MONITORING DECENTRALIZED SPECIFICATIONS

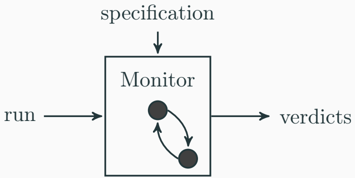
Antoine El-Hokayem   Yliès Falcone

Univ. Grenoble Alpes, Inria, CNRS
Grenoble, France

(Decentralized) Monitoring

MONITORING (AKA RUNTIME VERIFICATION) ↪ OVERVIEW

- Lightweight verification technique
- Checks whether a run of a program conforms to a specification
  (As opposed to model checking which verifies all runs)
- Monitors are synthesized and integrated to observe the system
- Monitors determine a verdict: $\mathbb{B}_3 = \{\top, \bot, ?\}$
  - $\top$ (true): run complies with specification
  - $\bot$ (false): run does not comply with specification
  - ?: verdict cannot be determined (yet)



MONITORING $\hookrightarrow$ SYSTEM ABSTRACTION

1. Components ($\mathcal{C}$)

Example

1. $\{c_0, c_1\}$ (Temp sensor + Fan)

MONITORING $\hookrightarrow$ SYSTEM ABSTRACTION

1. Components ($\mathcal{C}$)
2. Atomic propositions (AP)

Example

1. $\{c_0, c_1\}$ (Temp sensor + Fan)
2. $\{\mathrm{t_{low}}, \mathrm{t_{med}}, \mathrm{t_{high}}, \mathrm{t_{crit}}, \mathrm{fan}\}$ (e.g., $\mathrm{t_{crit}}$ "temperature critical")

MONITORING ↪ SYSTEM ABSTRACTION

1. Components ($\mathcal{C}$)
2. Atomic propositions (AP)
3. Observations/Events ($AP \to \mathbb{B}_2$, possibly partial)

Example

1. $\{c_0, c_1\}$ (Temp sensor + Fan)
2. $\{\mathrm{t_{low}}, \mathrm{t_{med}}, \mathrm{t_{high}}, \mathrm{t_{crit}}, \mathrm{fan}\}$ (e.g., $\mathrm{t_{crit}}$ "temperature critical")
3. $\{\langle \mathrm{t_{low}}, \top \rangle, \langle \mathrm{fan}, \bot \rangle\}$ — "temperature is low and fan is not on"

MONITORING $\hookrightarrow$ SYSTEM ABSTRACTION

1. Components ($\mathcal{C}$)
2. Atomic propositions (AP)
3. Observations/Events ($AP \to \mathbb{B}_2$, possibly partial)
4. Trace: a sequence of events for each component (partial function)

Example

1. $\{c_0, c_1\}$ (Temp sensor + Fan)
2. $\{\mathrm{t_{low}}, \mathrm{t_{med}}, \mathrm{t_{high}}, \mathrm{t_{crit}}, \mathrm{fan}\}$ (e.g., $\mathrm{t_{crit}}$ "temperature critical")
3. $\{\langle \mathrm{t_{low}}, \top \rangle, \langle \mathrm{fan}, \bot \rangle\}$ — "temperature is low and fan is not on"
4. $\begin{bmatrix} 0 \mapsto c_0 & \mapsto \{\langle \mathrm{t_{low}}, \top\rangle, \langle \mathrm{t_{med}}, \bot\rangle, \ldots\} & 0 \mapsto c_1 & \mapsto \{\langle \mathrm{fan}, \bot\rangle\} \\ 1 \mapsto c_0 & \mapsto \{\langle \mathrm{t_{med}}, \top\rangle, \ldots\} & 1 \mapsto c_1 & \mapsto \{\langle \mathrm{fan}, \bot\rangle\} \\ 2 \mapsto c_0 & \mapsto \{\langle \mathrm{t_{high}}, \top\rangle, \ldots\} & 2 \mapsto c_1 & \mapsto \{\langle \mathrm{fan}, \top\rangle\} \end{bmatrix}$

MONITORING ↪ SYSTEM ABSTRACTION

1. Components ($\mathcal{C}$)
2. Atomic propositions (AP)
3. Observations/Events ($AP \rightarrow \mathbb{B}_2$, possibly partial)
4. Trace: a sequence of events for each component (partial function)

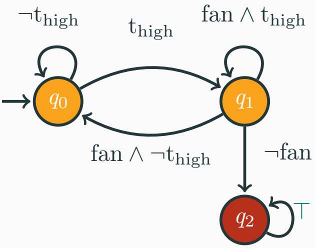
Example

1. $\{c_0, c_1\}$ (Temp sensor + Fan)
2. $\{\mathrm{t_{low}}, \mathrm{t_{med}}, \mathrm{t_{high}}, \mathrm{t_{crit}}, \mathrm{fan}\}$ (e.g., $\mathrm{t_{crit}}$ "temperature critical")
3. $\{\langle \mathrm{t_{low}}, \top \rangle, \langle \mathrm{fan}, \bot \rangle\}$ — "temperature is low and fan is not on"
4. $$\begin{bmatrix} 0 \mapsto c_0 & \mapsto \{\langle \mathrm{t_{low}}, \top \rangle, \langle \mathrm{t_{med}}, \bot \rangle, \ldots\} & 0 \mapsto c_1 & \mapsto \{\langle \mathrm{fan}, \bot \rangle\} \\ 1 \mapsto c_0 & \mapsto \{\langle \mathrm{t_{med}}, \top \rangle, \ldots\} & 1 \mapsto c_1 & \mapsto \{\langle \mathrm{fan}, \bot \rangle\} \\ 2 \mapsto c_0 & \mapsto \{\langle \mathrm{t_{high}}, \top \rangle, \ldots\} & 2 \mapsto c_1 & \mapsto \{\langle \mathrm{fan}, \top \rangle\} \end{bmatrix}$$

$$\{\langle \mathrm{t_{low}}, \top \rangle, \langle \mathrm{fan}, \bot \rangle, \ldots\} \cdot \{\langle \mathrm{t_{med}}, \top \rangle, \langle \mathrm{fan}, \bot \rangle, \ldots\} \cdot \{\langle \mathrm{t_{high}}, \top \rangle, \langle \mathrm{fan}, \top \rangle, \ldots\}$$

MONITORING USING AUTOMATA ↪ EXAMPLE

"Fan must always be turned on when temperature is high"

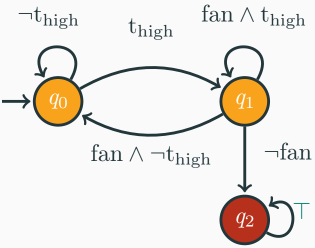


$$\boldsymbol{G}(t_{high} \implies \boldsymbol{X} fan)$$

1. At $t = 1$, from q_0:

MONITORING USING AUTOMATA ↪ EXAMPLE

"Fan must always be turned on when temperature is high"



$$\boldsymbol{G}(t_{high} \implies \boldsymbol{X} fan)$$

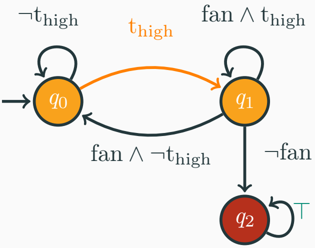
1. At $t = 1$, from q_0:

   1.1 Observe

t_{high}	$\top$
fan	$\bot$

MONITORING USING AUTOMATA ↪ EXAMPLE

"Fan must always be turned on when temperature is high"



$$G(t_{high} \implies X\text{fan})$$

1. At $t = 1$, from q_0:

1.1 Observe

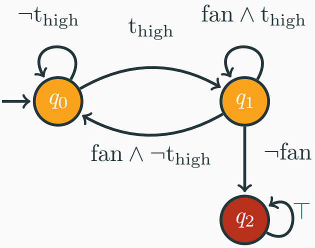
t_{high}	$\top$
fan	$\bot$

1.2 Eval

$\neg t_{high}$	$\bot$
t_{high}	$\top$

MONITORING USING AUTOMATA ↪ EXAMPLE

"Fan must always be turned on when temperature is high"



$$\boldsymbol{G}(\mathrm{t_{high}} \implies \boldsymbol{X}\mathrm{fan})$$

1. At $t = 1$, from q_0:

1.1 Observe

$\mathrm{t_{high}}$	$\top$
fan	$\bot$

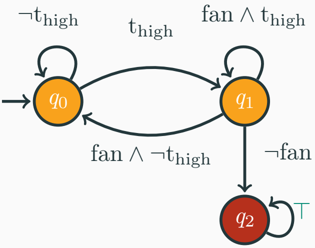
1.2 Eval

$\neg\mathrm{t_{high}}$	$\bot$
$\mathrm{t_{high}}$	$\top$

2. At $t = 2$, from q_1:

MONITORING USING AUTOMATA ↪ EXAMPLE

"Fan must always be turned on when temperature is high"



$$G(t_{high} \implies X fan)$$

1. At $t = 1$, from q_0:

   1.1 Observe

t_{high}	$\top$
fan	$\bot$

   1.2 Eval

$\neg t_{high}$	$\bot$
t_{high}	$\top$

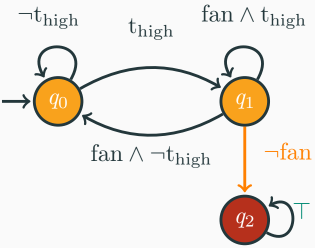
2. At $t = 2$, from q_1:

   2.1 Observe

t_{high}	$\top$
fan	$\bot$

Monitoring using automata ↪ Example

"Fan must always be turned on when temperature is high"



$$\boldsymbol{G}(t_{high} \implies \boldsymbol{X} fan)$$

1. At $t = 1$, from q_0:

   1.1 Observe

t_{high}	$\top$
fan	$\bot$

   1.2 Eval

$\neg t_{high}$	$\bot$
t_{high}	$\top$

2. At $t = 2$, from q_1:

   2.1 Observe

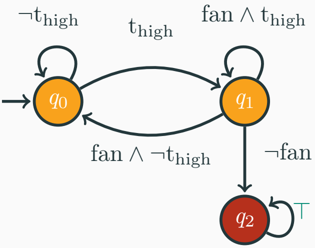
t_{high}	$\top$
fan	$\bot$

   2.2 Eval

$fan \wedge \neg t_{high}$	$\bot$
$fan \wedge t_{high}$	$\bot$
$\neg fan$	$\top$

MONITORING USING AUTOMATA ↪ EXAMPLE

"Fan must always be turned on when temperature is high"



$$G(t_{high} \implies X fan)$$

1. At $t = 1$, from q_0:

   1.1 Observe

t_{high}	$\top$
fan	$\bot$

   1.2 Eval

$\neg t_{high}$	$\bot$
t_{high}	$\top$

2. At $t = 2$, from q_1:

   2.1 Observe

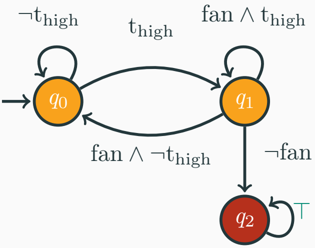
t_{high}	$\top$
fan	$\bot$

   2.2 Eval

$fan \wedge \neg t_{high}$	$\bot$
$fan \wedge t_{high}$	$\bot$
$\neg fan$	$\top$

Monitoring using Automata ↪ Example

"Fan must always be turned on when temperature is high"



$$\boldsymbol{G}(\mathrm{t_{high}} \implies \boldsymbol{X}\mathrm{fan})$$

1. At $t = 1$, from q_0:

   1.1 Observe

$\mathrm{t_{high}}$	$\top$
fan	$\bot$

   1.2 Eval

$\neg\mathrm{t_{high}}$	$\bot$
$\mathrm{t_{high}}$	$\top$

2. At $t = 2$, from q_1:

   2.1 Observe

$\mathrm{t_{high}}$	$\top$
fan	$\bot$

   2.2 Eval

$\mathrm{fan} \wedge \neg\mathrm{t_{high}}$	$\bot$
$\mathrm{fan} \wedge \mathrm{t_{high}}$	$\bot$
$\neg\mathrm{fan}$	$\top$

Monitoring this property requires a central observation point!

DECENTRALIZED MONITORING ↪ PROBLEM STATEMENT

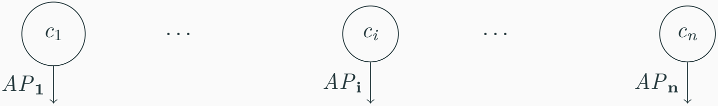
- General setting

DECENTRALIZED MONITORING ↪ PROBLEM STATEMENT

- General setting
  - $\mathcal{C} = \{c_0, \ldots, c_n\}$: components

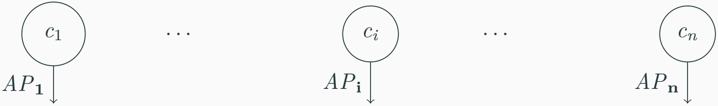
DECENTRALIZED MONITORING ↪ PROBLEM STATEMENT

- General setting
  - $\mathcal{C} = \{c_0, \ldots, c_n\}$: components
  - $AP = AP_0 \cup \ldots \cup AP_n$: atomic propositions, partitioned by $\mathcal{C}$



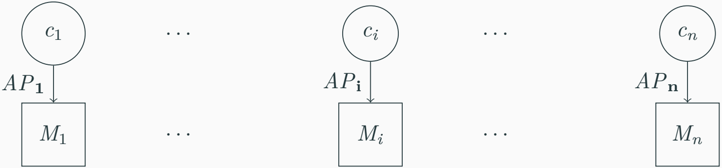
DECENTRALIZED MONITORING ↪ PROBLEM STATEMENT

- General setting
  - $\mathcal{C} = \{c_0, \ldots, c_n\}$: components
  - $AP = AP_0 \cup \ldots \cup AP_n$: atomic propositions, partitioned by $\mathcal{C}$
  - no central observation point



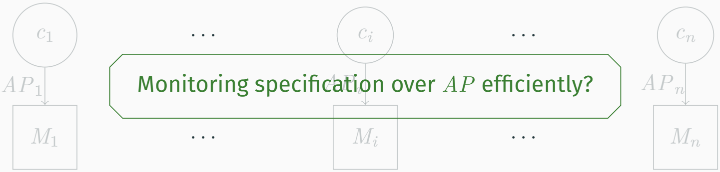
DECENTRALIZED MONITORING ↪ PROBLEM STATEMENT

- General setting
  - $\mathcal{C} = \{c_0, \ldots, c_n\}$: components
  - $AP = AP_0 \cup \ldots \cup AP_n$: atomic propositions, partitioned by $\mathcal{C}$
  - no central observation point
  - but monitors attached to components



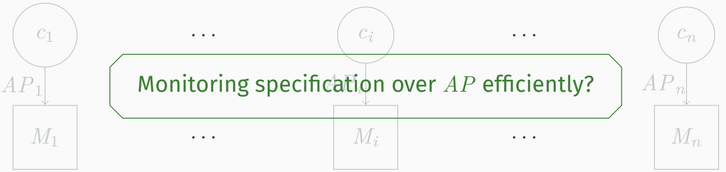
DECENTRALIZED MONITORING ↪ PROBLEM STATEMENT

- General setting



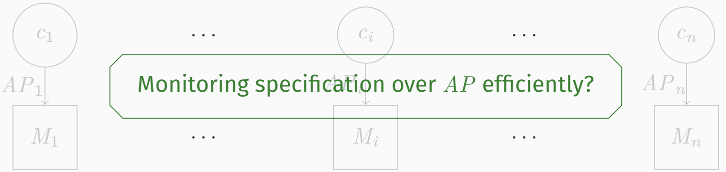
DECENTRALIZED MONITORING ↪ PROBLEM STATEMENT

- General setting
- Issues in decentralized monitoring



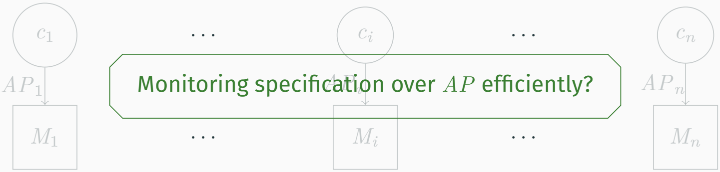
Decentralized Monitoring ↪ Problem statement

- General setting
- Issues in decentralized monitoring
  - partial views of AP – unknown global state



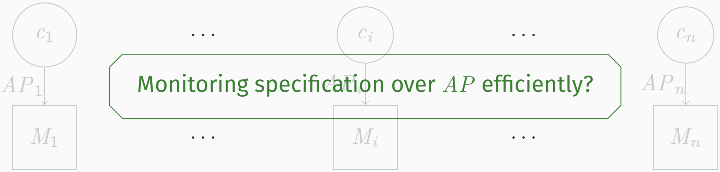
DECENTRALIZED MONITORING ↪ PROBLEM STATEMENT

- General setting
- Issues in decentralized monitoring
  - partial views of AP – unknown global state
  - partial execution of the automaton (evaluation)



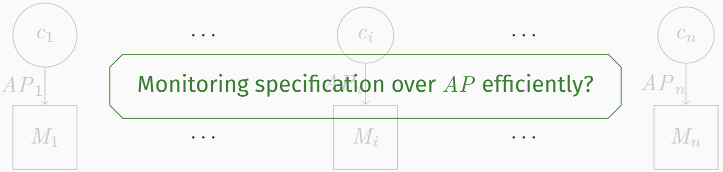
DECENTRALIZED MONITORING ↪ PROBLEM STATEMENT

- General setting
- Issues in decentralized monitoring
  - partial views of AP – unknown global state
  - partial execution of the automaton (evaluation)
  - communication between monitors



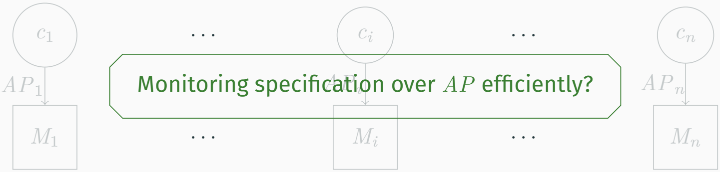
DECENTRALIZED MONITORING ↪ PROBLEM STATEMENT

- General setting
- Issues in decentralized monitoring
  - partial views of AP – unknown global state
  - partial execution of the automaton (evaluation)
  - communication between monitors
- Existing approaches:



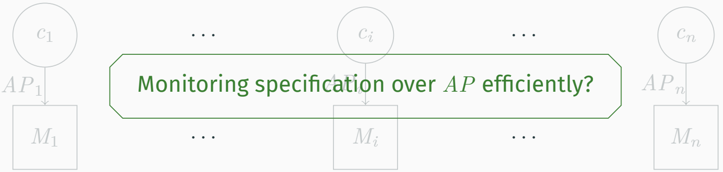
DECENTRALIZED MONITORING ↪ PROBLEM STATEMENT

- General setting
- Issues in decentralized monitoring
  - partial views of AP – unknown global state
  - partial execution of the automaton (evaluation)
  - communication between monitors
- Existing approaches:
  - based on LTL rewriting — unpredictability of monitor performance



DECENTRALIZED MONITORING ↪ PROBLEM STATEMENT

- General setting
- Issues in decentralized monitoring
  - partial views of AP – unknown global state
  - partial execution of the automaton (evaluation)
  - communication between monitors
- Existing approaches:
  - based on LTL rewriting — unpredictability of monitor performance
  - all monitors check the same specification — inefficiency



GOALS

Define a methodology of design and evaluation of decentralized monitoring

GOALS

Define a methodology of design and evaluation of decentralized monitoring

1. Aim for predictable behavior

GOALS

Define a methodology of design and evaluation of decentralized monitoring

1. Aim for predictable behavior
   - Move from LTL $\rightarrow$ Automata

GOALS

Define a methodology of design and evaluation of decentralized monitoring

1. Aim for predictable behavior
   - Move from LTL → Automata
   - Common ground to compare existing (and future) strategies

GOALS

Define a methodology of design and evaluation of decentralized monitoring

1. Aim for predictable behavior
   - Move from LTL → Automata
   - Common ground to compare existing (and future) strategies
2. Separate monitor synthesis from monitoring strategies

GOALS

Define a methodology of design and evaluation of decentralized monitoring

1. Aim for predictable behavior
   - Move from LTL → Automata
   - Common ground to compare existing (and future) strategies
2. Separate monitor synthesis from monitoring strategies
   - Centralized specification → Decentralized specification

GOALS

Define a methodology of design and evaluation of decentralized monitoring

1. Aim for predictable behavior
   - Move from LTL → Automata
   - Common ground to compare existing (and future) strategies
2. Separate monitor synthesis from monitoring strategies
   - Centralized specification → Decentralized specification
   - Monitorability of a decentralized specification

GOALS

Define a methodology of design and evaluation of decentralized monitoring

1. Aim for predictable behavior
   - Move from LTL → Automata
   - Common ground to compare existing (and future) strategies
2. Separate monitor synthesis from monitoring strategies
   - Centralized specification → Decentralized specification
   - Monitorability of a decentralized specification
   - Define a general decentralized monitoring algorithm

GOALS

Define a methodology of design and evaluation of decentralized monitoring

1. Aim for predictable behavior
   - Move from LTL → Automata
   - Common ground to compare existing (and future) strategies
2. Separate monitor synthesis from monitoring strategies
   - Centralized specification → Decentralized specification
   - Monitorability of a decentralized specification
   - Define a general decentralized monitoring algorithm

⋆ Extend tooling support for the design methodology

GOALS

Define a methodology of design and evaluation of decentralized monitoring

1. Aim for predictable behavior
   - Move from LTL → Automata
   - Common ground to compare existing (and future) strategies
2. Separate monitor synthesis from monitoring strategies
   - Centralized specification → Decentralized specification
   - Monitorability of a decentralized specification
   - Define a general decentralized monitoring algorithm

⋆ Extend tooling support for the design methodology

⋆ Ensure reproducibility

(Decentralized) Monitoring

Monitoring with EHEs

Monitoring Decentralized Specifications

The THEMIS Approach

Conclusions

Monitoring with EHEs

EXECUTION HISTORY ENCODING ↪ INFORMATION AS ATOMS

⋆ Encode the execution as a datastructure that

EXECUTION HISTORY ENCODING ↪ INFORMATION AS ATOMS

- ⋆ Encode the execution as a datastructure that
  - supports flexibility when receiving partial information

EXECUTION HISTORY ENCODING ↪ INFORMATION AS ATOMS

★ Encode the execution as a datastructure that

- supports flexibility when receiving partial information
- is insensitive to the reception order of information

Execution History Encoding ↪ Information as Atoms

- ⋆ Encode the execution as a datastructure that
  - supports flexibility when receiving partial information
  - is insensitive to the reception order of information
  - has predictable size and operations

EXECUTION HISTORY ENCODING $\hookrightarrow$ INFORMATION AS ATOMS

- ⋆ Encode the execution as a datastructure that
  - supports flexibility when receiving partial information
  - is insensitive to the reception order of information
  - has predictable size and operations
- Atomic propositions → Atoms

Execution History Encoding ↪ Information as Atoms

⋆ Encode the execution as a datastructure that
  - supports flexibility when receiving partial information
  - is insensitive to the reception order of information
  - has predictable size and operations
- Atomic propositions → Atoms
  - Allow algorithms to add data to observations ($\mathrm{enc} : AP \rightarrow Atoms$)

Execution History Encoding ↪ Information as Atoms

- ⋆ Encode the execution as a datastructure that
  - supports flexibility when receiving partial information
  - is insensitive to the reception order of information
  - has predictable size and operations
- Atomic propositions → Atoms
  - Allow algorithms to add data to observations ($\text{enc} : AP \rightarrow Atoms$)
  - Ordering information (timestamp, round number, vector clock etc.)

Execution History Encoding ↪ Information as Atoms

- ⋆ Encode the execution as a datastructure that
    - supports flexibility when receiving partial information
    - is insensitive to the reception order of information
    - has predictable size and operations
- Atomic propositions → Atoms
    - Allow algorithms to add data to observations ($\text{enc} : AP \rightarrow Atoms$)
    - Ordering information (timestamp, round number, vector clock etc.)
- Monitors store Atoms in their Memory

EXECUTION HISTORY ENCODING $\hookrightarrow$ INFORMATION AS ATOMS

- ⋆ Encode the execution as a datastructure that
  - supports flexibility when receiving partial information
  - is insensitive to the reception order of information
  - has predictable size and operations
- Atomic propositions $\rightarrow$ Atoms
  - Allow algorithms to add data to observations ($\text{enc} : AP \rightarrow Atoms$)
  - Ordering information (timestamp, round number, vector clock etc.)
- Monitors store Atoms in their Memory
- Monitors need to evaluate $Expr_{Atoms}$

Execution History Encoding ↪ Information as Atoms

- ⋆ Encode the execution as a datastructure that
  - supports flexibility when receiving partial information
  - is insensitive to the reception order of information
  - has predictable size and operations
- Atomic propositions → Atoms
  - Allow algorithms to add data to observations ($\text{enc} : AP \rightarrow Atoms$)
  - Ordering information (timestamp, round number, vector clock etc.)
- Monitors store Atoms in their Memory
- Monitors need to evaluate $Expr_{Atoms}$
  - rewrite using Memory

EXECUTION HISTORY ENCODING ↪ INFORMATION AS ATOMS

- ⋆ Encode the execution as a datastructure that
  - supports flexibility when receiving partial information
  - is insensitive to the reception order of information
  - has predictable size and operations
- Atomic propositions → Atoms
  - Allow algorithms to add data to observations (enc : $AP \rightarrow Atoms$)
  - Ordering information (timestamp, round number, vector clock etc.)
- Monitors store Atoms in their Memory
- Monitors need to evaluate $Expr_{Atoms}$
  - rewrite using Memory
  - simplify using Boolean logics (much easier than simplification for LTL)

EXECUTION HISTORY ENCODING $\hookrightarrow$ INFORMATION AS ATOMS

- ⋆ Encode the execution as a datastructure that
  - supports flexibility when receiving partial information
  - is insensitive to the reception order of information
  - has predictable size and operations
- Atomic propositions $\rightarrow$ Atoms
  - Allow algorithms to add data to observations ($\text{enc} : AP \rightarrow Atoms$)
  - Ordering information (timestamp, round number, vector clock etc.)
- Monitors store Atoms in their Memory
- Monitors need to evaluate $Expr_{Atoms}$
  - rewrite using Memory
  - simplify using Boolean logics (much easier than simplification for LTL)

$$Expr_{Atoms} \times Mem \rightarrow \mathbb{B}_3$$

$$\text{eval}(expr, \mathcal{M}) = \text{simplify}(\text{rw}(expr, \mathcal{M}))$$

$$\text{eval}(\langle 1, \text{t}_{\text{high}} \rangle \wedge \langle 2, \text{fan} \rangle, [\langle 1, \text{t}_{\text{high}} \rangle \mapsto \bot]) = \bot \wedge \langle 2, \text{fan} \rangle = \bot$$

EXECUTION HISTORY ENCODING ↪ AUTOMATA EXECUTION

- EHE is a partial function:

$$\mathcal{I} :$$

$$\mathcal{I}($$

Execution History Encoding ↪ Automata Execution

- EHE is a partial function:

$$\mathcal{I} : \mathbb{N}$$

$$\mathcal{I}(t$$

  - For a given timestamp t

Execution History Encoding ↪ Automata Execution

- EHE is a partial function:

$$\mathcal{I} : \mathbb{N} \times Q_{\mathcal{A}}$$
$$\mathcal{I}(t, q)$$

  - For a given timestamp t
  - The automaton is in state q iff

EXECUTION HISTORY ENCODING ↪ AUTOMATA EXECUTION

- EHE is a partial function:

$$\mathcal{I} : \mathbb{N} \times Q_{\mathcal{A}} \rightarrow \mathit{Expr}_{\mathit{Atoms}}$$
$$\mathcal{I}(t, q) = \mathit{expr}$$

  - For a given timestamp t
  - The automaton is in state q iff
  - $\text{eval}(\mathit{expr}, \mathcal{M}) = \top$

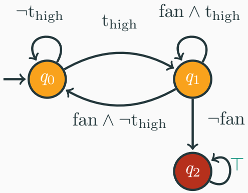
EXECUTION HISTORY ENCODING $\hookrightarrow$ AUTOMATA EXECUTION

- EHE is a partial function:

$$\mathcal{I} : \mathbb{N} \times Q_{\mathcal{A}} \to \mathit{Expr}_{\mathit{Atoms}}$$
$$\mathcal{I}(t, q) = \mathit{expr}$$

  - For a given timestamp t
  - The automaton is in state q iff
  - $\text{eval}(\mathit{expr}, \mathcal{M}) = \top$

$$\mathcal{I}(2, q_0) = [\neg\langle 1, \text{t}_{\text{high}}\rangle \wedge \neg\langle 2, \text{t}_{\text{high}}\rangle] \vee [\langle 1, \text{t}_{\text{high}}\rangle \wedge (\langle 2, \text{fan}\rangle \wedge \neg\langle 2, \text{t}_{\text{high}}\rangle)]$$



EXECUTION HISTORY ENCODING ↪ AUTOMATA EXECUTION

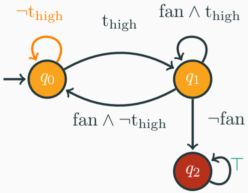
- EHE is a partial function:

$$\mathcal{I} : \mathbb{N} \times Q_{\mathcal{A}} \rightarrow Expr_{Atoms}$$
$$\mathcal{I}(t, q) = expr$$

  - For a given timestamp t
  - The automaton is in state q iff
  - $\text{eval}(expr, \mathcal{M}) = \top$

$$\mathcal{I}(2, q_0) = [\neg\langle 1, \text{t}_{\text{high}}\rangle \wedge \neg\langle 2, \text{t}_{\text{high}}\rangle] \vee [\langle 1, \text{t}_{\text{high}}\rangle \wedge (\langle 2, \text{fan}\rangle \wedge \neg\langle 2, \text{t}_{\text{high}}\rangle)]$$

$$\text{eval}(\mathcal{I}(2, q_0), [\langle 1, \text{t}_{\text{high}}\rangle \mapsto \bot]) = \text{eval}(\neg\langle 2, \text{t}_{\text{high}}\rangle, \ldots) = ?$$



Execution History Encoding $\hookrightarrow$ Automata Execution

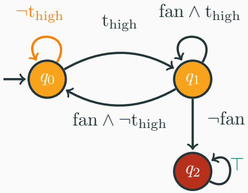
- EHE is a partial function:

$$\mathcal{I} : \mathbb{N} \times Q_{\mathcal{A}} \rightarrow Expr_{Atoms}$$
$$\mathcal{I}(t, q) = expr$$

  - For a given timestamp t
  - The automaton is in state q iff
  - $\text{eval}(expr, \mathcal{M}) = \top$

$$\mathcal{I}(2, q_0) = [\neg\langle 1, \text{t}_{\text{high}}\rangle \wedge \neg\langle 2, \text{t}_{\text{high}}\rangle] \vee [\langle 1, \text{t}_{\text{high}}\rangle \wedge (\langle 2, \text{fan}\rangle \wedge \neg\langle 2, \text{t}_{\text{high}}\rangle)]$$

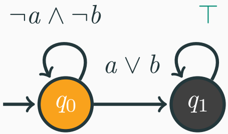
$$\text{eval}(\mathcal{I}(2, q_0), [\langle 1, \text{t}_{\text{high}}\rangle \mapsto \bot]) = \text{eval}(\neg\langle 2, \text{t}_{\text{high}}\rangle, \ldots) = ?$$



- EHE is constructed recursively and lazily (as needed and on-the-fly) using $\mathcal{A}$

EXECUTION HISTORY ENCODING ↪ CONSTRUCTION

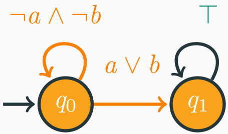
$$\mathcal{I}^2 = \text{mov}([0 \mapsto q_0 \mapsto \top], 0, 2)$$



t	q	expr
0	q_0	$\top$

EXECUTION HISTORY ENCODING ↪ CONSTRUCTION

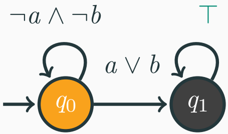
$$\mathcal{I}^2 = \text{mov}([0 \mapsto q_0 \mapsto \top], 0, 2)$$



t	q	expr
0	q_0	$\top$

EXECUTION HISTORY ENCODING ↪ CONSTRUCTION

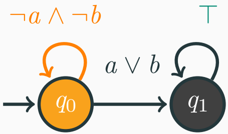
$$\mathcal{I}^2 = \text{mov}([0 \mapsto q_0 \mapsto \top], 0, 2)$$



t	q	expr
0	q_0	$\top$
1	q_0	
1	q_1	

EXECUTION HISTORY ENCODING $\hookrightarrow$ CONSTRUCTION

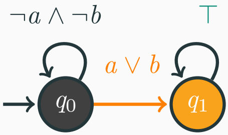
$$\mathcal{I}^2 = \text{mov}([0 \mapsto q_0 \mapsto \top], 0, 2)$$



t	q	expr
0	q_0	$\top$
1	q_0	$\top \wedge \quad \neg\langle 1, a\rangle \wedge \neg\langle 1, b\rangle$
1	q_1	

Execution History Encoding ↪ Construction

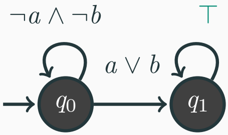
$$\mathcal{I}^2 = \text{mov}([0 \mapsto q_0 \mapsto \top], 0, 2)$$



t	q	expr
0	q_0	$\top$
1	q_0	$\neg\langle 1, a\rangle \wedge \neg\langle 1, b\rangle$
1	q_1	$\langle 1, a\rangle \vee \langle 1, b\rangle$

EXECUTION HISTORY ENCODING ↪ CONSTRUCTION

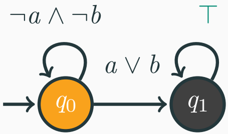
$$\mathcal{I}^2 = \text{mov}([0 \mapsto q_0 \mapsto \top], 0, 2)$$



t	q	expr
0	q_0	$\top$
1	q_0	$\neg\langle 1, a\rangle \wedge \neg\langle 1, b\rangle$
1	q_1	$\langle 1, a\rangle \vee \langle 1, b\rangle$
2	q_0	
2	q_1	

EXECUTION HISTORY ENCODING $\hookrightarrow$ CONSTRUCTION

$$\mathcal{I}^2 = \text{mov}([0 \mapsto q_0 \mapsto \top], 0, 2)$$

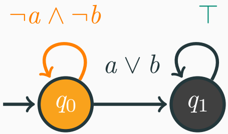


t	q	expr
0	q_0	$\top$
1	q_0	$\neg\langle 1, a\rangle \wedge \neg\langle 1, b\rangle$
1	q_1	$\langle 1, a\rangle \vee \langle 1, b\rangle$
2	q_0	$(\neg\langle 1, a\rangle \wedge \neg\langle 1, b\rangle)$
2	q_1	

$\vdots$

EXECUTION HISTORY ENCODING $\hookrightarrow$ CONSTRUCTION

$$\mathcal{I}^2 = \text{mov}([0 \mapsto q_0 \mapsto \top], 0, 2)$$

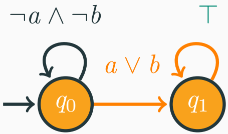


t	q	expr
0	q_0	$\top$
1	q_0	$\neg\langle 1, a\rangle \wedge \neg\langle 1, b\rangle$
1	q_1	$\langle 1, a\rangle \vee \langle 1, b\rangle$
2	q_0	$(\neg\langle 1, a\rangle \wedge \neg\langle 1, b\rangle) \wedge (\neg\langle 2, a\rangle \wedge \neg\langle 2, b\rangle)$
2	q_1	

$\vdots$

Execution History Encoding ↪ Construction

$$\mathcal{I}^2 = \text{mov}([0 \mapsto q_0 \mapsto \top], 0, 2)$$

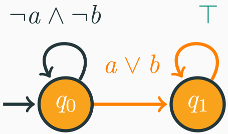


t	q	expr
0	q_0	$\top$
1	q_0	$\neg\langle 1, a\rangle \wedge \neg\langle 1, b\rangle$
1	q_1	$\langle 1, a\rangle \vee \langle 1, b\rangle$
2	q_0	$(\neg\langle 1, a\rangle \wedge \neg\langle 1, b\rangle) \wedge (\neg\langle 2, a\rangle \wedge \neg\langle 2, b\rangle)$
2	q_1	$[(\neg\langle 1, a\rangle \wedge \neg\langle 1, b\rangle) \quad] \vee [(\langle 1, a\rangle \vee \langle 1, b\rangle) \quad]$

⋮

EXECUTION HISTORY ENCODING $\hookrightarrow$ CONSTRUCTION

$$\mathcal{I}^2 = \text{mov}([0 \mapsto q_0 \mapsto \top], 0, 2)$$



t	q	expr
0	q_0	$\top$
1	q_0	$\neg\langle 1, a\rangle \wedge \neg\langle 1, b\rangle$
1	q_1	$\langle 1, a\rangle \vee \langle 1, b\rangle$
2	q_0	$(\neg\langle 1, a\rangle \wedge \neg\langle 1, b\rangle) \wedge (\neg\langle 2, a\rangle \wedge \neg\langle 2, b\rangle)$
2	q_1	$[(\neg\langle 1, a\rangle \wedge \neg\langle 1, b\rangle) \wedge (\langle 2, a\rangle \vee \langle 2, b\rangle)] \vee [(\langle 1, a\rangle \vee \langle 1, b\rangle) \wedge \top]$

$\vdots$

Execution History Encoding ↪ Properties

1. Soundness (provided that observations can be totally ordered)

Execution History Encoding ↪ Properties

1. Soundness (provided that observations can be totally ordered)
    - For the same trace, EHE and $\mathcal{A}$ report the same state

EXECUTION HISTORY ENCODING ↪ PROPERTIES

1. Soundness (provided that observations can be totally ordered)
   - For the same trace, EHE and $\mathcal{A}$ report the same state

   → They find the same verdict

Execution History Encoding $\hookrightarrow$ Properties

1. Soundness (provided that observations can be totally ordered)
   - For the same trace, EHE and $\mathcal{A}$ report the same state

   → They find the same verdict
2. Strong Eventual Consistency (SEC)

Execution History Encoding $\hookrightarrow$ Properties

1. Soundness (provided that observations can be totally ordered)
   - For the same trace, EHE and $\mathcal{A}$ report the same state

   $\rightarrow$ They find the same verdict
2. Strong Eventual Consistency (SEC)
   - We can merge EHEs by disjoining ($\vee$) each entry $\langle t, q\rangle$

EXECUTION HISTORY ENCODING $\hookrightarrow$ PROPERTIES

1. Soundness (provided that observations can be totally ordered)
   - For the same trace, EHE and $\mathcal{A}$ report the same state

   $\rightarrow$ They find the same verdict
2. Strong Eventual Consistency (SEC)
   - We can merge EHEs by disjoining ($\vee$) each entry $\langle t, q \rangle$
   - $\vee$ is commutative, associative and idempotent

Execution History Encoding ↪ Properties

1. Soundness (provided that observations can be totally ordered)
   - For the same trace, EHE and $\mathcal{A}$ report the same state

   → They find the same verdict
2. Strong Eventual Consistency (SEC)
   - We can merge EHEs by disjoining ($\vee$) each entry $\langle t, q \rangle$
   - $\vee$ is commutative, associative and idempotent

   → EHE is a state-based replicated data-type (CvRDT)

Execution History Encoding ↪ Properties

1. Soundness (provided that observations can be totally ordered)
    - For the same trace, EHE and $\mathcal{A}$ report the same state

   → They find the same verdict

2. Strong Eventual Consistency (SEC)
    - We can merge EHEs by disjoining ($\vee$) each entry $\langle t, q \rangle$
    - $\vee$ is commutative, associative and idempotent

   → EHE is a state-based replicated data-type (CvRDT)

   → Monitors that exchange their EHE find the same verdict

EXECUTION HISTORY ENCODING ↪ PROPERTIES

1. Soundness (provided that observations can be totally ordered)
   - For the same trace, EHE and $\mathcal{A}$ report the same state

   → They find the same verdict

2. Strong Eventual Consistency (SEC)
   - We can merge EHEs by disjoining ($\vee$) each entry $\langle t, q \rangle$
   - $\vee$ is commutative, associative and idempotent

   → EHE is a state-based replicated data-type (CvRDT)

   → Monitors that exchange their EHE find the same verdict

   → Can monitor centralized specification shared with multiple monitors

Execution History Encoding ↪ Properties

1. Soundness (provided that observations can be totally ordered)
   - For the same trace, EHE and $\mathcal{A}$ report the same state

   → They find the same verdict
2. Strong Eventual Consistency (SEC)
   - We can merge EHEs by disjoining ($\vee$) each entry $\langle t, q \rangle$
   - $\vee$ is commutative, associative and idempotent

   → EHE is a state-based replicated data-type (CvRDT)

   → Monitors that exchange their EHE find the same verdict

   → Can monitor centralized specification shared with multiple monitors
3. Predictable size

EXECUTION HISTORY ENCODING ↪ PROPERTIES

1. Soundness (provided that observations can be totally ordered)
    - For the same trace, EHE and $\mathcal{A}$ report the same state
    - → They find the same verdict
2. Strong Eventual Consistency (SEC)
    - We can merge EHEs by disjoining ($\vee$) each entry $\langle t, q \rangle$
    - $\vee$ is commutative, associative and idempotent
    - → EHE is a state-based replicated data-type (CvRDT)
    - → Monitors that exchange their EHE find the same verdict
    - → Can monitor centralized specification shared with multiple monitors
3. Predictable size
    - The EHE encodes all potential and past states, as needed

Execution History Encoding $\hookrightarrow$ Properties

1. Soundness (provided that observations can be totally ordered)
    - For the same trace, EHE and $\mathcal{A}$ report the same state

    $\rightarrow$ They find the same verdict
2. Strong Eventual Consistency (SEC)
    - We can merge EHEs by disjoining ($\vee$) each entry $\langle t, q \rangle$
    - $\vee$ is commutative, associative and idempotent

    $\rightarrow$ EHE is a state-based replicated data-type (CvRDT)

    $\rightarrow$ Monitors that exchange their EHE find the same verdict

    $\rightarrow$ Can monitor centralized specification shared with multiple monitors
3. Predictable size
    - The EHE encodes all potential and past states, as needed
    - The more we keep track of potential states, the bigger the size

Execution History Encoding ↪ Properties

1. Soundness (provided that observations can be totally ordered)
   - For the same trace, EHE and $\mathcal{A}$ report the same state

   → They find the same verdict
2. Strong Eventual Consistency (SEC)
   - We can merge EHEs by disjoining ($\vee$) each entry $\langle t, q \rangle$
   - $\vee$ is commutative, associative and idempotent

   → EHE is a state-based replicated data-type (CvRDT)

   → Monitors that exchange their EHE find the same verdict

   → Can monitor centralized specification shared with multiple monitors
3. Predictable size
   - The EHE encodes all potential and past states, as needed
   - The more we keep track of potential states, the bigger the size

   → We can assess algorithms by how they manipulate the EHE

EXECUTION HISTORY ENCODING ↪ ANALYSIS

$$t \mapsto q \mapsto \top$$

$$\left\{\begin{array}{lll}
t+1 & \mapsto & \left.\begin{array}{ll} q_0 & \mapsto e_{10} \\ q_1 & \mapsto e_{11} \\ & \vdots \\ q_{|Q|-1} & \mapsto e_{1(|Q|-1)} \end{array}\right\} \\
t+2 & \mapsto & \left.\begin{array}{ll} q_0 & \mapsto e_{20} \\ & \vdots \\ q_{|Q|-1} & \mapsto e_{2(|Q|-1)} \end{array}\right\} \\
\vdots & & \\
t+\delta & \mapsto & \left.\begin{array}{ll} q_0 & \mapsto e_{\delta 0} \\ q_1 & \mapsto e_{\delta 1} \\ & \vdots \\ q_{|Q|-1} & \mapsto e_{\delta(|Q|-1)} \end{array}\right\}
\end{array}\right.$$

EXECUTION HISTORY ENCODING ↪ ANALYSIS

$$t \mapsto q \mapsto \top$$

$$\begin{cases} t+1 \mapsto & \left.\begin{array}{ll} q_0 & \mapsto e_{10} \\ q_1 & \mapsto e_{11} \\ & \vdots \\ q_{|Q|-1} & \mapsto e_{1(|Q|-1)} \end{array}\right\} |Q| \\ t+2 \mapsto & \left.\begin{array}{ll} q_0 & \mapsto e_{20} \\ & \vdots \\ q_{|Q|-1} & \mapsto e_{2(|Q|-1)} \end{array}\right\} |Q| \\ \vdots & \\ t+\delta \mapsto & \left.\begin{array}{ll} q_0 & \mapsto e_{\delta 0} \\ q_1 & \mapsto e_{\delta 1} \\ & \vdots \\ q_{|Q|-1} & \mapsto e_{\delta(|Q|-1)} \end{array}\right\} |Q| \end{cases}$$

EXECUTION HISTORY ENCODING ↪ ANALYSIS

- Information Delay (δ)

  Timestamps needed to expand before determining a state

  Potential states to keep track of

$$
\begin{array}{llll}
t & \mapsto q & \mapsto \top & \\
\delta \left\{ \begin{array}{l} t+1 \mapsto \\ t+2 \mapsto \\ \vdots \\ t+\delta \mapsto \end{array} \right. &
\begin{array}{l} q_0 \mapsto e_{10} \\ q_1 \mapsto e_{11} \\ \vdots \\ q_{|Q|-1} \mapsto e_{1(|Q|-1)} \\ q_0 \mapsto e_{20} \\ \vdots \\ q_{|Q|-1} \mapsto e_{2(|Q|-1)} \\ q_0 \mapsto e_{\delta 0} \\ q_1 \mapsto e_{\delta 1} \\ \vdots \\ q_{|Q|-1} \mapsto e_{\delta(|Q|-1)} \end{array} &
\begin{array}{l} \} |Q| \\ \} |Q| \\ \} |Q| \end{array}
\end{array}
$$

EXECUTION HISTORY ENCODING ↪ ANALYSIS

- Information Delay (δ)

  Timestamps needed to expand before determining a state

  Potential states to keep track of

- Size of expression grows with each move beyond t

$$t \mapsto q \mapsto \top$$

$$\delta \left\{ \begin{array}{lll} t+1 \mapsto & \left. \begin{array}{ll} q_0 & \mapsto e_{10} \\ q_1 & \mapsto e_{11} \\ & \vdots \\ q_{|Q|-1} & \mapsto e_{1(|Q|-1)} \end{array} \right\} & |Q| \\ t+2 \mapsto & \left. \begin{array}{ll} q_0 & \mapsto e_{20} \\ & \vdots \\ q_{|Q|-1} & \mapsto e_{2(|Q|-1)} \end{array} \right\} & |Q| \\ \vdots & & \\ t+\delta \mapsto & \left. \begin{array}{ll} q_0 & \mapsto e_{\delta 0} \\ q_1 & \mapsto e_{\delta 1} \\ & \vdots \\ q_{|Q|-1} & \mapsto e_{\delta(|Q|-1)} \end{array} \right\} & |Q| \end{array} \right.$$

Execution History Encoding ↪ Analysis

- Information Delay (δ)

  Timestamps needed to expand before determining a state

  Potential states to keep track of

- Size of expression grows with each move beyond t

- Size of EHE:

$$|\mathcal{I}^\delta| = \mathcal{O}(\delta|Q|\sum_{1}^{\delta} LP)$$
$$= \mathcal{O}(\delta^2|Q|LP)$$

$$t \mapsto q \mapsto \top$$

$$\delta \left\{ \begin{array}{lll} t+1 \mapsto & \left.\begin{array}{ll} q_0 & \mapsto e_{10} \\ q_1 & \mapsto e_{11} \\ & \vdots \\ q_{|Q|-1} & \mapsto e_{1(|Q|-1)} \end{array}\right\} & |Q| \\ t+2 \mapsto & \left.\begin{array}{ll} q_0 & \mapsto e_{20} \\ & \vdots \\ q_{|Q|-1} & \mapsto e_{2(|Q|-1)} \end{array}\right\} & |Q| \\ \vdots & & \\ t+\delta \mapsto & \left.\begin{array}{ll} q_0 & \mapsto e_{\delta 0} \\ q_1 & \mapsto e_{\delta 1} \\ & \vdots \\ q_{|Q|-1} & \mapsto e_{\delta(|Q|-1)} \end{array}\right\} & |Q| \end{array} \right.$$

Monitoring Decentralized Specifications

DECENTRALIZED SPECIFICATION

- Each monitor is associated with a tuple $\langle \mathcal{A}, c \rangle$

DECENTRALIZED SPECIFICATION

- Each monitor is associated with a tuple $\langle \mathcal{A}, c \rangle$
    - $\mathcal{A}$ is its specification automaton

DECENTRALIZED SPECIFICATION

- Each monitor is associated with a tuple $\langle \mathcal{A}, c \rangle$
  - $\mathcal{A}$ is its specification automaton
  - c is the component the monitor is attached to

DECENTRALIZED SPECIFICATION

- Each monitor is associated with a tuple $\langle \mathcal{A}, c \rangle$
  - $\mathcal{A}$ is its specification automaton
  - c is the component the monitor is attached to
- The transition labels of an automaton $\mathcal{A}$ are restricted to:

DECENTRALIZED SPECIFICATION

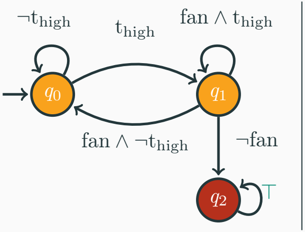
- Each monitor is associated with a tuple $\langle \mathcal{A}, c \rangle$
  - $\mathcal{A}$ is its specification automaton
  - c is the component the monitor is attached to
- The transition labels of an automaton $\mathcal{A}$ are restricted to:
  - Atomic propositions local to the attached component

DECENTRALIZED SPECIFICATION

- Each monitor is associated with a tuple $\langle \mathcal{A}, c \rangle$
  - $\mathcal{A}$ is its specification automaton
  - c is the component the monitor is attached to
- The transition labels of an automaton $\mathcal{A}$ are restricted to:
  - Atomic propositions local to the attached component
  - References to other monitors

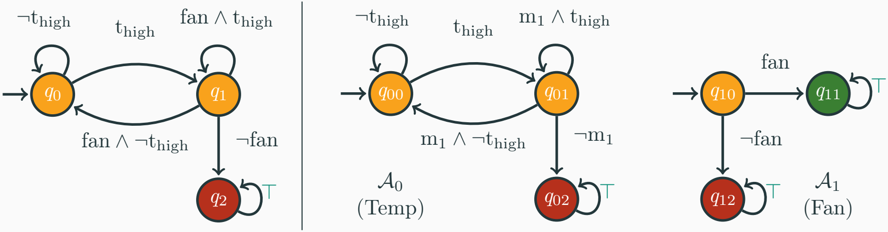
DECENTRALIZED SPECIFICATION

- Each monitor is associated with a tuple $\langle \mathcal{A}, c \rangle$
  - $\mathcal{A}$ is its specification automaton
  - c is the component the monitor is attached to
- The transition labels of an automaton $\mathcal{A}$ are restricted to:
  - Atomic propositions local to the attached component
  - References to other monitors



DECENTRALIZED SPECIFICATION

- Each monitor is associated with a tuple $\langle \mathcal{A}, c \rangle$
    - $\mathcal{A}$ is its specification automaton
    - c is the component the monitor is attached to
- The transition labels of an automaton $\mathcal{A}$ are restricted to:
    - Atomic propositions local to the attached component
    - References to other monitors



DECENTRALIZED SPECIFICATION $\hookrightarrow$ SEMANTICS & MONITORABILITY

- For an automaton $\mathcal{A}_k$, to evaluate a label m_j at t with a trace tr

DECENTRALIZED SPECIFICATION ↪ SEMANTICS & MONITORABILITY

- For an automaton $\mathcal{A}_k$, to evaluate a label m_j at t with a trace tr
  - Run tr starting with t on $\mathcal{A}_j$ starting from q_{j_0}

DECENTRALIZED SPECIFICATION ↪ SEMANTICS & MONITORABILITY

- For an automaton $\mathcal{A}_k$, to evaluate a label m_j at t with a trace tr
    - Run tr starting with t on $\mathcal{A}_j$ starting from q_{j_0}
    - Consider the verdict of the run to be the observation m_j at t

DECENTRALIZED SPECIFICATION ↪ SEMANTICS & MONITORABILITY

- For an automaton $\mathcal{A}_k$, to evaluate a label m_j at t with a trace tr
  - Run tr starting with t on $\mathcal{A}_j$ starting from q_{j_0}
  - Consider the verdict of the run to be the observation m_j at t

(!) If $\mathcal{A}_j$ never reaches a final verdict we will not be able to monitor $\mathcal{A}_k$

DECENTRALIZED SPECIFICATION ↪ SEMANTICS & MONITORABILITY

- For an automaton $\mathcal{A}_k$, to evaluate a label m_j at t with a trace tr
  - Run tr starting with t on $\mathcal{A}_j$ starting from q_{j_0}
  - Consider the verdict of the run to be the observation m_j at t

(!) If $\mathcal{A}_j$ never reaches a final verdict we will not be able to monitor $\mathcal{A}_k$

(?) Monitorability: "From any state in $\mathcal{A}_k$, we can reach a final verdict"

DECENTRALIZED SPECIFICATION ↪ SEMANTICS & MONITORABILITY

- For an automaton $\mathcal{A}_k$, to evaluate a label m_j at t with a trace tr
  - Run tr starting with t on $\mathcal{A}_j$ starting from q_{j_0}
  - Consider the verdict of the run to be the observation m_j at t

(!) If $\mathcal{A}_j$ never reaches a final verdict we will not be able to monitor $\mathcal{A}_k$

(?) Monitorability: "From any state in $\mathcal{A}_k$, we can reach a final verdict"

- monitorable($\mathcal{A}_k$) iff $\forall q \in Q_{\mathcal{A}_k}, \exists q_f \in Q_{\mathcal{A}_k}, \exists e_f \in \mathrm{paths}(q, q_f)$, s.t.

DECENTRALIZED SPECIFICATION $\hookrightarrow$ SEMANTICS & MONITORABILITY

- For an automaton $\mathcal{A}_k$, to evaluate a label m_j at t with a trace tr
    - Run tr starting with t on $\mathcal{A}_j$ starting from q_{j_0}
    - Consider the verdict of the run to be the observation m_j at t

(!) If $\mathcal{A}_j$ never reaches a final verdict we will not be able to monitor $\mathcal{A}_k$

(?) Monitorability: "From any state in $\mathcal{A}_k$, we can reach a final verdict"

- monitorable($\mathcal{A}_k$) iff $\forall q \in Q_{\mathcal{A}_k}, \exists q_f \in Q_{\mathcal{A}_k}, \exists e_f \in \mathrm{paths}(q, q_f)$, s.t.
    1. Path can be taken: e_f is satisfiable;

DECENTRALIZED SPECIFICATION ↪ SEMANTICS & MONITORABILITY

- For an automaton $\mathcal{A}_k$, to evaluate a label m_j at t with a trace tr
  - Run tr starting with t on $\mathcal{A}_j$ starting from q_{j_0}
  - Consider the verdict of the run to be the observation m_j at t

(!) If $\mathcal{A}_j$ never reaches a final verdict we will not be able to monitor $\mathcal{A}_k$

(?) Monitorability: "From any state in $\mathcal{A}_k$, we can reach a final verdict"

- monitorable($\mathcal{A}_k$) iff $\forall q \in Q_{\mathcal{A}_k}, \exists q_f \in Q_{\mathcal{A}_k}, \exists e_f \in \mathrm{paths}(q, q_f)$, s.t.
  1. Path can be taken: e_f is satisfiable;
  2. Path leads to a verdict: $\mathrm{ver}_k(q_f) \in \{\bot, \top\}$;

DECENTRALIZED SPECIFICATION ↪ SEMANTICS & MONITORABILITY

- For an automaton $\mathcal{A}_k$, to evaluate a label m_j at t with a trace tr
    - Run tr starting with t on $\mathcal{A}_j$ starting from q_{j_0}
    - Consider the verdict of the run to be the observation m_j at t

(!) If $\mathcal{A}_j$ never reaches a final verdict we will not be able to monitor $\mathcal{A}_k$

(?) Monitorability: "From any state in $\mathcal{A}_k$, we can reach a final verdict"

- $\text{monitorable}(\mathcal{A}_k)$ iff $\forall q \in Q_{\mathcal{A}_k}, \exists q_f \in Q_{\mathcal{A}_k}, \exists e_f \in \text{paths}(q, q_f)$, s.t.
    1. Path can be taken: e_f is satisfiable;
    2. Path leads to a verdict: $\text{ver}_k(q_f) \in \{\bot, \top\}$;
    3. All its dependencies are monitorable:
       $\forall \mathrm{m}_j \in \text{dep}(e_f)\colon \text{monitorable}(\mathcal{A}_j)$.

Decentralized Specification ↪ Semantics & Monitorability

- For an automaton $\mathcal{A}_k$, to evaluate a label m_j at t with a trace tr
  - Run tr starting with t on $\mathcal{A}_j$ starting from q_{j_0}
  - Consider the verdict of the run to be the observation m_j at t

(!) If $\mathcal{A}_j$ never reaches a final verdict we will not be able to monitor $\mathcal{A}_k$

(?) Monitorability: "From any state in $\mathcal{A}_k$, we can reach a final verdict"

- monitorable($\mathcal{A}_k$) iff $\forall q \in Q_{\mathcal{A}_k}, \exists q_f \in Q_{\mathcal{A}_k}, \exists e_f \in \mathrm{paths}(q, q_f)$, s.t.
  1. Path can be taken: e_f is satisfiable;
  2. Path leads to a verdict: $\mathrm{ver}_k(q_f) \in \{\bot, \top\}$;
  3. All its dependencies are monitorable:
     $\forall \mathrm{m}_j \in \mathrm{dep}(e_f)\colon \mathrm{monitorable}(\mathcal{A}_j)$.
- Expressions that determine paths between states (n = path length)

DECENTRALIZED SPECIFICATION $\hookrightarrow$ SEMANTICS & MONITORABILITY

- For an automaton $\mathcal{A}_k$, to evaluate a label m_j at t with a trace tr
  - Run tr starting with t on $\mathcal{A}_j$ starting from q_{j_0}
  - Consider the verdict of the run to be the observation m_j at t

(!) If $\mathcal{A}_j$ never reaches a final verdict we will not be able to monitor $\mathcal{A}_k$

(?) Monitorability: "From any state in $\mathcal{A}_k$, we can reach a final verdict"

- monitorable($\mathcal{A}_k$) iff $\forall q \in Q_{\mathcal{A}_k}, \exists q_f \in Q_{\mathcal{A}_k}, \exists e_f \in \mathrm{paths}(q, q_f)$, s.t.
  1. Path can be taken: e_f is satisfiable;
  2. Path leads to a verdict: $\mathrm{ver}_k(q_f) \in \{\bot, \top\}$;
  3. All its dependencies are monitorable:

     $\forall \mathrm{m}_j \in \mathrm{dep}(e_f)\colon \mathrm{monitorable}(\mathcal{A}_j)$.

- Expressions that determine paths between states (n = path length)
  - $\mathrm{paths}(q_s, q_e) = \left\{ \texttt{expr} \;\middle|\; \begin{array}{l} \exists n \in \mathbb{N} : \mathcal{I}^n(n, q_e) = \texttt{expr} \\ \wedge \mathcal{I}^n = \mathrm{mov}([0 \mapsto q_s \mapsto \top], 0, n) \end{array} \right\}$

Generalized Monitoring Algorithm ↪ Overview

1. Setup (Deploy)
    1.1 Analyze and convert the specification as necessary
    1.2 Create monitors, and assign them a specification
        (!) The monitor handles encoding of AP and Memory
    1.3 Attach monitors to components

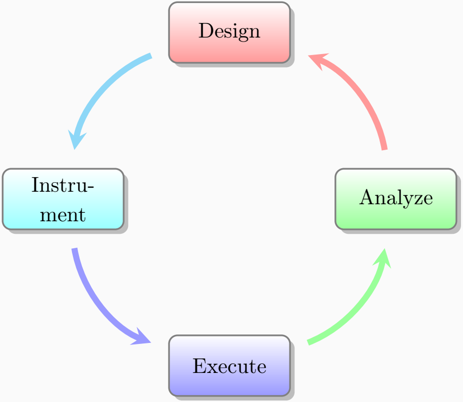
GENERALIZED MONITORING ALGORITHM ↪ OVERVIEW

1. Setup (Deploy)
   1.1 Analyze and convert the specification as necessary
   1.2 Create monitors, and assign them a specification
      (!) The monitor handles encoding of AP and Memory
   1.3 Attach monitors to components

2. Monitoring
   2.1 Wait to receive observations from attached component
   2.2 Receive messages (EHE) from monitors
   2.3 Process observations and messages (update the local EHE)
   2.4 Communicate with other monitors

The THEMIS Approach

THEMIS ↪ Overview



- **Design**: Design a monitoring algorithm
- **Instrument**: Create or re-use metrics. Metrics are automatically instrumented using AspectJ
- **Execute**: Use `THEMIS` tools to execute one or more monitoring run(s)
- **Analyze**: Measures are stored in a database for postmortem analysis

Setup

```
Map<Integer, ? extends Monitor>
↪  setup() {
  config.getSpec().put("root",
      Convert.makeAutomataSpec(
        config.getSpec().get("root")));
  Map<Integer, Monitor> mons = new
  ↪  HashMap<Integer, Monitor>();
  Integer i = 0;
  for(Component comp :
  ↪  config.getComponents()) {
    MonMigrate mon = new
    ↪  MonMigrate(i);
    attachMonitor(comp, mon);
    mons.put(i, mon);
    i++;
  }
  return mons;
}
```

Monitor

```
void monitor(int t, Memory<Atom> observations)
throws ReportVerdict, ExceptionStopMonitoring {
    m.merge(observations);
    if(receive()) isMonitoring = true;
    if(isMonitoring) {
      if(!observations.isEmpty())
        ehe.tick();
      boolean b = ehe.update(m, -1);
      if(b) {
        VerdictTimed v = ehe.scanVerdict();
        if(v.isFinal())
          throw new
          ↪  ReportVerdict(v.getVerdict(), t);
        ehe.dropResolved();
      }
      int next  = getNext();
      if(next != getID()) {
        Representation toSend = ehe.sliceLive();
        send(next, new
        ↪  RepresentationPacket(toSend));
        isMonitoring = false;
      }
    }
  }
```

EXAMPLES ↪ METRICS

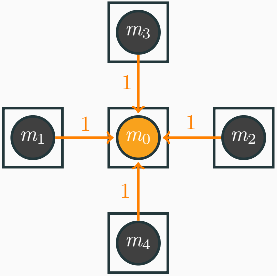
```
void setupRun(MonitoringAlgorithm alg) {
 addMeasure(new Measure("msg_num","Msgs",0L,Measures.addLong));
}
after(Integer to, Message m) : Commons.sendMessage(to, m) {
 update("msg_num" , 1L);
}
```

```
SELECT alg, comps, avg(msg_num), avg(msg_data), count(*)
FROM bench WHERE alg in ('Migration', 'MigrationRR')
GROUP BY alg, comps
```

	alg	comps	avg(msg_num)	avg(msg_data)	count(*)
1	Migration	3	2.04226336011177	267.8458714635	572600
2	Migration	4	2.16402472527473	668.129401098901	364000
3	Migration	5	3.33806822465134	3954.09705050886	530600
4	MigrationRR	3	32.7222301781348	482.572275585051	572600
5	MigrationRR	4	31.8533351648352	932.708425824176	364000
6	MigrationRR	5	19.2345269506219	4361.30746324915	530600

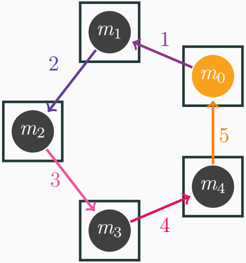
EXISTING ALGORITHMS

STUDYING EXISTING ALGORITHMS ↪ EXPECTED BEHAVIOR



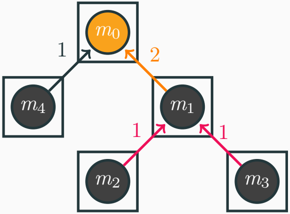
Orchestration

- δ is constant



Migration

- δ is linear in components

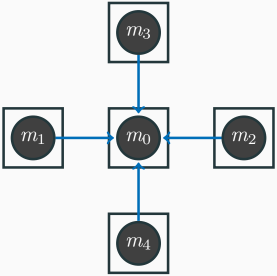


Choreography

- δ is linear in network depth (algorithm)

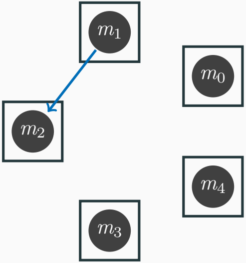
#Msgs and |Msg| are predicted on a per round basis

Studying Existing Algorithms $\hookrightarrow$ Expected Behavior



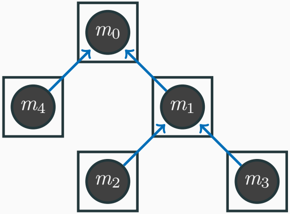
Orchestration

- δ is constant
- #Msgs is linear in components



Migration

- δ is linear in components
- #Msgs is constant

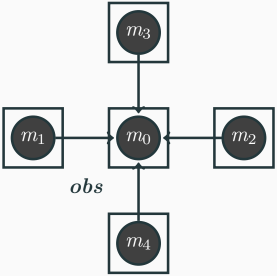


Choreography

- δ is linear in network depth (algorithm)
- #Msgs is linear in network edges

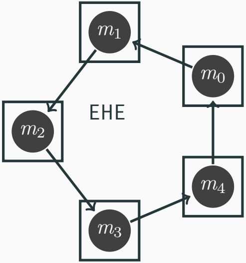
#Msgs and |Msg| are predicted on a per round basis

STUDYING EXISTING ALGORITHMS ↪ EXPECTED BEHAVIOR



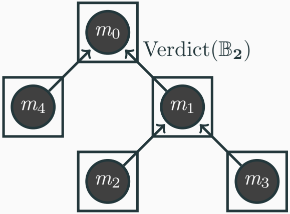
Orchestration

- δ is constant
- #Msgs is linear in components
- |Msg| constant: observations per component



Migration

- δ is linear in components
- #Msgs is constant
- |Msg| is size of EHE: $\mathcal{O}(\delta^2)$, quadratic in components



Choreography

- δ is linear in network depth (algorithm)
- #Msgs is linear in network edges
- |Msg| is constant

#Msgs and |Msg| are predicted on a per round basis

Conclusions

SUMMARY AND FUTURE WORK

- ⋆ Decentralized Monitoring of (De)Centralized Specifications
    1. Aim for predictable behavior → Automata + EHE data structure

- ⋆ Future Work

SUMMARY AND FUTURE WORK

- ⋆ Decentralized Monitoring of (De)Centralized Specifications
  1. Aim for predictable behavior $\rightarrow$ Automata + EHE data structure
  2. Separate synthesis from monitoring: decentralized specifications

- ⋆ Future Work

SUMMARY AND FUTURE WORK

- ⋆ Decentralized Monitoring of (De)Centralized Specifications
  1. Aim for predictable behavior → Automata + EHE data structure
  2. Separate synthesis from monitoring: decentralized specifications
  3. Methodology + tool support for designing, measuring, comparing and extending decentralized RV algorithms

- ⋆ Future Work

SUMMARY AND FUTURE WORK

- ⋆ Decentralized Monitoring of (De)Centralized Specifications
  1. Aim for predictable behavior → Automata + EHE data structure
  2. Separate synthesis from monitoring: decentralized specifications
  3. Methodology + tool support for designing, measuring, comparing and extending decentralized RV algorithms
  4. Adapted and compared existing algorithms
- ⋆ Future Work

SUMMARY AND FUTURE WORK

- ⋆ Decentralized Monitoring of (De)Centralized Specifications
  1. Aim for predictable behavior → Automata + EHE data structure
  2. Separate synthesis from monitoring: decentralized specifications
  3. Methodology + tool support for designing, measuring, comparing and extending decentralized RV algorithms
  4. Adapted and compared existing algorithms
- ⋆ Future Work
  1. Centralised specification → equivalent decentralized specifications

SUMMARY AND FUTURE WORK

- ⋆ Decentralized Monitoring of (De)Centralized Specifications
  1. Aim for predictable behavior → Automata + EHE data structure
  2. Separate synthesis from monitoring: decentralized specifications
  3. Methodology + tool support for designing, measuring, comparing and extending decentralized RV algorithms
  4. Adapted and compared existing algorithms
- ⋆ Future Work
  1. Centralised specification → equivalent decentralized specifications
     - Optimize existing methods

SUMMARY AND FUTURE WORK

- Decentralized Monitoring of (De)Centralized Specifications
  1. Aim for predictable behavior → Automata + EHE data structure
  2. Separate synthesis from monitoring: decentralized specifications
  3. Methodology + tool support for designing, measuring, comparing and extending decentralized RV algorithms
  4. Adapted and compared existing algorithms
- Future Work
  1. Centralised specification → equivalent decentralized specifications
     - Optimize existing methods
     - Take into account topology of the monitored system

SUMMARY AND FUTURE WORK

- ⋆ Decentralized Monitoring of (De)Centralized Specifications
  1. Aim for predictable behavior → Automata + EHE data structure
  2. Separate synthesis from monitoring: decentralized specifications
  3. Methodology + tool support for designing, measuring, comparing and extending decentralized RV algorithms
  4. Adapted and compared existing algorithms
- ⋆ Future Work
  1. Centralised specification → equivalent decentralized specifications
     - Optimize existing methods
     - Take into account topology of the monitored system
  2. Extend THEMIS

SUMMARY AND FUTURE WORK

- ⋆ Decentralized Monitoring of (De)Centralized Specifications
  1. Aim for predictable behavior → Automata + EHE data structure
  2. Separate synthesis from monitoring: decentralized specifications
  3. Methodology + tool support for designing, measuring, comparing and extending decentralized RV algorithms
  4. Adapted and compared existing algorithms
- ⋆ Future Work
  1. Centralised specification → equivalent decentralized specifications
     - Optimize existing methods
     - Take into account topology of the monitored system
  2. Extend THEMIS
     - New metrics

SUMMARY AND FUTURE WORK

- ⋆ Decentralized Monitoring of (De)Centralized Specifications
  1. Aim for predictable behavior → Automata + EHE data structure
  2. Separate synthesis from monitoring: decentralized specifications
  3. Methodology + tool support for designing, measuring, comparing and extending decentralized RV algorithms
  4. Adapted and compared existing algorithms
- ⋆ Future Work
  1. Centralised specification → equivalent decentralized specifications
     - Optimize existing methods
     - Take into account topology of the monitored system
  2. Extend THEMIS
     - New metrics
     - Support a fully-asynchronous monitoring approach

SUMMARY AND FUTURE WORK

- ⋆ Decentralized Monitoring of (De)Centralized Specifications
  1. Aim for predictable behavior → Automata + EHE data structure
  2. Separate synthesis from monitoring: decentralized specifications
  3. Methodology + tool support for designing, measuring, comparing and extending decentralized RV algorithms
  4. Adapted and compared existing algorithms
- ⋆ Future Work
  1. Centralised specification → equivalent decentralized specifications
     - Optimize existing methods
     - Take into account topology of the monitored system
  2. Extend THEMIS
     - New metrics
     - Support a fully-asynchronous monitoring approach
     - Better visualization of (the behavior of) algorithms

SUMMARY AND FUTURE WORK

- ⋆ Decentralized Monitoring of (De)Centralized Specifications
  1. Aim for predictable behavior → Automata + EHE data structure
  2. Separate synthesis from monitoring: decentralized specifications
  3. Methodology + tool support for designing, measuring, comparing and extending decentralized RV algorithms
  4. Adapted and compared existing algorithms
- ⋆ Future Work
  1. Centralised specification → equivalent decentralized specifications
     - Optimize existing methods
     - Take into account topology of the monitored system
  2. Extend THEMIS
     - New metrics
     - Support a fully-asynchronous monitoring approach
     - Better visualization of (the behavior of) algorithms
  3. Runtime enforcement of centralized and decentralized specifications

Related Work and Goals

RELATED WORK ↪ DECENTRALIZED RV

- General setting

RELATED WORK ↪ DECENTRALIZED RV

- General setting
  - $\mathcal{C}$: a set of components

RELATED WORK $\hookrightarrow$ DECENTRALIZED RV

- General setting
  - $\mathcal{C}$: a set of components
  - AP: a set of atomic propositions, partitioned by $\mathcal{C}$

RELATED WORK ↪ DECENTRALIZED RV

- General setting
  - $\mathcal{C}$: a set of components
  - AP: a set of atomic propositions, partitioned by $\mathcal{C}$
- Issues in decentralized monitoring

RELATED WORK $\hookrightarrow$ DECENTRALIZED RV

- General setting
  - $\mathcal{C}$: a set of components
  - AP: a set of atomic propositions, partitioned by $\mathcal{C}$
- Issues in decentralized monitoring
  - partial views of AP – unknown global state

RELATED WORK ↪ DECENTRALIZED RV

- General setting
  - $\mathcal{C}$: a set of components
  - AP: a set of atomic propositions, partitioned by $\mathcal{C}$
- Issues in decentralized monitoring
  - partial views of AP – unknown global state
  - partial execution of the automaton (evaluation)

RELATED WORK ↪ DECENTRALIZED RV

- General setting
  - $\mathcal{C}$: a set of components
  - AP: a set of atomic propositions, partitioned by $\mathcal{C}$
- Issues in decentralized monitoring
  - partial views of AP – unknown global state
  - partial execution of the automaton (evaluation)
  - communication between monitors

RELATED WORK ↪ DECENTRALIZED RV

- General setting
  - $\mathcal{C}$: a set of components
  - AP: a set of atomic propositions, partitioned by $\mathcal{C}$
- Issues in decentralized monitoring
  - partial views of AP – unknown global state
  - partial execution of the automaton (evaluation)
  - communication between monitors
- Rewriting-based techniques

RELATED WORK ↪ DECENTRALIZED RV

- General setting
  - $\mathcal{C}$: a set of components
  - AP: a set of atomic propositions, partitioned by $\mathcal{C}$
- Issues in decentralized monitoring
  - partial views of AP – unknown global state
  - partial execution of the automaton (evaluation)
  - communication between monitors
- Rewriting-based techniques
  - (safety) LTL [Rosu et al 05], (full) LTL [BauerFalcone12,ColomboFalcone16]

Related Work ↪ Decentralized RV

- General setting
    - $\mathcal{C}$: a set of components
    - AP: a set of atomic propositions, partitioned by $\mathcal{C}$
- Issues in decentralized monitoring
    - partial views of AP – unknown global state
    - partial execution of the automaton (evaluation)
    - communication between monitors
- Rewriting-based techniques
    - (safety) LTL [Rosu et al 05], (full) LTL [BauerFalcone12,ColomboFalcone16]
    - (safety) MTTL (real-time systems) [ThatiRosu05,Basin et al 15]

RELATED WORK ↪ DECENTRALIZED RV

- General setting
  - $\mathcal{C}$: a set of components
  - AP: a set of atomic propositions, partitioned by $\mathcal{C}$
- Issues in decentralized monitoring
  - partial views of AP – unknown global state
  - partial execution of the automaton (evaluation)
  - communication between monitors
- Rewriting-based techniques
  - (safety) LTL [Rosu et al 05], (full) LTL [BauerFalcone12,ColomboFalcone16]
  - (safety) MTTL (real-time systems) [ThatiRosu05,Basin et al 15]
  - Common assumptions

RELATED WORK $\hookrightarrow$ DECENTRALIZED RV

- General setting
  - $\mathcal{C}$: a set of components
  - AP: a set of atomic propositions, partitioned by $\mathcal{C}$
- Issues in decentralized monitoring
  - partial views of AP – unknown global state
  - partial execution of the automaton (evaluation)
  - communication between monitors
- Rewriting-based techniques
  - (safety) LTL [Rosu et al 05], (full) LTL [BauerFalcone12,ColomboFalcone16]
  - (safety) MTTL (real-time systems) [ThatiRosu05,Basin et al 15]
  - Common assumptions
    - Reliable network with fully-connected components

Related Work ↪ Decentralized RV

- General setting
  - $\mathcal{C}$: a set of components
  - AP: a set of atomic propositions, partitioned by $\mathcal{C}$
- Issues in decentralized monitoring
  - partial views of AP – unknown global state
  - partial execution of the automaton (evaluation)
  - communication between monitors
- Rewriting-based techniques
  - (safety) LTL [Rosu et al 05], (full) LTL [BauerFalcone12,ColomboFalcone16]
  - (safety) MTTL (real-time systems) [ThatiRosu05,Basin et al 15]
  - Common assumptions
    - Reliable network with fully-connected components
    - Global clock

Related Work ↪ Decentralized RV

- General setting
  - $\mathcal{C}$: a set of components
  - AP: a set of atomic propositions, partitioned by $\mathcal{C}$
- Issues in decentralized monitoring
  - partial views of AP – unknown global state
  - partial execution of the automaton (evaluation)
  - communication between monitors
- Rewriting-based techniques
  - (safety) LTL [Rosu et al 05], (full) LTL [BauerFalcone12,ColomboFalcone16]
  - (safety) MTTL (real-time systems) [ThatiRosu05,Basin et al 15]
  - Common assumptions
    - Reliable network with fully-connected components
    - Global clock
    - Oblivious to order of messages

RELATED WORK ↪ DECENTRALIZED RV

- General setting
  - $\mathcal{C}$: a set of components
  - AP: a set of atomic propositions, partitioned by $\mathcal{C}$
- Issues in decentralized monitoring
  - partial views of AP – unknown global state
  - partial execution of the automaton (evaluation)
  - communication between monitors
- Rewriting-based techniques
  - (safety) LTL [Rosu et al 05], (full) LTL [BauerFalcone12,ColomboFalcone16]
  - (safety) MTTL (real-time systems) [ThatiRosu05,Basin et al 15]
  - Common assumptions
    - Reliable network with fully-connected components
    - Global clock
    - Oblivious to order of messages

(!) Unpredictable runtime behavior of rewriting

RELATED WORK ↪ DECENTRALIZED RV

- General setting
  - $\mathcal{C}$: a set of components
  - AP: a set of atomic propositions, partitioned by $\mathcal{C}$
- Issues in decentralized monitoring
  - partial views of AP – unknown global state
  - partial execution of the automaton (evaluation)
  - communication between monitors
- Rewriting-based techniques
  - (safety) LTL [Rosu et al 05], (full) LTL [BauerFalcone12,ColomboFalcone16]
  - (safety) MTTL (real-time systems) [ThatiRosu05,Basin et al 15]
  - Common assumptions
    - Reliable network with fully-connected components
    - Global clock
    - Oblivious to order of messages

(!) Unpredictable runtime behavior of rewriting

→ Hard to compare various strategies

RELATED WORK ↪ DECENTRALIZED RV (CONT'D)

- Automata-based techniques for regular languages [Falcone et al 14]

RELATED WORK ↪ DECENTRALIZED RV (CONT'D)

- Automata-based techniques for regular languages [Falcone et al 14]
    - Same assumptions as rewriting

RELATED WORK ↪ DECENTRALIZED RV (CONT'D)

- Automata-based techniques for regular languages [Falcone et al 14]
    - Same assumptions as rewriting
    - + More expressive than LTL

RELATED WORK ↪ DECENTRALIZED RV (CONT'D)

- Automata-based techniques for regular languages [Falcone et al 14]
  - Same assumptions as rewriting
  - \+ More expressive than LTL
  - \+ Predictable behavior

RELATED WORK $\hookrightarrow$ DECENTRALIZED RV (CONT'D)

- Automata-based techniques for regular languages [Falcone et al 14]
    - Same assumptions as rewriting
    - + More expressive than LTL
    - + Predictable behavior
    - – Tightly linked to specification (synthesis)

Related Work ↪ Decentralized RV (Cont'd)

- Automata-based techniques for regular languages [Falcone et al 14]
    - Same assumptions as rewriting
    - \+ More expressive than LTL
    - \+ Predictable behavior
    - – Tightly linked to specification (synthesis)
    - – No monitor topology nor communication strategy

Related Work ↪ Decentralized RV (Cont'd)

- Automata-based techniques for regular languages [Falcone et al 14]
    - Same assumptions as rewriting
    - + More expressive than LTL
    - + Predictable behavior
    - – Tightly linked to specification (synthesis)
    - – No monitor topology nor communication strategy
- Monitor Consensus [MostafaBonakdarpour16]

RELATED WORK ↪ DECENTRALIZED RV (CONT'D)

- Automata-based techniques for regular languages [Falcone et al 14]
    - Same assumptions as rewriting
    - \+ More expressive than LTL
    - \+ Predictable behavior
    - – Tightly linked to specification (synthesis)
    - – No monitor topology nor communication strategy
- Monitor Consensus [MostafaBonakdarpour16]
    - monitors deciding the same verdict

RELATED WORK ↪ DECENTRALIZED RV (CONT'D)

- Automata-based techniques for regular languages [Falcone et al 14]
    - Same assumptions as rewriting
    - \+ More expressive than LTL
    - \+ Predictable behavior
    - – Tightly linked to specification (synthesis)
    - – No monitor topology nor communication strategy
- Monitor Consensus [MostafaBonakdarpour16]
    - monitors deciding the same verdict
    - Assumptions

RELATED WORK ↪ DECENTRALIZED RV (CONT'D)

- Automata-based techniques for regular languages [Falcone et al 14]
    - Same assumptions as rewriting
    - \+ More expressive than LTL
    - \+ Predictable behavior
    - – Tightly linked to specification (synthesis)
    - – No monitor topology nor communication strategy
- Monitor Consensus [MostafaBonakdarpour16]
    - monitors deciding the same verdict
    - Assumptions
        - Fully-connected components

RELATED WORK ↪ DECENTRALIZED RV (CONT'D)

- Automata-based techniques for regular languages [Falcone et al 14]
  - Same assumptions as rewriting
  - \+ More expressive than LTL
  - \+ Predictable behavior
  - – Tightly linked to specification (synthesis)
  - – No monitor topology nor communication strategy
- Monitor Consensus [MostafaBonakdarpour16]
  - monitors deciding the same verdict
  - Assumptions
    - Fully-connected components
    - Asynchronous Systems (Alternating Numbers)

RELATED WORK ↪ DECENTRALIZED RV (CONT'D)

- Automata-based techniques for regular languages [Falcone et al 14]
  - Same assumptions as rewriting
  - \+ More expressive than LTL
  - \+ Predictable behavior
  - – Tightly linked to specification (synthesis)
  - – No monitor topology nor communication strategy
- Monitor Consensus [MostafaBonakdarpour16]
  - monitors deciding the same verdict
  - Assumptions
    - Fully-connected components
    - Asynchronous Systems (Alternating Numbers)
    - \+ Unreliable links (Monitors + System)

RELATED WORK ↪ DECENTRALIZED RV (CONT'D)

- Automata-based techniques for regular languages [Falcone et al 14]
    - Same assumptions as rewriting
    + More expressive than LTL
    + Predictable behavior
    – Tightly linked to specification (synthesis)
    – No monitor topology nor communication strategy
- Monitor Consensus [MostafaBonakdarpour16]
    - monitors deciding the same verdict
    - Assumptions
        - Fully-connected components
        - Asynchronous Systems (Alternating Numbers)
        + Unreliable links (Monitors + System)
        – $2k + 2$ verdicts when resilience up to k failures

Related Work ↪ Decentralized RV (cont'd)

- Automata-based techniques for regular languages [Falcone et al 14]
    - Same assumptions as rewriting
    + More expressive than LTL
    + Predictable behavior
    – Tightly linked to specification (synthesis)
    – No monitor topology nor communication strategy
- Monitor Consensus [MostafaBonakdarpour16]
    - monitors deciding the same verdict
    - Assumptions
        - Fully-connected components
        - Asynchronous Systems (Alternating Numbers)
        + Unreliable links (Monitors + System)
        – $2k + 2$ verdicts when resilience up to k failures
    → Determine consensus on a verdict in case of failures

RELATED WORK ↪ DECENTRALIZED RV (CONT'D)

- Automata-based techniques for regular languages [Falcone et al 14]
    - Same assumptions as rewriting
    - \+ More expressive than LTL
    - \+ Predictable behavior
    - – Tightly linked to specification (synthesis)
    - – No monitor topology nor communication strategy
- Monitor Consensus [MostafaBonakdarpour16]
    - monitors deciding the same verdict
    - Assumptions
        - Fully-connected components
        - Asynchronous Systems (Alternating Numbers)
        - \+ Unreliable links (Monitors + System)
        - – $2k + 2$ verdicts when resilience up to k failures
    - → Determine consensus on a verdict in case of failures

(!) All monitors check the same specification

Experiments

Studying Existing Algorithms ↪ Verifying Behavior

- Experiment Setup (5,868,800 runs)

Studying Existing Algorithms ↪ Verifying Behavior

- Experiment Setup (5,868,800 runs)
    - 200 synthetic random traces of 100 events (2 observations/component)

Studying Existing Algorithms $\hookrightarrow$ Verifying Behavior

- Experiment Setup (5,868,800 runs)
  - 200 synthetic random traces of 100 events (2 observations/component)
  - Vary $|\mathcal{C}|$ from 3 to 5

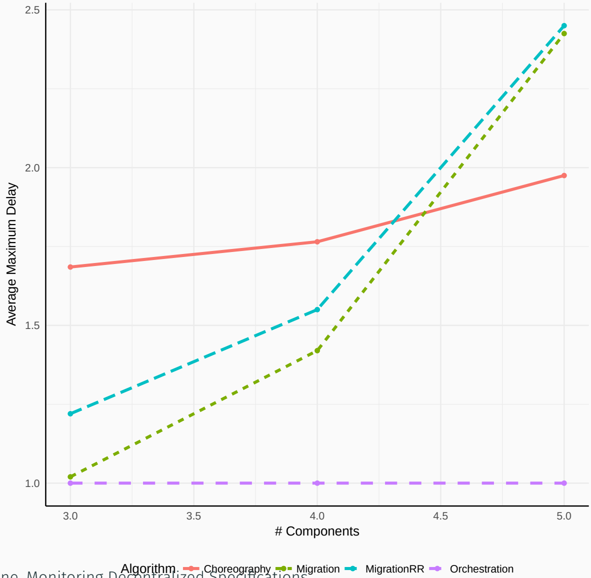
Studying Existing Algorithms ↪ Verifying Behavior

- Experiment Setup (5,868,800 runs)
    - 200 synthetic random traces of 100 events (2 observations/component)
    - Vary $|\mathcal{C}|$ from 3 to 5
    - At least 1,000 random specifications per scenario

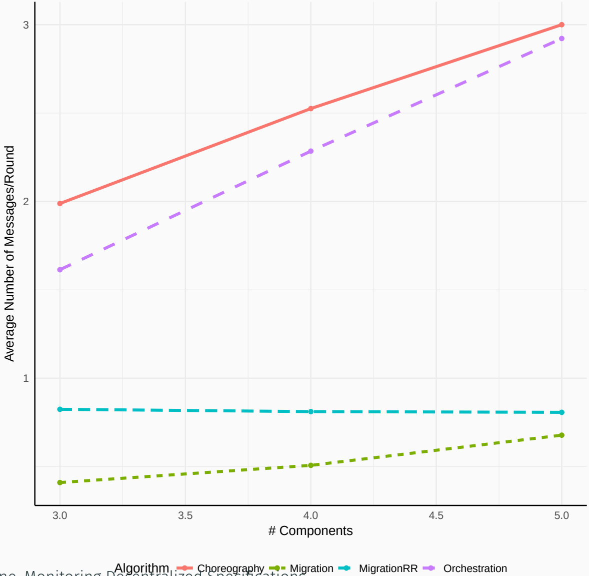
Studying Existing Algorithms ↪ Verifying Behavior

- Experiment Setup (5,868,800 runs)
  - 200 synthetic random traces of 100 events (2 observations/component)
  - Vary $|\mathcal{C}|$ from 3 to 5
  - At least 1,000 random specifications per scenario
  - Monitoring is done by rounds

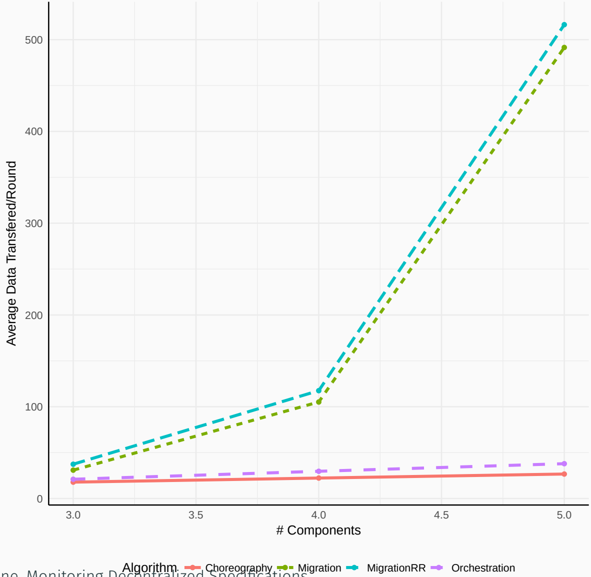
RESULTS ↪ DELAY



RESULTS ↪ Number of Messages



RESULTS ↪ DATA TRANSFERED



RESULTS

Alg.	$\lvert\mathcal{C}\rvert$	δ	#Msgs	Data	#S	#S/Mon	Conv
Chor	3	2.37	2.02	18.05	15.27	6.63	0.18
	4	2.49	2.54	22.62	18.22	6.79	0.20
	5	2.37	3.08	27.18	21.29	6.95	0.22
Migr	3	1.02	0.36	49.46	4.80	4.80	1.00
	4	1.38	0.41	128.26	5.67	5.67	1.00
	5	2.28	0.57	646.86	9.40	9.40	1.00
Migrr	3	1.09	0.86	58.02	5.00	5.00	1.00
	4	1.49	0.85	144.62	5.91	5.91	1.00
	5	2.32	0.83	684.81	9.60	9.60	1.00
Orch	3	0.63	1.68	21.01	4.13	4.13	1.00
	4	0.65	2.43	30.42	4.11	4.11	1.00
	5	0.81	3.04	38.51	5.55	5.55	1.00

Lower conv = more evenly distributed computation across monitors

RESULTS

Alg.	$\|\mathcal{C}\|$	δ	#Msgs	Data	#S	#S/Mon	Conv
Chor	3	2.37	2.02	18.05	15.27	6.63	0.18
	4	2.49	2.54	22.62	18.22	6.79	0.20
	5	2.37	3.08	27.18	21.29	6.95	0.22
Migr	3	1.02	0.36	49.46	4.80	4.80	1.00
	4	1.38	0.41	128.26	5.67	5.67	1.00
	5	2.28	0.57	646.86	9.40	9.40	1.00
Migrr	3	1.09	0.86	58.02	5.00	5.00	1.00
	4	1.49	0.85	144.62	5.91	5.91	1.00
	5	2.32	0.83	684.81	9.60	9.60	1.00
Orch	3	0.63	1.68	21.01	4.13	4.13	1.00
	4	0.65	2.43	30.42	4.11	4.11	1.00
	5	0.81	3.04	38.51	5.55	5.55	1.00

Lower conv = more evenly distributed computation across monitors

RESULTS

Alg.	$\|\mathcal{C}\|$	δ	#Msgs	Data	#S	#S/Mon	Conv
Chor	3	2.37	2.02	18.05	15.27	6.63	0.18
	4	2.49	2.54	22.62	18.22	6.79	0.20
	5	2.37	3.08	27.18	21.29	6.95	0.22
Migr	3	1.02	0.36	49.46	4.80	4.80	1.00
	4	1.38	0.41	128.26	5.67	5.67	1.00
	5	2.28	0.57	646.86	9.40	9.40	1.00
Migrr	3	1.09	0.86	58.02	5.00	5.00	1.00
	4	1.49	0.85	144.62	5.91	5.91	1.00
	5	2.32	0.83	684.81	9.60	9.60	1.00
Orch	3	0.63	1.68	21.01	4.13	4.13	1.00
	4	0.65	2.43	30.42	4.11	4.11	1.00
	5	0.81	3.04	38.51	5.55	5.55	1.00

Lower $\texttt{conv}$ = more evenly distributed computation across monitors

RESULTS

Alg.	$\|\mathcal{C}\|$	δ	#Msgs	Data	#S	#S/Mon	Conv
Chor	3	2.37	2.02	18.05	15.27	6.63	0.18
	4	2.49	2.54	22.62	18.22	6.79	0.20
	5	2.37	3.08	27.18	21.29	6.95	0.22
Migr	3	1.02	0.36	49.46	4.80	4.80	1.00
	4	1.38	0.41	128.26	5.67	5.67	1.00
	5	2.28	0.57	646.86	9.40	9.40	1.00
Migrr	3	1.09	0.86	58.02	5.00	5.00	1.00
	4	1.49	0.85	144.62	5.91	5.91	1.00
	5	2.32	0.83	684.81	9.60	9.60	1.00
Orch	3	0.63	1.68	21.01	4.13	4.13	1.00
	4	0.65	2.43	30.42	4.11	4.11	1.00
	5	0.81	3.04	38.51	5.55	5.55	1.00

Lower conv = more evenly distributed computation across monitors

RESULTS

Alg.	$\|\mathcal{C}\|$	δ	#Msgs	Data	#S	#S/Mon	Conv
Chor	3	2.37	2.02	18.05	15.27	6.63	0.18
	4	2.49	2.54	22.62	18.22	6.79	0.20
	5	2.37	3.08	27.18	21.29	6.95	0.22
Migr	3	1.02	0.36	49.46	4.80	4.80	1.00
	4	1.38	0.41	128.26	5.67	5.67	1.00
	5	2.28	0.57	646.86	9.40	9.40	1.00
Migrr	3	1.09	0.86	58.02	5.00	5.00	1.00
	4	1.49	0.85	144.62	5.91	5.91	1.00
	5	2.32	0.83	684.81	9.60	9.60	1.00
Orch	3	0.63	1.68	21.01	4.13	4.13	1.00
	4	0.65	2.43	30.42	4.11	4.11	1.00
	5	0.81	3.04	38.51	5.55	5.55	1.00

Lower conv = more evenly distributed computation across monitors

Soundness

Given a decentralized trace tr of length n, we reconstruct the global trace $\overline{e} = \rho(\text{tr}) = e_0 \cdot \ldots \cdot e_n$, we have: $\Delta^*(q_0, \overline{e}) = \text{sel}(\mathcal{I}^n, \mathcal{M}^n, n)$, with:

$$\begin{aligned} \mathcal{I}^n &= \text{mov}([0 \mapsto q_0 \mapsto \top], 0, n), \text{and} \\ \mathcal{M}^n &= \biguplus^2_{t \in [1,n]} \{\text{memc}(e_t, \text{ts}_t)\}. \end{aligned}$$

Convergence

$$\text{convergence} = \frac{1}{n}\sum_{t=1}^{n}\left(\sum_{c\in\mathcal{C}}\left(\frac{s_c^t}{\overline{s}^t} - \frac{1}{|\mathcal{C}|}\right)^2\right)\ \text{, with } \overline{s}^t = \sum_{c\in\mathcal{C}} s_c^t$$

STUDYING EXISTING ALGORITHMS

- Example Algorithms

STUDYING EXISTING ALGORITHMS

- Example Algorithms
  - Orchestration: Central monitor + forwarding monitors

STUDYING EXISTING ALGORITHMS

- Example Algorithms
    - Orchestration: Central monitor + forwarding monitors
    - Migration: Specification hops from one component to another

STUDYING EXISTING ALGORITHMS

- Example Algorithms
    - Orchestration: Central monitor + forwarding monitors
    - Migration: Specification hops from one component to another
    - Choreography: Monitors are organized in a tree

STUDYING EXISTING ALGORITHMS

- Example Algorithms
    - Orchestration: Central monitor + forwarding monitors
    - Migration: Specification hops from one component to another
    - Choreography: Monitors are organized in a tree

- Expected behavior of algorithms

Algorithm	δ	# Msg	$\lvert\text{Msg}\rvert$
Orchestration	$\Theta(1)$	$\Theta(\lvert\mathcal{C}\rvert)$	$O(\lvert AP_c\rvert)$
Migration	$O(\lvert\mathcal{C}\rvert)$	$O(m)$	$O(m\lvert\mathcal{C}\rvert^2)$
Choreography	$O(\text{depth}(\text{m}_{\text{root}}))$	$\Theta(\lvert E\rvert)$	$\Theta(1)$