Epistemic Logic for verifying runtime verification communication protocols

Antonis Achilleos, **Elli Anastasiadi**, Adrian Francalanza, & Jasmine Xuereb

ICE-TCS, School of Computer Science, Reykjavik University

13th Panhellenic Logic Symposium
July 6-10, Volos, Greece

In this work

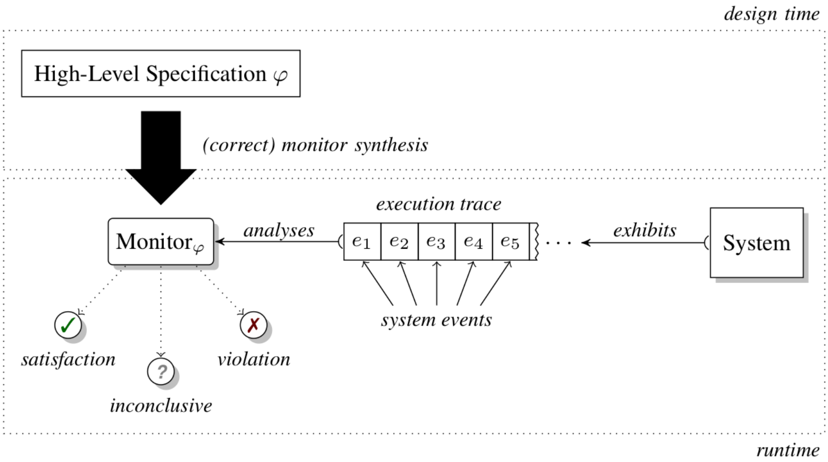
Adrian Francalanza

Jasmine Xuereb

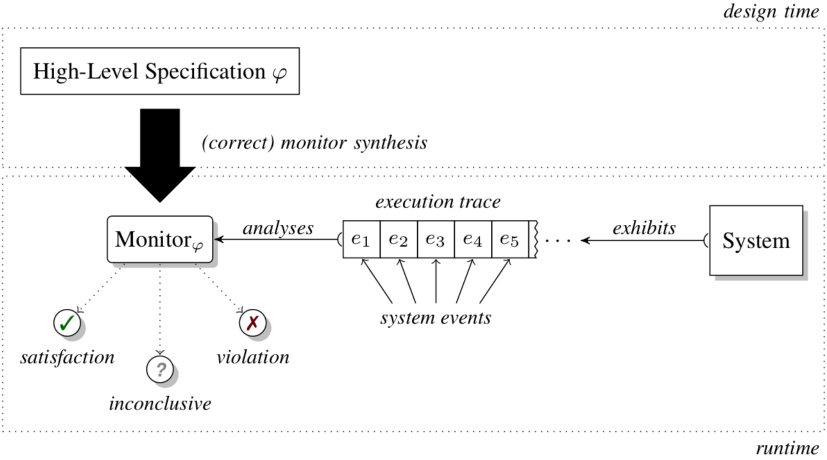
Antonis Achilleos

Elli Anastasiadi

What is this about?



What is this about?



How about monitoring a global property over many systems/components/agents?

Aspects of the problem

- Stating global properties - what does “now” mean?.
- Tackling asynchronous components (trace collection, local clocks).
- Processing **many** traces at runtime.

In this work we try it all!*

Aspects of the problem

- Stating global properties - what does “now” mean?.
- Tackling asynchronous components (trace collection, local clocks).
- Processing **many** traces at runtime.

In this work we try it all!*

*****In a very restricted way.

Aspects of the problem

- Stating global properties - what does “now” mean?.
- Tackling asynchronous components (trace collection, local clocks).
- Processing **many** traces at runtime.

In this work we try it all!*

*****In a very restricted way. For now.

Aspects of the problem

- Stating global properties - what does "now" mean?.
- Tackling asynchronous components (trace collection, local clocks).
- Processing **many** traces at runtime.

In this work we try it all!*

*In a very restricted way. For now.

Logic | Parallel Setup | Monitoring Setup

Aspects of the problem

- Stating global properties - what does “now” mean?.
- Tackling asynchronous components (trace collection, local clocks).
- Processing **many** traces at runtime.

In this work we try it all!*

*In a very restricted way. For now.

Logic | Parallel Setup | **Monitoring Setup**

The Logic: $Hyper^1$-sHML

$$
\begin{array}{llllll}
\varphi ::= \exists_\pi \varphi & \mid \forall_\pi \varphi & \mid \varphi \wedge \varphi & \mid \varphi \vee \varphi & \mid \psi \\
\psi ::= \mathtt{tt} & \mid \mathtt{ff} & \mid [A]\psi & \mid \langle A \rangle \psi & \\
\quad\quad \mid \psi \wedge \psi & \mid \psi \vee \psi & \mid \max x.\psi & \mid \min x.\psi & \mid x,
\end{array}
$$

where $\pi \in \Pi$, $x \in V$, and $A \subseteq \textsc{Act}_{\Pi}$ is consistent.

The semantics is given over a finite set of infinite traces T, which is interpreting the quantification.

The Logic: $Hyper^1$-sHML

$$
\begin{array}{llllll}
\varphi ::= \exists_\pi \varphi & \mid \forall_\pi \varphi & \mid \varphi \wedge \varphi & \mid \varphi \vee \varphi & \mid \psi \\
\psi ::= \mathtt{tt} & \mid \mathtt{ff} & \mid [A]\psi & \mid \langle A \rangle \psi & \\
\quad \mid \psi \wedge \psi & \mid \psi \vee \psi & \mid \max x.\psi & \mid \min x.\psi & \mid x,
\end{array}
$$

where $\pi \in \Pi$, $x \in V$, and $A \subseteq \textsc{Act}_{\Pi}$ is consistent.

The semantics is given over a finite set of infinite traces T, which is interpreting the quantification.

Example: The formula

$$\forall_\pi [a]\mathtt{ff} \sqcap \exists_\pi [b](max\ x.([a]\mathtt{ff}\ \wedge\ [b]x))\ ,$$

over $\textsc{Act} = \{a, b\}$, means that none of the traces start with a, and $b^\omega \in T$.

The Monitors

Sets of traces have different cardinalities. $\Rightarrow$ The monitor for a property should treat the cardinality as a parameter!

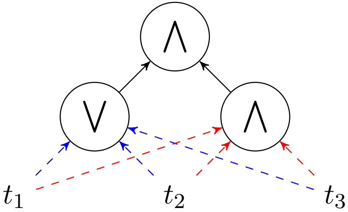
The Monitors

Sets of traces have different cardinalities. $\Rightarrow$ The monitor for a property should treat the cardinality as a parameter!
Our idea: A monitor M is a family of **circuits**.

The Monitors

Sets of traces have different cardinalities. $\Rightarrow$ The monitor for a property should treat the cardinality as a parameter!

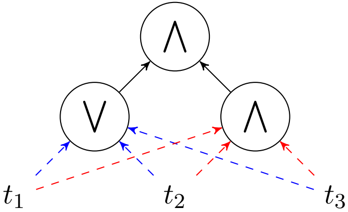
Our idea: A monitor M is a family of **circuits**.



The Monitors

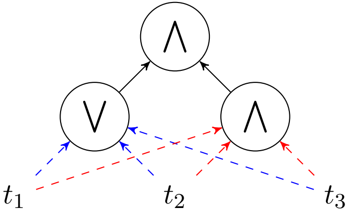
Sets of traces have different cardinalities. $\Rightarrow$ The monitor for a property should treat the cardinality as a parameter!

Our idea: A monitor M is a family of **circuits**.



$\leftarrow$ this is not an easy object to work with.

Sets of traces have different cardinalities. $\Rightarrow$ The monitor for a property should treat the cardinality as a parameter!
Our idea: A monitor M is a family of **circuits**.

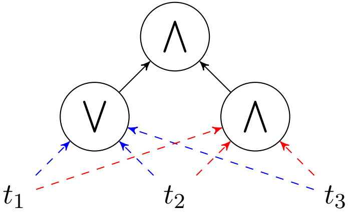


$\leftarrow$ this is not an easy object to work with.

Instead: Syntax!

The Monitors

Sets of traces have different cardinalities. $\Rightarrow$ The monitor for a property should treat the cardinality as a parameter!
Our idea: A monitor M is a family of **circuits**.



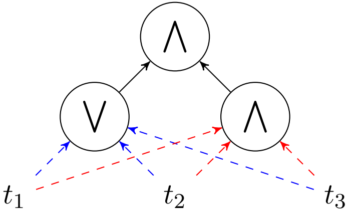
$\leftarrow$ this is not an easy object to work with.

Instead: Syntax!

$$M \in \textsc{Cmon} ::= \bigvee [m]_k \quad | \quad \bigwedge [m]_k \quad | \quad M \vee M \quad | \quad M \wedge M$$

$$m ::= yes \quad | \quad no \quad | \quad end \quad | \quad a.m \quad | \quad m + n \quad | \quad rec\ x.m \quad | \quad x$$

Sets of traces have different cardinalities. $\Rightarrow$ The monitor for a property should treat the cardinality as a parameter!
Our idea: A monitor M is a family of **circuits**.



$\leftarrow$ this is not an easy object to work with.

Instead: Syntax!

$$M \in C\textsc{mon} ::= \bigvee[m]_k \quad | \quad \bigwedge[m]_k \quad | \quad M \vee M \quad | \quad M \wedge M$$

$$m ::= yes \;|\; no \;|\; end \;|\; a.m \;|\; m + n \;|\; rec\ x.m \;|\; x$$

How do we go from syntax to circuits?

Monitor semantics:

$$\frac{s[m_{[i]}] = yes}{s \to s[{}^{yes}/\bigvee[m]_k]} \qquad \frac{s[m_{[i]}] = no}{s \to s[\bigvee[m]_k \backslash i]} \qquad \frac{s[m_{[i]}] = end}{s \to s[\bigvee[m]_k \backslash end_i]}$$

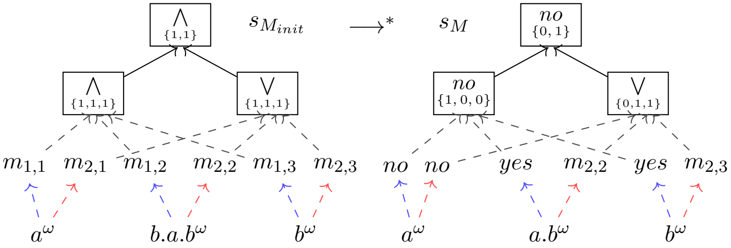
$$\frac{s[\bigvee[m]_k] = 0}{s \to s[{}^{no}/\bigvee[m]_k]} \qquad \frac{s[\bigvee[m]_k] = 2^k}{s \to s[{}^{end}/\bigvee[m]_k]}$$

Instrumentation:

$$\frac{m \xrightarrow{\tau} m'}{m \triangleleft t \xrightarrow{\tau} m' \triangleleft t} \qquad \frac{m \xrightarrow{a} m'}{m \triangleleft a.t \xrightarrow{a} m' \triangleleft t} \qquad \frac{\forall j \in \{1, \dots r\},\ m_{j[i]} \triangleleft t \xrightarrow{a} m'_{j[i]} \triangleleft t'}{s \triangleleft (\overrightarrow{m} \triangleleft T) \to s \triangleleft (\overrightarrow{m}[{}^{m'_{j[i]}}/{}_{m_{j[i]}},\ \forall j] \triangleleft T[{}^{t'}/{}_{t}])}$$

$$\frac{s \to s'}{s \triangleleft (\overrightarrow{m} \triangleleft T) \to s' \triangleleft (\overrightarrow{m} \triangleleft T)} \qquad \frac{s \triangleleft (\overrightarrow{m} \triangleleft T) \to s \triangleleft (\overrightarrow{m}[{}^{v}/{}_{n_{j[i]}}] \triangleleft T[{}^{t'}/{}_{t}])}{s \triangleleft (\overrightarrow{m} \triangleleft T) \to s[{}^{v}/{}_{g_{m_{j[i]}}}] \triangleleft (\overrightarrow{m}[{}^{v}/{}_{n_{j[i]}}] \triangleleft T[{}^{t'}/{}_{t}])}$$

Formula: $\forall_\pi [a]\texttt{ff} \sqcap \exists_\pi [b](max\ x.([a]\texttt{ff}\ \wedge\ [b]x))$.
Monitor:



Soundness and Completeness

...are very important to have.

...are very important to have.

Soundness: If the monitor for formula ϕ produces the verdict *yes* (symmetrically for *no*) after reading the finite trace T_0 then, $T_0 \cdot \Sigma^\omega \subseteq [\![\phi]\!]$.

Soundness and Completeness

...are very important to have.

Soundness: If the monitor for formula ϕ produces the verdict *yes* (symmetrically for *no*) after reading the finite trace T_0 then, $T_0 \cdot \Sigma^\omega \subseteq [\![\phi]\!]$.

- I.e. The monitors do not report wrong things.

Soundness and Completeness

...are very important to have.

Soundness: If the monitor for formula ϕ produces the verdict *yes* (symmetrically for *no*) after reading the finite trace T_0 then, $T_0 \cdot \Sigma^\omega \subseteq [\![\phi]\!]$.

- I.e. The monitors do not report wrong things.

Completeness: Many kinds.
Violation Completeness:
A monitor M is violation complete for φ if, for each finite trace s, $s \cdot \Sigma^\omega \subseteq \Sigma^\omega \setminus [\![\varphi]\!]$ implies that M produces *no* upon reading s.

Soundness and Completeness

...are very important to have.

Soundness: If the monitor for formula ϕ produces the verdict *yes* (symmetrically for *no*) after reading the finite trace T_0 then, $T_0 \cdot \Sigma^\omega \subseteq [\![\phi]\!]$.

- I.e. The monitors do not report wrong things.

Completeness: Many kinds.
Violation Completeness:
A monitor M is violation complete for φ if, for each finite trace s, $s \cdot \Sigma^\omega \subseteq \Sigma^\omega \setminus [\![\varphi]\!]$ implies that M produces *no* upon reading s.

- I.e. the monitors report **all** problems.

Soundness and Completeness

...are very important to have.

Soundness: If the monitor for formula ϕ produces the verdict *yes* (symmetrically for *no*) after reading the finite trace T_0 then, $T_0 \cdot \Sigma^\omega \subseteq [\![\phi]\!]$.

- I.e. The monitors do not report wrong things.

Completeness: Many kinds.
Violation Completeness:
A monitor M is violation complete for φ if, for each finite trace s, $s \cdot \Sigma^\omega \subseteq \Sigma^\omega \setminus [\![\varphi]\!]$ implies that M produces *no* upon reading s.

- I.e. the monitors report **all** problems.

And we proved we **can** have these.

Intermission

Intermission

End of past work part, time for more logic.

Intermission

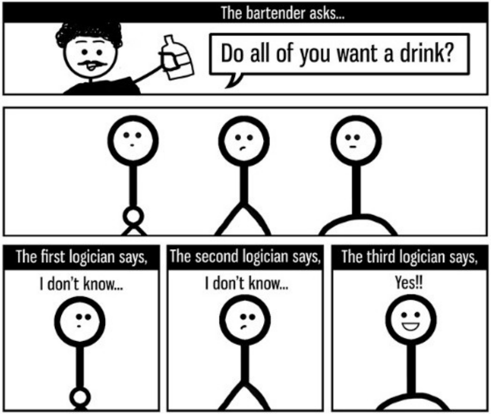
End of past work part, time for more logic.

Why?

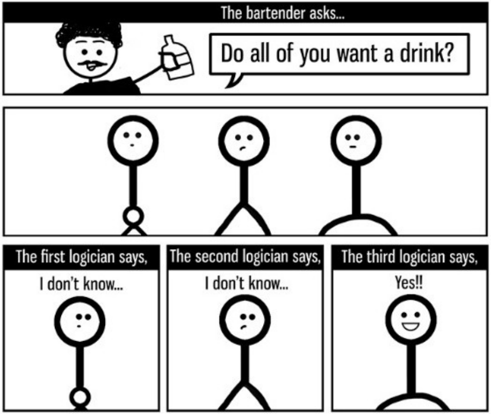
Intermission
End of past work part, time for more logic.

Why? ...

Three Logicians walk into a bar

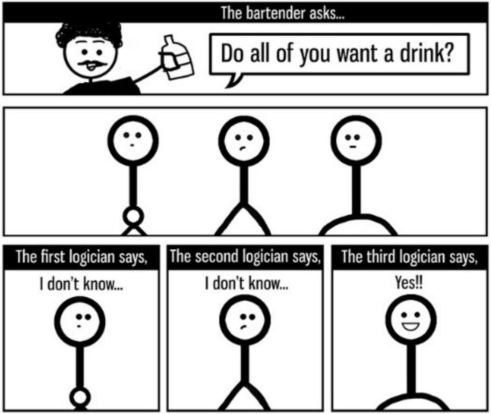


Three Logicians walk into a bar



Uses Dynamic Epidemic Logic ↗.

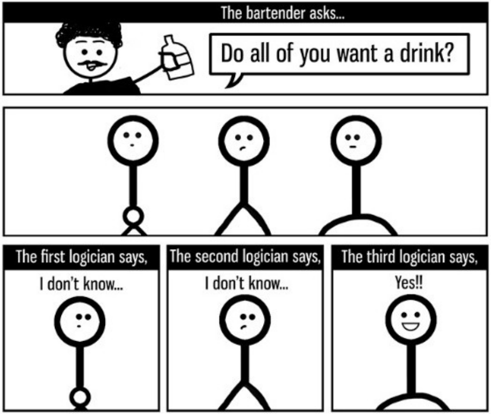
Three Logicians walk into a bar



Uses Dynamic Epidemic Logic $\nearrow$.

We want:

- Use this (correct) proofs to extract good monitoring procedures.



Uses Dynamic Epidemic Logic ↗.

We want:

- Use this (correct) proofs to extract good monitoring procedures.
- Hopefully extract them automatically (proofs and protocols).

Definition (Multi-agent epistemic logic)

For a set of agents $\mathcal{A}$, a formula ϕ in the multi-agent modal logic is defined as:

$$\phi ::= \top \quad | \; \bot \quad | \; p \quad | \; \neg\phi \quad | \; \phi \wedge \phi \quad | \; K_i\phi \quad | \; C_G\phi$$

where p is an atomic formula, $i \in \mathcal{A}$, and $G \subseteq \mathcal{A}$. Implication and disjunction can be defined from the other operators as usual.

Lets get our hands dirty!

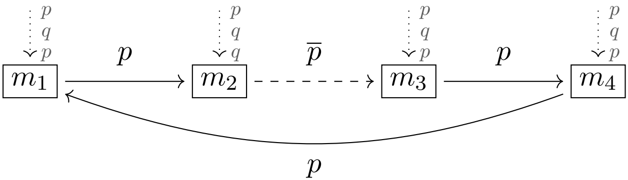
Definition (Multi-agent epistemic logic)

For a set of agents $\mathcal{A}$, a formula ϕ in the multi-agent modal logic is defined as:

$$\phi ::= \top \quad | \ \bot \quad | \ p \quad | \ \neg\phi \quad | \ \phi \wedge \phi \quad | \ K_i\phi \quad | \ C_G\phi$$

where p is an atomic formula, $i \in \mathcal{A}$, and $G \subseteq \mathcal{A}$. Implication and disjunction can be defined from the other operators as usual.

- This logic can help us express the ***state*** of knowledge between agents in some distributed setup.
- Evolution of knowledge of agents (monitors) coming soon.



A Communication Protocol for

$$\varphi = \forall\pi\forall\pi' max\ x([p_\pi, \overline{p}_{\pi'}]\mathtt{ff} \wedge [p_\pi, p_{\pi'}]x \wedge [\overline{p}_\pi, \overline{p}_{\pi'}]x)\ ,$$

i.e. "all traces agree on p."

An example correctness proof - Natural Deduction!

1. A monitor is an agent r.

An example correctness proof - Natural Deduction!

1. A monitor is an agent r.
2. Model the protocol as epistemic logic statements that are common knowledge.

An example correctness proof - Natural Deduction!

1. A monitor is an agent r.
2. Model the protocol as epistemic logic statements that are common knowledge.
   Example: $\forall r : C_{\mathcal{A}}(K_r p_r \rightarrow K_{r+1} p_r)$.

An example correctness proof - Natural Deduction!

1. A monitor is an agent r.
2. Model the protocol as epistemic logic statements that are common knowledge.
   Example: $\forall r : C_{\mathcal{A}}(K_r p_r \to K_{r+1} p_r)$.
3. Use those as premises.

An example correctness proof - Natural Deduction!

1. A monitor is an agent r.
2. Model the protocol as epistemic logic statements that are common knowledge.
   Example: $\forall r : C_{\mathcal{A}}(K_r p_r \rightarrow K_{r+1} p_r)$.
3. Use those as premises.
4. Use (known) natural deduction systems for epistemic logic to prove theorems.

An example correctness proof - Natural Deduction!

1. A monitor is an agent r.
2. Model the protocol as epistemic logic statements that are common knowledge.
   Example: $\forall r : C_{\mathcal{A}}(K_r p_r \rightarrow K_{r+1} p_r)$.
3. Use those as premises.
4. Use (known) natural deduction systems for epistemic logic to prove theorems.
5. Success.*

An example correctness proof - Natural Deduction!

1. A monitor is an agent r.
2. Model the protocol as epistemic logic statements that are common knowledge.
   Example: $\forall r : C_{\mathcal{A}}(K_r p_r \rightarrow K_{r+1} p_r)$.
3. Use those as premises.
4. Use (known) natural deduction systems for epistemic logic to prove theorems.
5. Success.*

* We managed to prove that the described protocol can detect all violations of the stated property.

An example correctness proof - Natural Deduction!

1. A monitor is an agent r.
2. Model the protocol as epistemic logic statements that are common knowledge.
   Example: $\forall r : C_{\mathcal{A}}(K_r p_r \rightarrow K_{r+1} p_r)$.
3. Use those as premises.
4. Use (known) natural deduction systems for epistemic logic to prove theorems.
5. Success.*

* We managed to prove that the described protocol can detect all violations of the stated property.

Now what?

An example correctness proof - Natural Deduction!

1. A monitor is an agent r.
2. Model the protocol as epistemic logic statements that are common knowledge.
   Example: $\forall r : C_{\mathcal{A}}(K_r p_r \to K_{r+1} p_r)$.
3. Use those as premises.
4. Use (known) natural deduction systems for epistemic logic to prove theorems.
5. Success.*

* We managed to prove that the described protocol can detect all violations of the stated property.

Now what? Automate proofs (tableaus), use logic that expresses the *acquisition of knowledge*, even more success.

Thank you for your attention!

Questions?