observation:** the world today at 10am
  \[ u_{10} = \]

- **Update:** the world at 11am
  \[ u_{11} = \]

\[ \varphi_s \text{ Is there always coffee?} \quad u_{10} \rightarrow ? \]
Monitoring LTL

\[ \varphi_s = \square \quad \varphi_l = \lozenge \quad \varphi_r = \square \lozenge \]

- **Observation:** the world today at 10am
  \[ u_{10} = \]

- **Update:** the world at 11am
  \[ u_{11} = \]

\[ \varphi_s \text{ Is there always coffee?} \quad u_{10} \rightarrow ?, \quad u_{11} \rightarrow x \]
Monitoring LTL

\[ \varphi_s = \square \quad \varphi_l = \Diamond \quad \varphi_r = \square \Diamond \]

- **Observation:** the world today at 10am
  
  \[ u_{10} = \]

- **Update:** the world at 11am
  
  \[ u_{11} = \]

\[ \varphi_s \text{ Is there always coffee?} \quad u_{10} \rightarrow ? \text{, } u_{11} \rightarrow \times \]

\[ \varphi_l \text{ Is there eventually coffee?} \quad u_{10} \rightarrow \checkmark \text{, } u_{11} \rightarrow \checkmark \]
Monitoring LTL

\[ \varphi_s = \square \quad \varphi_l = \Diamond \quad \varphi_r = \square \Diamond \]

- **Observation**: the world today at 10am
  \[ u_{10} = \text{coffee} \]

- **Update**: the world at 11am
  \[ u_{11} = \text{coffee} \]

\[ \varphi_s \quad \text{Is there always coffee?} \quad u_{10} \rightarrow ?, \quad u_{11} \rightarrow \times \]
\[ \varphi_l \quad \text{Is there eventually coffee?} \quad u_{10} \rightarrow \checkmark, \quad u_{11} \rightarrow \checkmark \]
\[ \varphi_r \quad \text{Is there always eventually coffee?} \quad u_{10} \rightarrow ?, \quad u_{11} \rightarrow ? \]
Monitoring LTL

**Monitoring**: decide whether a given property $\varphi$ is **permanently** satisfied (✔), violated (✗), or neither (؟), at runtime.
Monitoring LTL

Monitoring: decide whether a given property $\varphi$ is permanently satisfied (✓), violated (✗), or neither (❓), given a finite observation $u$. 

Definition

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

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

$u \models v \varphi$ iff for all $t \in \Sigma^\omega$ such that $u \preceq t$, $t \not\models \varphi$
Monitoring LTL

Monitoring: decide whether a given property $\varphi$ is permanently satisfied (√), violated (✗), or neither (?), given a finite observation $u$.

Definition

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

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

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

Monitoring: decide whether a given property $\varphi$ is permanently satisfied (✓), violated (✗), or neither (❓), given a finite observation $u$.

**Definition**

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

- $u \vDash^s \varphi$ if $\forall t \in \Sigma^\omega$ such that $u \preceq t$, $t \vDash \varphi$
- $u \vDash^v \varphi$ if $\forall t \in \Sigma^\omega$ such that $u \preceq t$, $t \nolhd \varphi$

$$u_{11} = \ldots$$

$$u_{11} \nolhd^s \square$$
$$u_{11} \vDash^v \square$$
$$u_{11} \vDash^s \Diamond$$
$$u_{11} \nolhd^v \Diamond$$
Monitors for LTL

Monitoring: decide whether a given property $\varphi$ is permanently satisfied ($\checkmark$), violated ($\times$), or neither ($?$), given a finite observation $u$. 

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

Monitoring: decide whether a given property $\varphi$ is permanently satisfied (✓), violated (✗), or neither (؟), given a finite observation $u$.

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

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

Monitoring: decide whether a given property $\varphi$ is permanently satisfied ($\checkmark$), violated ($\times$), or neither ($?$), given a finite observation $u$.

A monitor for a property $\varphi$ is a computable function $M_\varphi : \Sigma^* \to \{\checkmark, \times, ?\}$ that decides a verdict for $\varphi$ given a finite $u$.

The monitor $M_\varphi$ is sound if

$$u \models^s \varphi \quad \text{if} \quad M_\varphi(u) = \checkmark, \quad u \models^v \varphi \quad \text{if} \quad M_\varphi(u) = \times$$
Monitors for LTL

Monitoring: decide whether a given property $\varphi$ is permanently satisfied ($\checkmark$), violated ($\times$), or neither ($?$), given a finite observation $u$.

A monitor for a property $\varphi$ is a computable function $M_\varphi : \Sigma^* \rightarrow \{\checkmark, \times, ?\}$ that decides a verdict for $\varphi$ given a finite $u$.

The monitor $M_\varphi$ is sound if

$$u \models^s \varphi \quad \text{if} \quad M_\varphi(u) = \checkmark, \quad u \models^v \varphi \quad \text{if} \quad M_\varphi(u) = \times$$

The monitor $M_\varphi$ is complete if

$$M_\varphi(u) = \checkmark \quad \text{if} \quad u \models^s \varphi, \quad M_\varphi(u) = \times \quad \text{if} \quad u \models^v \varphi, \quad M_\varphi(u) = ? \quad \text{o/w}.$$
Monitors for LTL

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

A monitor for a property \( \varphi \) is a computable function \( M_\varphi: \Sigma^* \to \{\checkmark, \times, ?\} \) that decides a verdict for \( \varphi \) given a finite \( u \).

The monitor \( M_\varphi \) is sound if

\[
\text{if } M_\varphi(u) = \checkmark, \quad \text{then } u \models^s \varphi \\
\text{if } M_\varphi(u) = \times, \quad \text{then } u \models^v \varphi
\]

The monitor \( M_\varphi \) is complete if

\[
M_\varphi(u) = \checkmark \text{ if } u \models^s \varphi, \quad M_\varphi(u) = \times \text{ if } u \models^v \varphi, \quad M_\varphi(u) = ? \text{ o/w.}
\]

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

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

$$\varphi_r = \Box \Diamond \frown$$

$u_{11} = 111111111111111$

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

$u_{11} \not\models^v \varphi_r$
Monitorability of LTL formulas

Monitoring: decide whether a given property $\varphi$ is permanently satisfied (√), violated (×), or neither (?), given a finite observation $u$.

$$\varphi_r = \Box \Diamond \ \ \ \ \ u_{11} = \text{observation}$$

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

Observation: $u \not\models^s \varphi_r$ and $u \not\models^v \varphi_r$ for any $u$. 
Monitorability of LTL formulas

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

$\varphi_r = \blacksquare \lozenge \lozenge$  

$u_{11} = \blacksquare \blacksquare \blacksquare \blacksquare \blacksquare \blacksquare \blacksquare \blacksquare \blacksquare \blacksquare \blacksquare \blacksquare \blacksquare

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

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

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

*There’s no point in monitoring $\varphi_r$!*
Monitorability of LTL formulas

Monitoring: decide whether a given property $\varphi$ is permanently satisfied (√), violated (✗), or neither (¿), given a finite observation $u$.

$$\varphi_r = \square \Diamond \varphi$$

$$u_{11} = \text{observation}$$

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

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

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

*There’s no point in monitoring $\varphi_r$!*

Definition (Pnueli & Zaks 2006)

A formula $\varphi$ 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.  
  ⇒ 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 (□◇) are not.

---


... and many more!
LTL – Summary

• Properties defined over individual traces.
  ⇒ 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 (□◇) are not.


... and many more!
Hyperproperties – HyperLTL

HyperLTL involves concepts such as trace/hyper and black/gray, along with computability.
Hyperproperties – HyperLTL

\[ \varphi_u = \forall \pi. \forall \tau. \Box (\pi \rightarrow \tau) \quad \varphi_a = \forall \pi. \exists \tau. \Box (\pi \rightarrow \tau) \]
Hyperproperties – HyperLTL

\[ \varphi_u = \forall \pi. \forall \tau. \Box (\pi \rightarrow \tau) \quad \varphi_a = \forall \pi. \exists \tau. \Box (\pi \rightarrow \tau) \]

\[ T_1 = \{ \ldots \} \quad T_1 \models \varphi_u \quad T_1 \models \varphi_a \]

\[ T_2 = \{ \ldots , \ldots \} \quad T_2 \not\models \varphi_u \quad T_2 \models \varphi_a \]

\[ T_3 = \{ \ldots , \ldots , \ldots \} \quad T_3 \not\models \varphi_u \quad T_3 \not\models \varphi_a \]
Hyperproperties – HyperLTL

$$\varphi_u = \forall \pi. \forall \tau. \Box (\pi \rightarrow \tau)$$  \hspace{1cm}  $$\varphi_a = \forall \pi. \exists \tau. \Box (\pi \rightarrow \tau)$$

$$T_1 = \{\text{...}\}$$  \hspace{1cm}  $$T_1 \models \varphi_u$$  \hspace{1cm}  $$T_1 \models \varphi_a$$

$$T_2 = \{\text{...} \text{, } \text{...} \text{, } \text{...} \}$$  \hspace{1cm}  $$T_2 \nmodels \varphi_u$$  \hspace{1cm}  $$T_2 \models \varphi_a$$

$$T_3 = \{\text{...} \text{, } \text{...} \text{, } \text{...} \text{, } \text{...} \}$$  \hspace{1cm}  $$T_3 \nmodels \varphi_u$$  \hspace{1cm}  $$T_3 \nmodels \varphi_a$$

$$\varphi ::= \forall \pi. \varphi \mid \exists \pi. \varphi \mid \psi$$  \hspace{1cm}  $$\psi ::= a_{\pi} \mid \neg \psi \mid \psi \lor \psi \mid \Box \psi \mid \psi U \psi$$

$$\Pi \models a_{\pi}$$  \hspace{1cm}  iff  \hspace{1cm}  $$a \in \Pi(\pi)[0]$$

$$\Pi \models \psi_1 \lor \psi_2$$  \hspace{1cm}  iff  \hspace{1cm}  $$\Pi \models \psi_1$$ or $$\Pi \models \psi_2$$

$$\Pi \models \neg \psi$$  \hspace{1cm}  iff  \hspace{1cm}  $$\Pi \nmodels \psi$$

$$\Pi \models \Box \psi$$  \hspace{1cm}  iff  \hspace{1cm}  $$\Pi[1..] \models \psi$$

$$\Pi \models \psi_1 U \psi_2$$  \hspace{1cm}  iff  \hspace{1cm}  for some $$i$$, $$\Pi[i, ..] \models \psi_2$$, and for all $$j < i$$, $$T, \Pi[j, ..] \models \psi_1$$

$$T, \Pi \models \forall \pi. \varphi$$  \hspace{1cm}  iff  \hspace{1cm}  $$T, \Pi[\pi \rightarrow t] \models \varphi$$ for all $$t \in T$$

$$T, \Pi \models \exists \pi. \varphi$$  \hspace{1cm}  iff  \hspace{1cm}  $$T, \Pi[\pi \rightarrow t] \models \varphi$$ for some $$t \in T$$

$$T, \Pi \models \psi$$  \hspace{1cm}  iff  \hspace{1cm}  $$\Pi \models \psi$$
Monitoring HyperLTL

\[ \varphi_u = \forall \pi. \forall \tau. \square (\pi \rightarrow \tau) \quad \varphi_a = \forall \pi. \exists \tau. \square (\pi \rightarrow \tau) \]
Monitoring HyperLTL

\[ \varphi_u = \forall \pi. \forall \tau. \Box (\pi \rightarrow \tau) \quad \varphi_a = \forall \pi. \exists \tau. \Box (\pi \rightarrow \tau) \]

- **Observation**: the world today at 10am
  
  \[ U_{10} = \{ \text{coffee mugs} \} \]
Monitoring HyperLTL

\[
\varphi_u = \forall \pi. \forall \tau. \square (\pi \rightarrow \tau) \quad \varphi_a = \forall \pi. \exists \tau. \square (\pi \rightarrow \tau)
\]

- **Observation:** the world today at 10am
  
  \[U_{10} = \{\text{coffee everywhere}\}\]

- **Update:** the world at 11am
  
  \[U_{11} = \{\text{coffee everywhere}, \text{coffee everywhere}\}\]

\[\begin{align*}
\text{Observation: the world today at 10am} \\
U_{10} = \{\text{coffee everywhere}\}\end{align*}\]

\[\begin{align*}
\text{Update: the world at 11am} \\
U_{11} = \{\text{coffee everywhere}, \text{coffee everywhere}\}\end{align*}\]
Monitoring HyperLTL

\[ \varphi_u = \forall \pi. \forall \tau. \Box (\pi \rightarrow \tau) \quad \varphi_a = \forall \pi. \exists \tau. \Box (\pi \rightarrow \tau) \]

- **Observation:** the world today at 10am
  
  \[ U_{10} = \{ \text{coffee} \} \]

- **Update:** the world at 11am
  
  \[ U_{11} = \{ \text{coffee}, \text{coffee}, \text{coffee} \} \]
Monitoring HyperLTL

\[ \varphi_u = \forall \pi. \forall \tau. \Box (\pi \rightarrow \tau) \quad \varphi_a = \forall \pi. \exists \tau. \Box (\pi \rightarrow \tau) \]

- **Observation:** the world today at 10am
  \[ U_{10} = \{\text{coffee everywhere}\} \]

- **Update:** the world at 11am
  \[ U_{11} = \{\text{coffee everywhere}, \text{coffee on a shelf}, \text{coffee on a table}\} \]

\[ \varphi_u \text{ Is there always coffee everywhere at the same time?} \]
Monitoring HyperLTL

\( \varphi_u = \forall \pi. \forall \tau. \Box (\pi \rightarrow \tau) \quad \varphi_a = \forall \pi. \exists \tau. \Box (\pi \rightarrow \tau) \)

- **Observation:** the world today at 10am
  
  \( U_{10} = \{ \text{coffee everywhere} \} \)

- **Update:** the world at 11am
  
  \( U_{11} = \{ \text{coffee everywhere}, \text{coffee on the table}, \text{coffee on the desk} \} \)

\( \varphi_u \)  Is there always coffee everywhere at the same time?  \( U_{10} \rightarrow ? \),
Monitoring HyperLTL

\[ \varphi_u = \forall \pi. \forall \tau. \Box (\pi \rightarrow \tau) \quad \varphi_a = \forall \pi. \exists \tau. \Box (\pi \rightarrow \tau) \]

- **Observation**: the world today at 10am
  
  \[ U_{10} = \{ \text{coffee everywhere} \} \]

- **Update**: the world at 11am
  
  \[ U_{11} = \{ \text{coffee everywhere}, \text{no coffee everywhere} \} \]

Is there always coffee everywhere at the same time? \[ U_{10} \rightarrow \text{?}, \ U_{11} \rightarrow \times \]
Monitoring HyperLTL

\[ \varphi_u = \forall \pi. \forall \tau. \square(\pi \rightarrow \tau) \quad \varphi_a = \forall \pi. \exists \tau. \square(\pi \rightarrow \tau) \]

- **Observation:** the world today at 10am
  \[ U_{10} = \{ \text{coffee at 10am} \} \]

- **Update:** the world at 11am
  \[ U_{11} = \{ \text{coffee at 11am} \} \]

\[ \varphi_u \] Is there always coffee everywhere at the same time? \[ U_{10} \rightarrow \text{?}, \ U_{11} \rightarrow \times \]

\[ \varphi_a \] Is there always coffee somewhere? \[ U_{10} \rightarrow \text{?}, \ U_{11} \rightarrow \text{?} \]
Monitoring HyperLTL

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

\[
\text{Definition}
\]

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

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

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

\[
U_1 = \{\},
\]

$U_1 \not\models s \forall \pi. \forall \tau. (\pi \rightarrow \tau)$

$U_1 \models v \forall \pi. \forall \tau. (\pi \rightarrow \tau)$
Monitoring HyperLTL

Monitoring: decide whether a given property $\varphi$ is permanently satisfied (✔), violated (✗), or neither (？), given a finite observation $U$.

**Definition**

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

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

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

**Monitoring:** decide whether a given property \( \varphi \) is **permanently** satisfied (✓), violated (✗), or neither (?)

**Definition**

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

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

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

\[
U_{11} = \{ \ldots, \ldots, \ldots \}
\]

\[
U_{11} \not\models^s \forall \pi. \forall \tau. \Diamond (\rightarrow )
\]

\[
U_{11} \models^v \forall \pi. \forall \tau. \Box (\rightarrow )
\]

\[
U_{11} \not\models^v \forall \pi. \exists \tau. \Box (\rightarrow )
\]

\[
U_{11} \not\models^s \forall \pi. \exists \tau. \Box (\rightarrow )
\]
Monitorability of HyperLTL formulas

Monitoring: decide whether a given property $\varphi$ is permanently satisfied ($\checkmark$), violated ($\times$), or neither ($?$), given a finite observation $U$.

$$U_{11} = \{ \text{observation 1}, \text{observation 2}, \text{observation 3} \}$$

$$\varphi_a = \forall \pi. \exists \tau. (\square(\pi \rightarrow \tau))$$

$U_{11} \not\models^s \varphi_a$  \hspace{1cm} $U_{11} \not\models^v \varphi_a$
Monitorability of HyperLTL formulas

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

$$U_{11} = \{\ldots, \square, \ldots\}$$

$$\varphi_a = \forall \pi. \exists \tau. (\square \pi \rightarrow \square \tau)$$

$U_{11} \not\models^s \varphi_a$ \hspace{1cm} $U_{11} \not\models^v \varphi_a$

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 $\varphi$ is permanently satisfied (√), violated (✗), or neither (؟), given a finite observation $U$.

$$U_{11} = \{\ldots, \text{state 1}, \text{state 2}, \ldots\}$$

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

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

There’s no point in monitoring $\varphi_a$!
Monitorability of HyperLTL formulas

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

\[
U_{11} = \{ \ldots, ..., \ldots \}
\]

\[\varphi_a = \forall \pi. \exists \tau. (\pi \rightarrow \tau)\]

\[U_{11} \not\models^s \varphi_a \quad U_{11} \not\models^v \varphi_a\]

**Observation:** \( U \not\models^s \varphi_a \) and \( U \not\models^v \varphi_a \) for any \( U \).

*There’s no point in monitoring \( \varphi_a \)!

**Definition (Agrawal & Bonakdarpour 2016)**

A formula \( \varphi \) is *(semantically) monitorable* if every observation \( U \) has an extension \( V \geq U \), such that \( V \models^s \varphi \) or \( V \models^v \varphi \).
HyperLTL – Summary

- Properties defined over sets of traces.  
  ⇒ 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.
  ⇒ 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!


Gray-box monitoring (of hyperproperties)
Why is $\varphi_a$ not monitorable?

**Theorem**

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

Proof. $U \not\models^v \varphi_a U \preceq \Sigma \omega$, and $\Sigma \omega \models^s \varphi_a$ because $\cdots \in \Sigma \omega$; $U \not\models^s \varphi_a$ define $T$ as $T = \{t \in \Sigma \omega | w = u \cdots \text{for } u \in U\}$; then $U \preceq T$ and $T \not\models^v \varphi_a$. 

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

• a binary (non-temporal) predicate,
• serial,
• non-reflexive.
Why is $\varphi_a$ not monitorable?

**Theorem**

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

**Proof.**

$U \not\models^v \varphi_a \quad U \preceq \Sigma^\omega$, and $\Sigma^\omega \models \varphi_a$ because $\triangleleft \cdots \in \Sigma^\omega$;
Why is $\varphi_a$ not monitorable?

**Theorem**

Let $\varphi_a = \forall \pi. \exists \tau. (\square \pi \rightarrow \diamond \tau)$.

Then $U \not\models^s \varphi_a$ and $U \not\models^v \varphi_a$ for all $U \in \mathcal{P}_{\text{fin}}(\Sigma^*)$.

**Proof.**

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

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

\[ \square \]
Why is $\varphi_a$ 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 $U \in \mathcal{P}_{\text{fin}}(\Sigma^*)$.

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

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.
Why is $\varphi_a$ 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 $U \in \mathcal{P}_{\text{fin}}(\Sigma^*)$.

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 $T = \{ t \in \Sigma^\omega | w = u \cdots \in U \}$;
then $U \preceq T$ and $T \not\models \varphi_a$.

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 $\varphi_a$ not monitorable?

**Theorem**

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

**Proof.**

$U \not\models^v \varphi_a$

$U \leq \Sigma^\omega$, and $\Sigma^\omega \models \varphi_a$ because $\ldots \in \Sigma^\omega$;

\ldots

This step is somewhat dubious.
Why is $\varphi_a$ not monitorable?

**Theorem**

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

**Proof.**

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

\ldots

This step is somewhat dubious.

- Realistic systems don’t realize every possible trace.
Why is $\varphi_a$ 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 $U \in P_{\text{fin}}(\Sigma^*)$.

**Proof.**

$U \not\models^v \varphi_a$  

$U \leq \Sigma^\omega$, and $\Sigma^\omega \models \varphi_a$ because \[\ldots\] $\in \Sigma^\omega$;

\[\ldots\]

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 $\varphi_a$ not monitorable?

**Theorem**

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

**Proof.**

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

\[ \cdots \]

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 $\varphi$ is permanently satisfied (✓), violated (✗), or neither (❔), given a finite observation $U$.

Definition

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

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

$$U \models^v \varphi \iff \text{for all } T \in \mathcal{P}(\Sigma^\omega) \text{ such that } U \preceq T, T \not\models \varphi$$
Gray-box monitoring of HyperLTL properties

**Monitoring:** decide whether a given property $\varphi$ is **permanently** satisfied (√), violated (❌), or neither (❓), given a finite observation $U$ of a system $S$.

**Definition**

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

- $U \models^s \varphi$  iff  for all $T \in \mathcal{P}(\Sigma^\omega)$ such that $U \preceq T$, $T \models \varphi$
- $U \models^v \varphi$  iff  for all $T \in \mathcal{P}(\Sigma^\omega)$ such that $U \preceq T$, $T \not\models \varphi$
Gray-box monitoring of HyperLTL properties

Monitoring: decide whether a given property $\varphi$ is permanently satisfied (√), violated (×), or neither (?), given a finite observation $U$ of a system $S$.

Definition

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

$U \models^s_S \varphi$ iff for all $T \in S$ such that $U \preceq T$, $T \models \varphi$

$U \models^v_S \varphi$ iff for all $T \in S$ such that $U \preceq T$, $T \not\models \varphi$
Gray-box monitoring of HyperLTL properties

Monitoring: decide whether a given property $\varphi$ is permanently satisfied (✅), violated (❌), or neither (❓), given a finite observation $U$ of a system $S$.

Definition

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

$U \models^s_S \varphi$ iff for all $T \in S$ such that $U \preceq T$, $T \models \varphi$

$U \models^v_S \varphi$ iff for all $T \in S$ such that $U \preceq T$, $T \not\models \varphi$

$$S = \{T \in \mathcal{P}(\Sigma^\omega) \mid |T| = 3\} \quad U = \{\text{icons of system behaviors}\}$$

$U \not\models^s \forall \pi. \exists \tau. \Box (\pi \rightarrow \tau)$

$U \models^v \forall \pi. \exists \tau. \Box (\pi \rightarrow \tau)$
Gray-box monitoring in general

Monitoring: decide whether a given property $\varphi$ is permanently satisfied (✓), violated (✗), or neither (؟), given a finite observation $O$ of a system $S$.

Definition

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

\[ O \models^s_S \varphi \quad \text{iff} \quad \text{for all } B \in S \text{ such that } O \preceq B, \quad B \models \varphi \]

\[ O \models^v_S \varphi \quad \text{iff} \quad \text{for all } B \in S \text{ such that } O \preceq B, \quad B \not\models \varphi \]
Gray-box monitoring in general

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

**Definition**

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

- $O \models^S_S \varphi$ iff for all $B \in S$ such that $O \preceq B$, $B \models \varphi$
- $O \models^v_S \varphi$ iff for all $B \in S$ such that $O \preceq B$, $B \not\models \varphi$

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


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

Monitoring: decide whether a given property $\varphi$ is permanently satisfied (✓), violated (❌), or neither (❓), given a finite observation $O$ of a system $S$. 

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

Example: $\varphi = \forall \pi. \exists \tau. (\pi \rightarrow \tau) S = \{ T \in P(\Sigma^\omega) \mid |T| = 3 \}$ 

Negate $\varphi$: $\neg \varphi = \exists \pi. \neg \exists \tau. (\pi \rightarrow \tau) \{,\} \mapsto \{\cdots,\cdots,\cdots\} \mapsto \emptyset \{,\} \mapsto \emptyset}$
Gray-box monitors for $\forall^+\exists^+$-properties

Monitoring: decide whether a given property $\varphi$ is permanently satisfied ($\checkmark$), violated ($\times$), or neither ($?$), given a finite observation $O$ of a system $S$.

A monitor for a property $\varphi$ and a system $S$ is a computable function $M_{\varphi,S} : O \rightarrow \{\checkmark, \times, ?\}$ that decides a verdict for $\varphi$ given a finite $O$ in $S$. 

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

Example: $\varphi_a = \forall \pi. \exists \tau. (\pi \rightarrow \tau)$

$S = \{T \in \mathcal{P}(\Sigma^\omega) | |T| = 3\}$

Negate $\varphi_a$: $\neg \varphi_a = \exists \pi. \neg \exists \tau. (\pi \rightarrow \tau)$

$\{\checkmark, \times, ?\} \mapsto \{\checkmark, \times, ?\}$

$\{\checkmark, \times, ?\} \mapsto \emptyset$
Gray-box monitors for $\forall^+\exists^+$-properties

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

A monitor for a property $\varphi$ and a system $S$ is a computable function $M_{\varphi,S}: O \rightarrow \{\checkmark, \times, ?\}$ that decides a verdict for $\varphi$ given a finite $O$ in $S$.

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

Example: $\varphi_{a} = \forall \pi. \exists \tau. (\pi \rightarrow \tau)$ $S = \{T \in \mathcal{P}(\Sigma^\omega) \mid \|T\| = 3\}$

Negate $\varphi_a$: $\neg \varphi_a = \exists \pi. \neg \exists \tau. (\pi \rightarrow \tau)$ $\{\cdot\} \rightarrow \{\cdot\}$ $\{\cdot\} \rightarrow \emptyset$
Gray-box monitors for $\forall^+\exists^+$-properties

Monitoring: decide whether a given property $\varphi$ is permanently satisfied ($\checkmark$), violated ($\times$), or neither ($?$), given a finite observation $O$ of a system $S$.

A monitor for a property $\varphi$ and a system $S$ is a computable function $M_{\varphi,S}: O \rightarrow \{\checkmark, \times, ?\}$ that decides a verdict for $\varphi$ given a finite $O$ in $S$.

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

Example: $\varphi_a = \forall \pi. \exists \tau. \square(\pi \rightarrow \tau)$, $S = \{T \in \mathcal{P}(\Sigma^\omega) \mid |T| = 3\}$
Gray-box monitors for $\forall^+\exists^+$-properties

Monitoring: decide whether a given property $\varphi$ is permanently satisfied ($\checkmark$), violated ($\times$), or neither ($?$), given a finite observation $O$ of a system $S$.

A monitor for a property $\varphi$ and a system $S$ is a computable function $M_{\varphi,S} : O \rightarrow \{\checkmark, \times, ?\}$ that decides a verdict for $\varphi$ given a finite $O$ in $S$.

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

Example: $\varphi_a = \forall \pi. \exists \tau. \square(\pi \rightarrow \tau)$ \hspace{1cm} $S = \{T \in \mathcal{P}(\Sigma^\omega) \mid |T| = 3\}$

Negate $\varphi_a$: $\neg \varphi_a = \exists \pi. \neg \exists \tau. \square(\pi \rightarrow \tau)$
Gray-box monitors for $\forall^+ \exists^+$-properties

Monitoring: decide whether a given property $\varphi$ is permanently satisfied ($\checkmark$), violated ($\times$), or neither ($?$), given a finite observation $O$ of a system $S$.

A monitor for a property $\varphi$ and a system $S$ is a computable function $M_{\varphi,S}: O \rightarrow \{\checkmark, \times, ?\}$ that decides a verdict for $\varphi$ given a finite $O$ in $S$.

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

Example: $\varphi_a = \forall \pi. \exists \tau. \Box(\pi \rightarrow \tau)$ \quad $S = \{T \in \mathcal{P}(\Sigma^\omega) \mid |T| = 3\}$

Negate $\varphi_a$: $\neg \varphi_a = \exists \pi. \neg \exists \tau. \Box(\pi \rightarrow \tau)$ \quad instantiate
Gray-box monitors for $\forall^+\exists^+$-properties

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

A monitor for a property $\varphi$ and a system $S$ is a computable function $M_{\varphi,S}: O \to \{\checkmark, \times, ?\}$ that decides a verdict for $\varphi$ given a finite $O$ in $S$.

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

Example: $\varphi_a = \forall \pi. \exists \tau. \Box(\pi \rightarrow \tau)$ \hspace{1cm} $S = \{T \in \mathcal{P}(\Sigma^\omega) \mid |T| = 3\}$

Negate $\varphi_a$: $\neg \varphi_a = \exists \pi. \exists \tau. \Box(\pi \rightarrow \tau)$ \hspace{1cm} instantiate \hspace{1cm} solve
Gray-box monitors for \( \forall^+ \exists^+ \)-properties

Monitoring: decide whether a given property \( \varphi \) is permanently satisfied (\( \checkmark \)), violated (\( \times \)), or neither (\( ? \)), given a finite observation \( O \) of a system \( S \).

A monitor for a property \( \varphi \) and a system \( S \) is a computable function \( M_{\varphi,S}: O \rightarrow \{\checkmark, \times, ?\} \) that decides a verdict for \( \varphi \) given a finite \( O \) in \( S \).

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

Example: \( \varphi_a = \forall \pi. \exists \tau. (\pi \rightarrow \tau) \) \( S = \{ T \in \mathcal{P}(\Sigma^\omega) \mid |T| = 3 \} \)

Negate \( \varphi_a \): \( \neg \varphi_a = \exists \pi. \neg \exists \tau. (\pi \rightarrow \tau) \) instantiate solve

\[
\begin{align*}
\{\text{Coffee, Coffee, Coffee}\} & \quad \mapsto \quad \{\text{Coffee, Coffee, Coffee, Coffee, Coffee, Coffee, Coffee, Coffee, Coffee}\} \\
\{\text{Coffee, Coffee, Coffee, Coffee, Coffee, Coffee, Coffee, Coffee, Coffee}\} & \quad \mapsto \quad \emptyset
\end{align*}
\]
Gray-box monitoring – Summary

- Properties defined over observations (e.g. traces or sets of traces).
  - 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 $S$ imposes enough constraints.
- Monitorability of formulas depends on set of valid system behaviors $S$.
  - For example, $\forall^+ \exists^+$-properties are monitorable for some choices of $S$.
  - We will see a more interesting example later…
Undecidable hyperproperties

- Trace/hyper
- Black/gray
- Computability
Monitorability is not existence of monitors

A formula $\varphi$ is **semantically gray-box monitorable** for a system $S$ if every observation $O$ has an extension $P \succeq O$ in $S$, such that $P \models_S \varphi$ or $P \models^{\neg}_S \varphi$. 
Monitorability is not existence of monitors

A formula \( \varphi \) is \textbf{semantically gray-box monitorable} for a system \( S \) if every observation \( O \) has an extension \( P \geq O \) in \( S \), such that \( P \models^s \varphi \) or \( P \models^v \varphi \).

A \textbf{monitor} for a property \( \varphi \) and a system \( S \) is a \textbf{computable} function \( M_{\varphi,S} : O\{\text{✓, ✗, ?}\} \) that decides a \textbf{verdict} for \( \varphi \) given a finite \( u \).
Monitorability is not existence of monitors

A formula \( \varphi \) is semantically gray-box monitorable for a system \( S \) if every observation \( O \) has an extension \( P \succeq O \) in \( S \), such that \( P \models^S \varphi \) or \( P \models^? \varphi \).

A monitor for a property \( \varphi \) and a system \( S \) is a computable function \( M_{\varphi,S} : O \{\checkmark, \times, ?\} \) that decides a verdict for \( \varphi \) given a finite \( u \).

Observation: Monitorability of \( \varphi \) in \( S \) does not guarantee the existence of a sound and complete monitor \( M_{\varphi,S} \).
Monitorability is not existence of monitors

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

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

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

**Example:** Let $T$ be some Turing machine.

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

Because $T$ is deterministic, either $u \models_S^s \varphi$ or $u \models_S^v \varphi$, for any $u$ in $S$.
Monitorability is not existence of monitors

A formula $\varphi$ is semantically gray-box monitorable for a system $S$ if every observation $O$ has an extension $P \succeq O$ in $S$, such that $P \models_S \varphi$ or $P \models_S \neg \varphi$.

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

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

Example: Let $T$ be some Turing machine.

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

Because $T$ is deterministic, either $u \models_S \varphi$ or $u \models_S \neg \varphi$, for any $u$ in $S$.

$\Rightarrow$ $\varphi$ is monitorable in $S$.
Monitorability is not existence of monitors

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

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

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

Example: Let $T$ be some Turing machine.

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

Because $T$ is deterministic, either $u \models^s_S \varphi$ or $u \models^v_S \varphi$, for any $u$ in $S$.

$\Rightarrow$ $\varphi$ is monitorable in $S$;

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

Monitorability is not existence of monitors

A formula $\varphi$ is semantically gray-box monitorable for a system $S$ if every observation $O$ has an extension $P \geq O$ in $S$, such that $P \models_S \varphi$ or $P \models_S \neg \varphi$.

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

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

Example: Let $T$ be some Turing machine.

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

Because $T$ is deterministic, either $u \models_S \varphi$ or $u \models_S \neg \varphi$, for any $u$ in $S$.

$\Rightarrow \varphi$ is monitorable in $S$;

$\Rightarrow$ there is a sound monitor $M_{\varphi,S}$ that only answers $\checkmark$ or $?!
Case study: distributed data minimality
Non-monitorable examples

- Storage limitation (Article 5): Personal data shall be [...] adequate relevant, and limited to what is necessary in relation to the purposes for which they are processed (data minimization) [...] 

- Data minimization (attempt at formalization)
  
  collect (data, dataid, dsid) IMPLIES EVENTUALLY use(data, dataid, dsid)

- But MFOTL semantics requires collected data used in EVERY run of the system.
  
  - Not finitely falsifiable (liveness) and interpretation is also too strong.
  
  - Example: when booking a long-haul flight, customers provide emergency contact for an account. In majority of cases, data is collected, not used, and deleted.

- Better would be a CTL formulation (although not monitorable on a trace)
  
  collect (data, dataids, dsid) IMPLIES EXISTS EVENTUALLY use(data, dataid, dsid)
Case study: distributed data minimality

- Distributed data minimality (DDM)
  - privacy property (GDPR)
  - generalization of data minimality to a multi-input setting
  - $\forall\exists\exists$-hyperproperty
    
    $$
    \varphi_i = \forall\pi.\forall\pi'.\exists\tau.\exists\tau'. \neg \text{same}_i(\pi, \pi') \rightarrow 
    \left( \begin{array}{c}
    \text{same}_i(\pi, \tau) \land \text{same}_i(\pi', \tau') \land \\
    \text{almost}_i(\tau, \tau') \land \neg \text{output}(\tau, \tau')
    \end{array} \right)
    $$

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

Yet, we have a monitor... here's how...
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
    \[
    \varphi_i = \forall \pi. \forall \pi'. \exists \tau. \exists \tau'. \neg \text{same}_i(\pi, \pi') \rightarrow \left( \text{same}_i(\pi, \tau) \land \text{same}_i(\pi', \tau') \land \text{almost}_i(\tau, \tau') \land \neg \text{output}(\tau, \tau') \right)
    \]

- 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
    \[
    \varphi_i = \forall \pi. \exists \pi'. \exists \tau. \exists \tau'. \neg \text{same}_i(\pi, \pi') \rightarrow \left( \text{same}_i(\pi, \tau) \wedge \text{same}_i(\pi', \tau') \wedge \text{almost}_i(\tau, \tau') \wedge \neg \text{output}(\tau, \tau') \right)
    \]

- 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
    \[
    \varphi_i = \forall\pi.\forall\pi'.\exists\tau.\exists\tau'. \neg \text{same}_i(\pi, \pi') \rightarrow \left( \text{same}_i(\pi, \tau) \land \text{same}_i(\pi', \tau') \land \text{almost}_i(\tau, \tau') \land \neg \text{output}(\tau, \tau') \right)
    \]

- 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

\[
\varphi_i = \forall \pi. \forall \pi'. \exists \tau. \exists \tau'. \neg \text{same}_i(\pi, \pi') \rightarrow \left( \begin{array}{c}
\text{same}_i(\pi, \tau) \land \text{same}_i(\pi', \tau') \land \\
\text{almost}_i(\tau, \tau') \land \neg \text{output}(\tau, \tau')
\end{array} \right)
\]

\[
\varphi_{\text{dm}} = \bigwedge_{i=1}^{n} \varphi_i,
\Sigma_f^\# = \{ (x, y) \mid f(x) = y \},
S_f = \mathcal{P}(\Sigma_f^\#)
\]
Distributed data minimality

\[ \varphi_i = \forall \pi . \forall \pi'. \exists \tau . \exists \tau'. \neg \text{same}_i(\pi, \pi') \rightarrow \left( \begin{array}{c} \text{same}_i(\pi, \tau) \land \text{same}_i(\pi', \tau') \land \end{array} \right) \]

\[ \text{almost}_i(\tau, \tau') \land \neg \text{output}(\tau, \tau') \]

\[ \varphi_{dm} = \bigwedge_{i=1}^{n} \varphi_i, \quad \Sigma_f^\# = \{(x, y) \mid f(x) = y\}, \quad S_f = \mathcal{P}(\Sigma_f^\#) \]

Using the generalized framework

- Set of observable behaviors \( \mathcal{O} = \Sigma_f^\# \) are valid function applications.
Distributed data minimality

\[ \varphi_i = \forall \pi. \forall \pi'. \exists \tau. \exists \tau'. \neg \text{same}_i(\pi, \pi') \rightarrow \left( \text{same}_i(\pi, \tau) \land \text{same}_i(\pi', \tau') \land \right. \]
\[ \left. \text{almost}_i(\tau, \tau') \land \neg \text{output}(\tau, \tau') \right) \]

\[ \varphi_{dm} = \bigwedge_{i=1}^{n} \varphi_i, \quad \Sigma_f^\# = \{(x, y) \mid f(x) = y\}, \quad S_f = \mathcal{P}(\Sigma_f^\#) \]

Using the generalized framework

- Set of observable behaviors \( \mathcal{O} = \Sigma_f^\# \) are valid function applications.
- Not black-box monitorable.
Distributed data minimality

\[ \varphi_i = \forall \pi. \forall \pi'. \exists \tau. \exists \tau'. \neg \text{same}_i(\pi, \pi') \rightarrow \left( \begin{array}{c} \text{same}_i(\pi, \tau) \land \text{same}_i(\pi', \tau') \land \\ \text{almost}_i(\tau, \tau') \land \neg \text{output}(\tau, \tau') \end{array} \right) \]

\[ \varphi_{dm} = \bigwedge_{i=1}^{n} \varphi_i, \quad \Sigma_f^\# = \{(x, y) \mid f(x) = y\}, \quad S_f = \mathcal{P}(\Sigma_f^\#) \]

Using the generalized framework

- Set of observable behaviors \( O = \Sigma_f^\# \) are valid function applications.
- Not black-box monitorable, but gray-box monitorable (thanks to \( S \)).
A sound monitor for distributed data minimality

\[ \varphi_i = \forall \pi. \forall \pi'. \exists \tau. \exists \tau'. \neg \text{same}_i(\pi, \pi') \rightarrow \begin{cases} \text{same}_i(\pi, \tau) \land \text{same}_i(\pi', \tau') \land & \\
\text{almost}_i(\tau, \tau') \land \neg \text{output}(\tau, \tau') \end{cases} \]

\[ \varphi_{dm} = \bigwedge_{i=1}^{n} \varphi_i, \quad \Sigma^\#_f = \{(x, y) \mid f(x) = y\}, \quad S_f = \mathcal{P}(\Sigma^\#_f) \]
A sound monitor for distributed data minimality

$$\varphi_i = \forall \pi. \forall \pi'. \exists \tau. \exists \tau'. \neg \text{same}_i(\pi, \pi') \rightarrow \left( \text{same}_i(\pi, \tau) \land \text{same}_i(\pi', \tau') \land \right. $$

$$\left( \text{almost}_i(\tau, \tau') \land \neg \text{output}(\tau, \tau') \right)$$

$$\varphi_{dm} = \bigwedge_{i=1}^{n} \varphi_i,$$

$$\Sigma_f^\# = \{(x, y) \mid f(x) = y\}, \quad S_f = \mathcal{P}(\Sigma_f^\#)$$

We build a monitor

$$M_{dm}(U) = \begin{cases} ? & \text{if } f(u_{in}) \neq u_{out} \text{ for some } u \in U, \\
? & \text{if } \bigwedge_{i=1}^{n} \bigwedge_{u, u' \in U} N_{f,i}(\text{proj}_i(u_{in}), \text{proj}_i(u_{in}')) \neq \mathbf{x}, \\
\mathbf{x} & \text{otherwise}. \end{cases}$$
A sound monitor for distributed data minimality

$$\varphi_i = \forall \pi. \forall \pi'. \exists \tau. \exists \tau'. \neg \text{same}_i(\pi, \pi') \rightarrow \left( \text{same}_i(\pi, \tau) \land \text{same}_i(\pi', \tau') \land \text{almost}_i(\tau, \tau') \land \neg \text{output}(\tau, \tau') \right)$$

$$\varphi_{dm} = \bigwedge_{i=1}^{n} \varphi_i,$$

$$\Sigma_f^# = \{(x, y) \mid f(x) = y\}, \quad S_f = \mathcal{P}(\Sigma_f^#)$$

We build a monitor

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

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

$$N_{f,i}(x, y) = \begin{cases} \text{✓ or ?} & \text{if } \exists z \in I. f(z[i \mapsto x]) \neq f(z[i \mapsto y]), \\ \text{x or ?} & \text{otherwise.} \end{cases}$$
A sound monitor for distributed data minimality

We build a monitor

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

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

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

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

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.  

Andreas Bauer, Martin Leucker, and Christian Schallhart.  
Runtime verification for LTL and TLTL.  

Andreas Bauer, Martin Leucker, and Christian Schallhart.  
The good, the bad, and the ugly—but how ugly is ugly?  

Borzoo Bonakdarpour, César Sánchez, and Gerardo Schneider.  
Monitoring hyperproperties by combining static analysis and runtime verification.
What can you verify and enforce at runtime?

Monitoring hyperproperties.

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.

Amir Pnueli and Aleksandr Zaks.
PSL model checking and run-time verification via testers.

Xian Zhang, Martin Leucker, and Wei Dong.
Runtime verification with predictive semantics.
Except where otherwise noted, this work is licensed under

http://creativecommons.org/licenses/by/3.0/